DNSSEC und DANE Einführungskurs

Sven Duscha und Bernhard Schmidt

DNSSEC/DANE Kurs am LRZ

Kurze Vorstellungsrunde

- Name, Universität / Hochschule
- Vorwissen über Kryptographie?
- Kenntnisse und Erfahrung mit DNS, DNSSEC?
- Was erwarten Sie sich von diesem Kurs?

Inhaltsübersicht

DNSSEC

- DNS Server - Hierarchische Abfragen
- DNS Funktionsweise und Schwachpunkte
- Public Key Infrastructure (PKI) Grundlagen
- DNSSEC Records und Zusammenhänge
- DNSSEC-Konfiguration am Beispiel BIND-9.9
- DNSSEC in der Praxis
- Zusammenfassung DNSSEC

Inhaltsübersicht

- DANE - Domain name-based authenticated named entity
- DANE Funktionsweise
- Anwendungen von DANE
- Beispiel Mailserver-Authentizität garantieren
- Zusammenfassung DANE

DNS - Funktionsweise und Schwachpunkte

DNS Funktionsweise

- DNS - Domain name system ordnet die IP Adressen Domainnamen zu (und umgekehrt)
- DNS ist dezentral, keine zentrale Datenbank
- Jeder Nameserver verwaltet seine Zone
- Abfragen durchlaufen hierarchisch den Domain tree



129.187.4.40

2001:4ca0:0:305::40

root NS → TLD NS → Domain NS → IP des Servers

- Autoritative und Resolving Nameserver
- Master und Slave Nameserver

DNS Nameserver Zusammenspiel

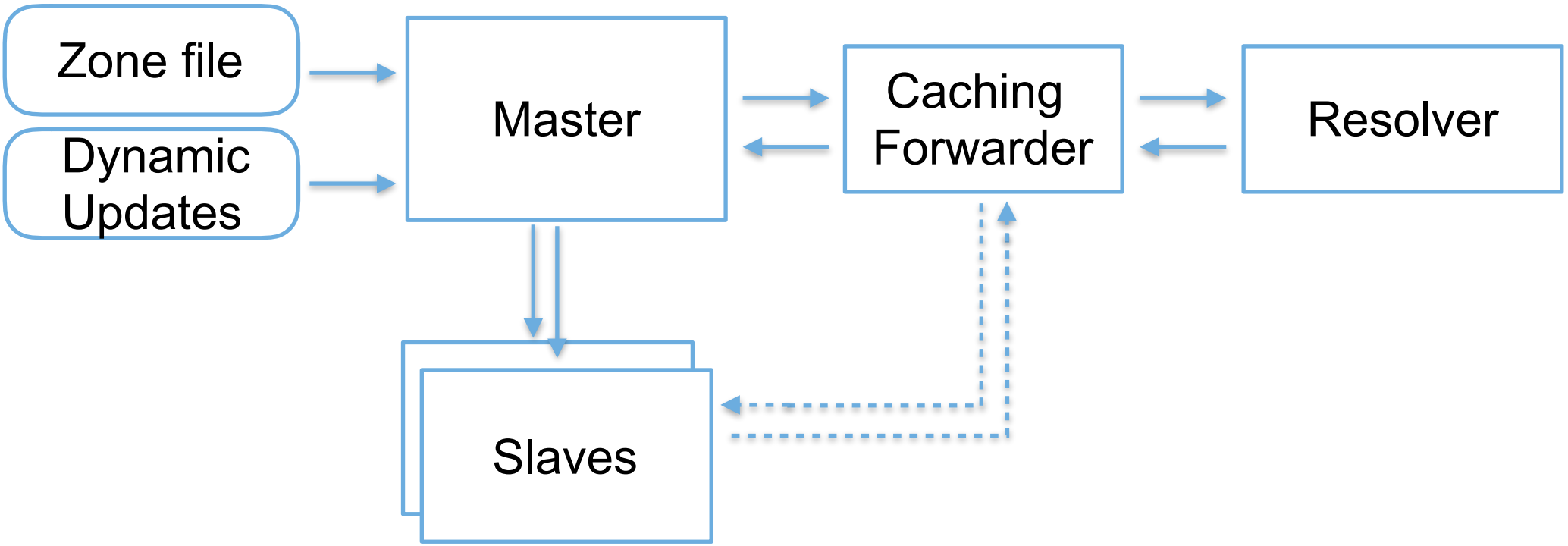


Bild: RIPE NCC

DNS Abfrage-Beispiel: confluence.lrz.de

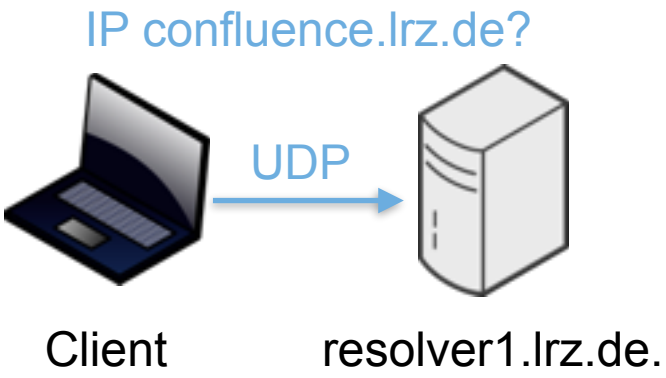
IP confluence.lrz.de?

1. Benutzer will auf: confluence.lrz.de

Client

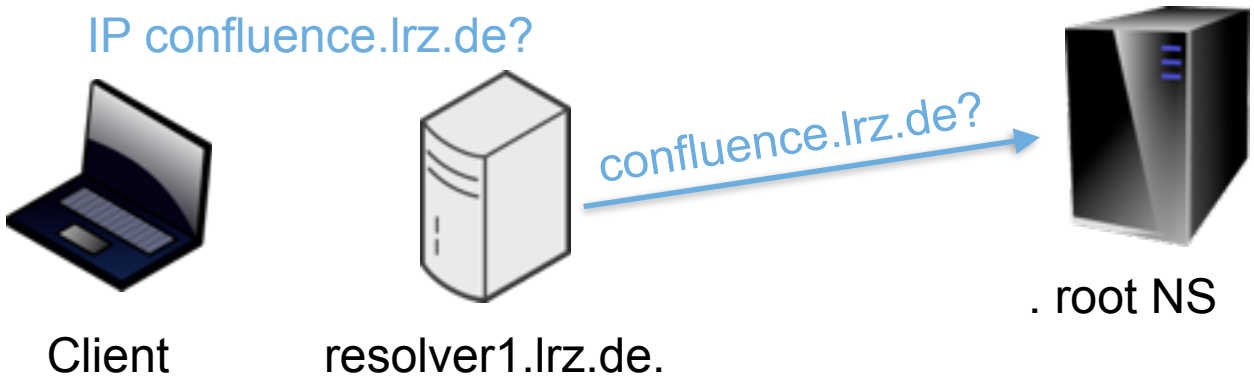
DNS Abfrage-Beispiel: confluence.lrz.de

1. Benutzer will auf: confluence.lrz.de
2. Browser fragt 10.156.33.53, resolver1.lrz.de



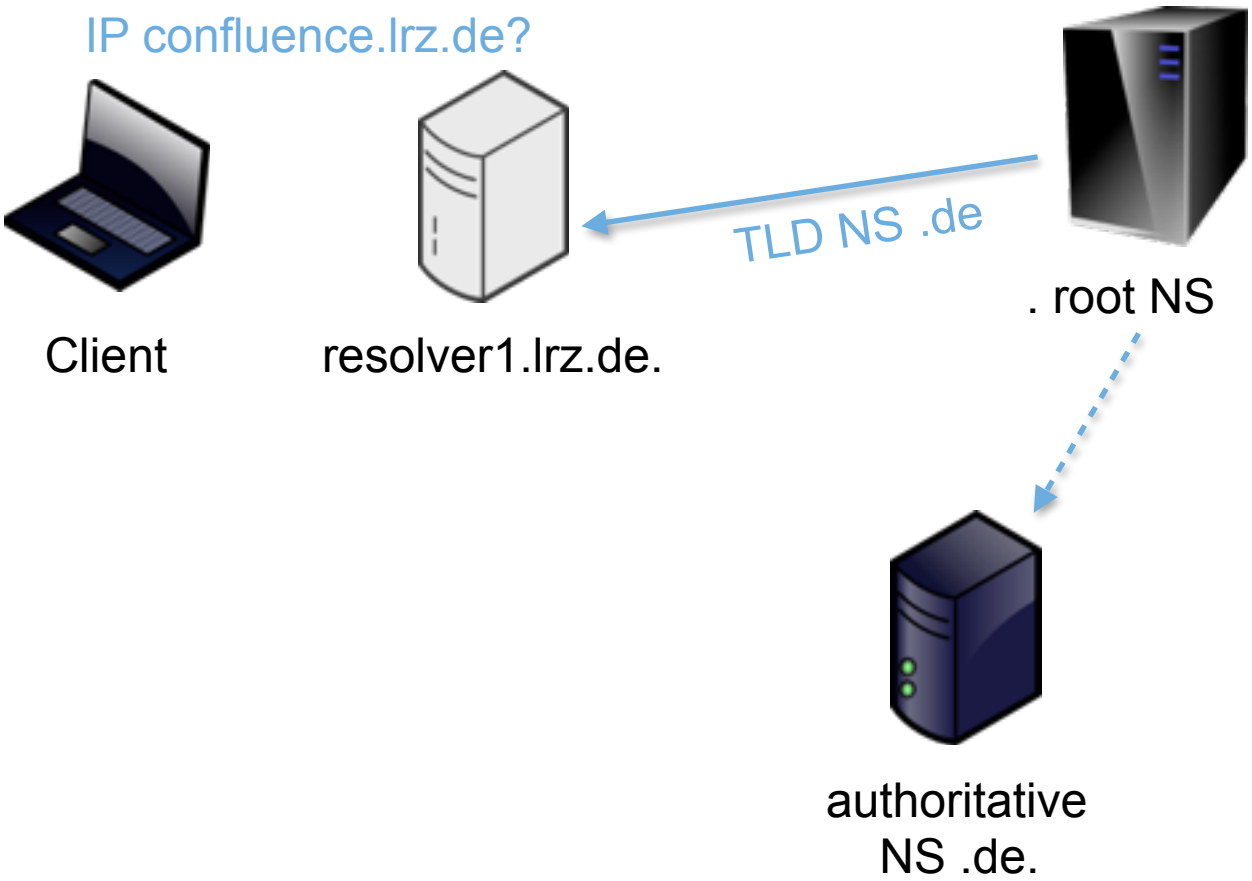
DNS Abfrage-Beispiel: confluence.lrz.de

1. Benutzer will auf: confluence.lrz.de
2. Browser fragt 10.156.33.53, resolver1.lrz.de
3. Resolving DNS fragt root Nameserver



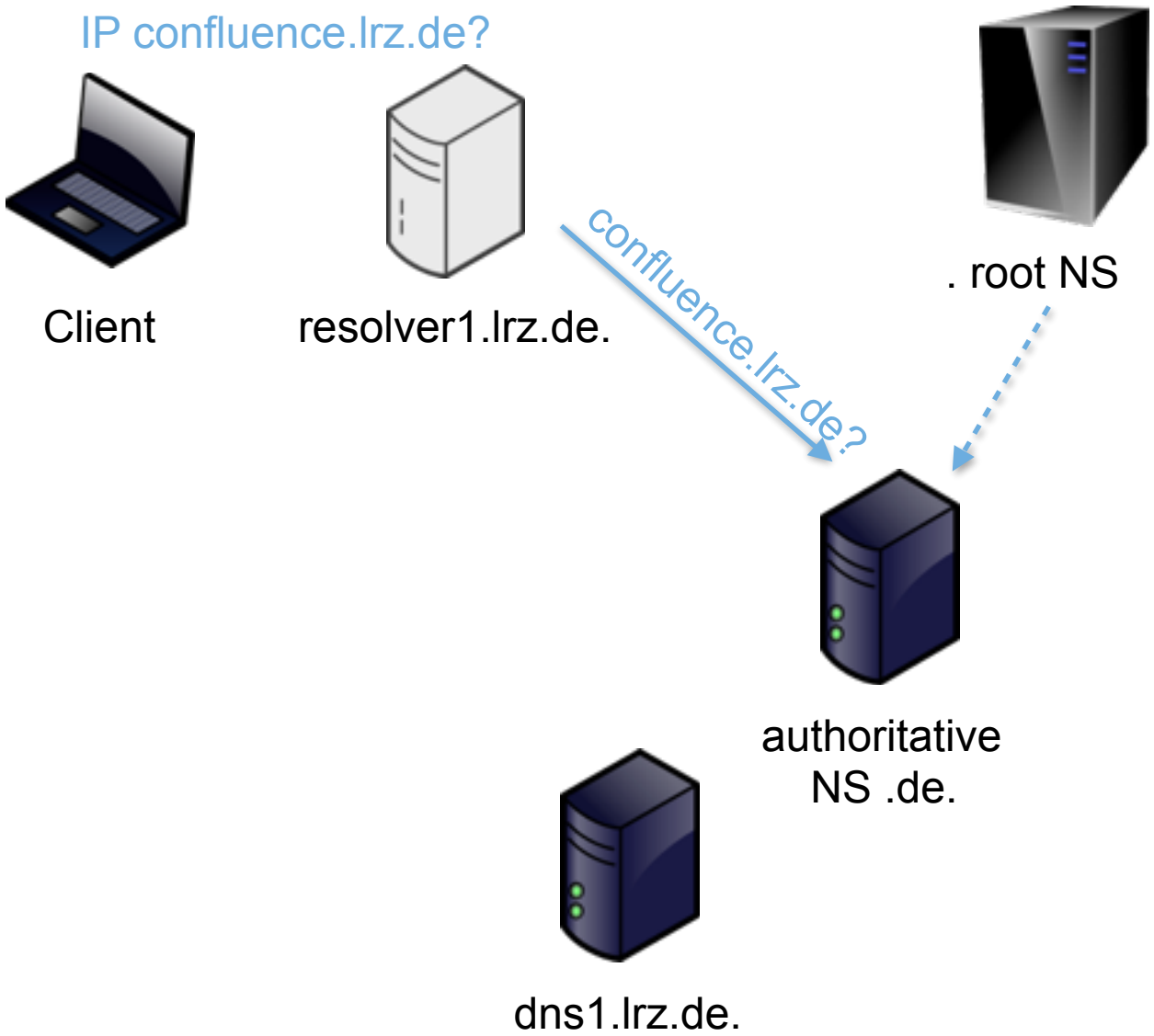
DNS Abfrage-Beispiel: confluence.lrz.de

1. Benutzer will auf: confluence.lrz.de
2. Browser fragt 10.156.33.53, resolver1.lrz.de
3. Resolving DNS fragt root Nameserver
4. Verweis auf .de. TLD authoritative Nameserver



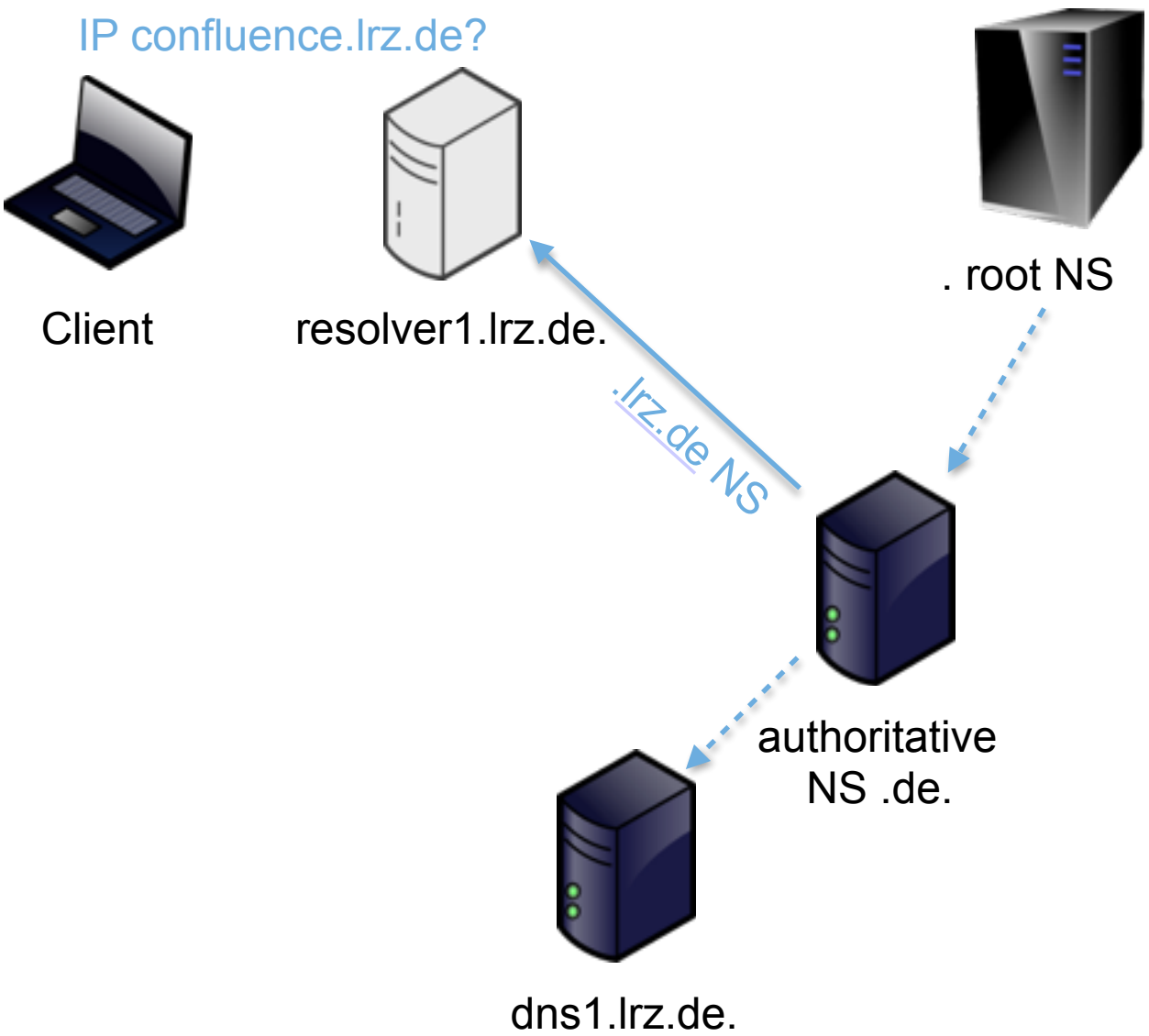
DNS Abfrage-Beispiel: confluence.lrz.de

1. Benutzer will auf: confluence.lrz.de
2. Browser fragt 10.156.33.53, resolver1.lrz.de
3. Resolving DNS fragt root Nameserver
4. Verweis auf .de. TLD authoritative Nameserver
5. Resolver frage .de. TLD NS



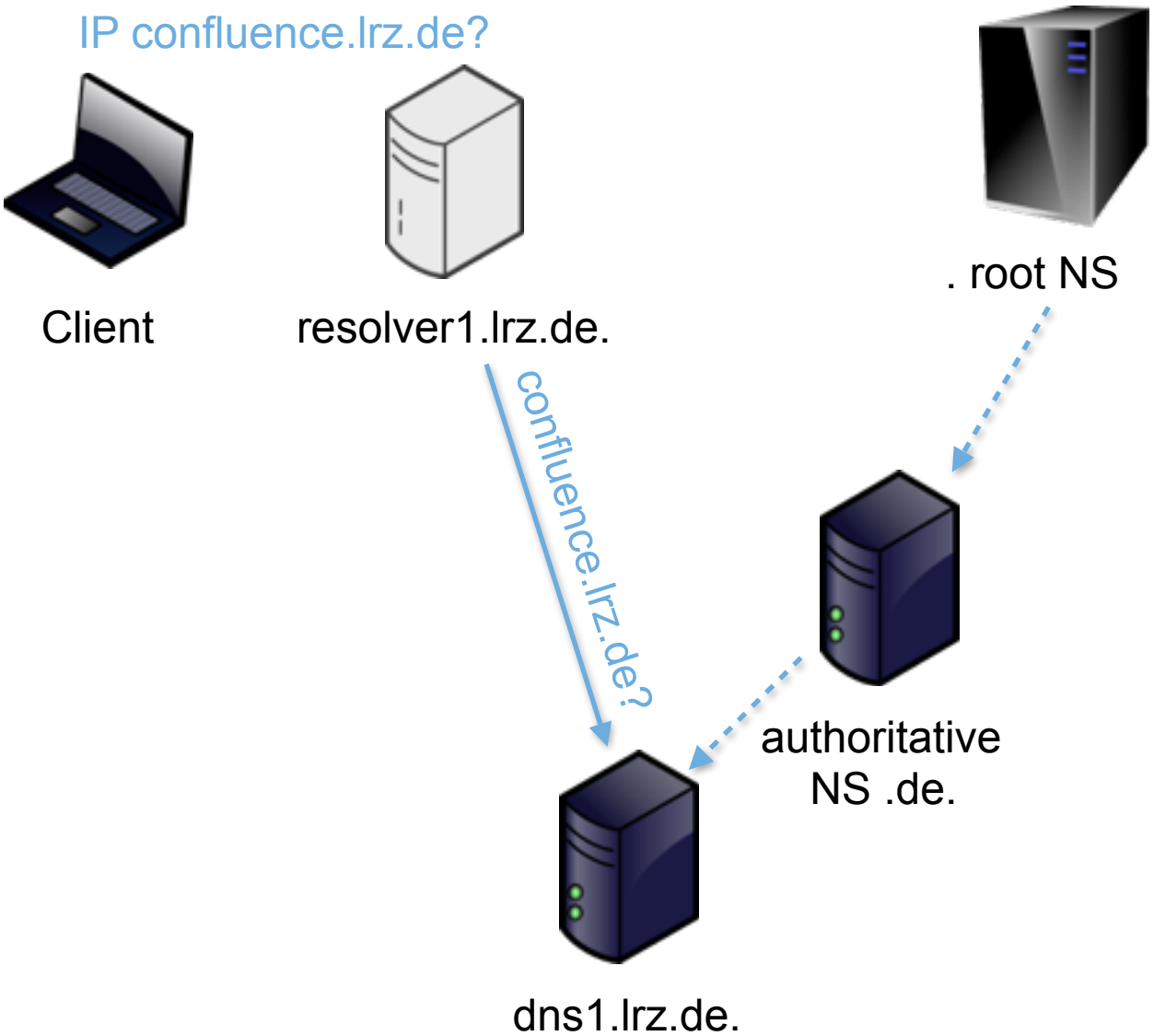
DNS Abfrage-Beispiel: confluence.lrz.de

1. Benutzer will auf: confluence.lrz.de
2. Browser fragt 10.156.33.53, resolver1.lrz.de
3. Resolving DNS fragt root Nameserver
4. Verweis auf .de. TLD authoritative Nameserver
5. Resolver frage .de. TLD NS
6. Verweis auf LRZ NS dns1.lrz.de



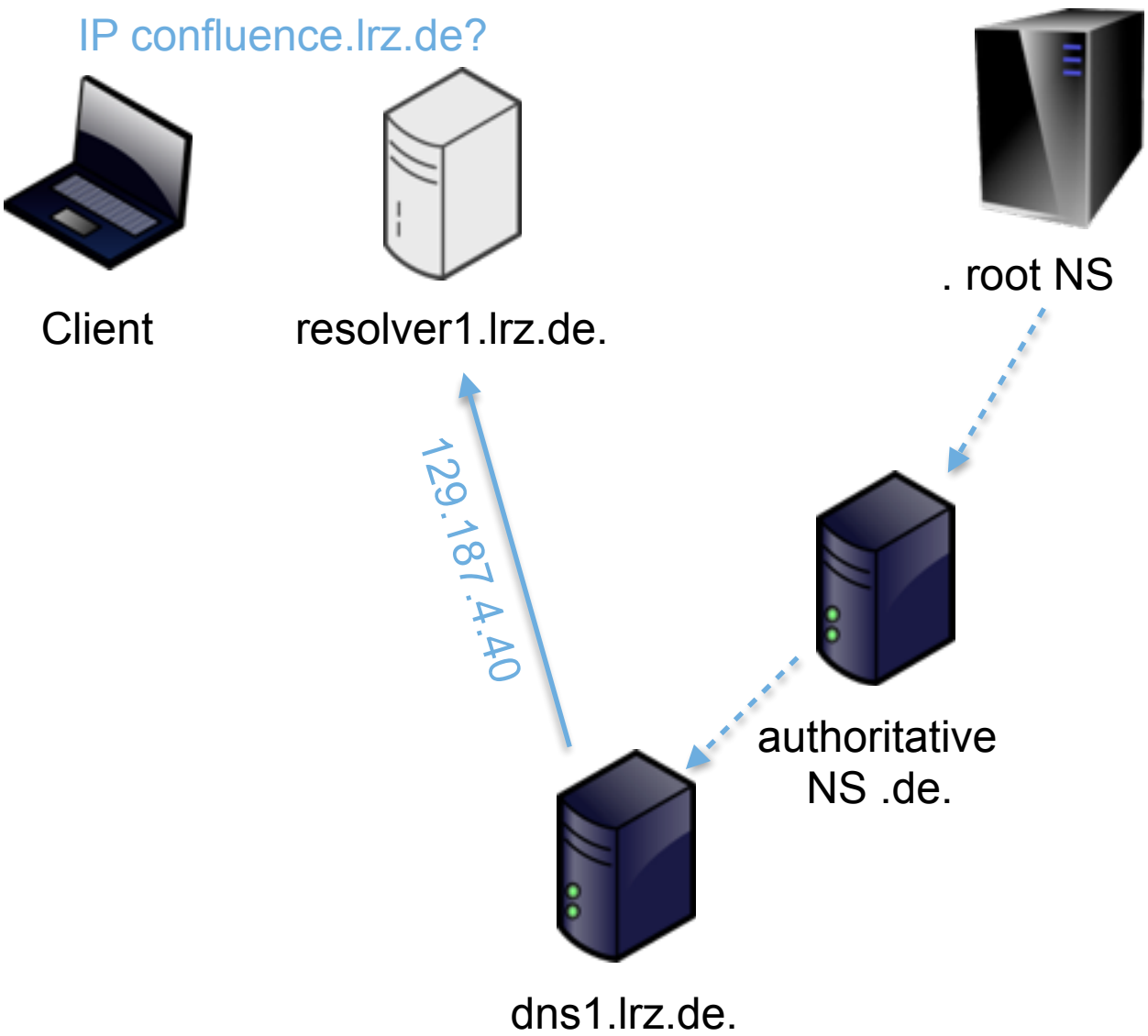
DNS Abfrage-Beispiel: confluence.lrz.de

1. Benutzer will auf: confluence.lrz.de
2. Browser fragt 10.156.33.53, resolver1.lrz.de
3. Resolving DNS fragt root Nameserver
4. Verweis auf .de. TLD authoritative Nameserver
5. Resolver frage .de. TLD NS
6. Verweis auf LRZ NS dns1.lrz.de
7. Resolver fragt dns1.lrz.de.



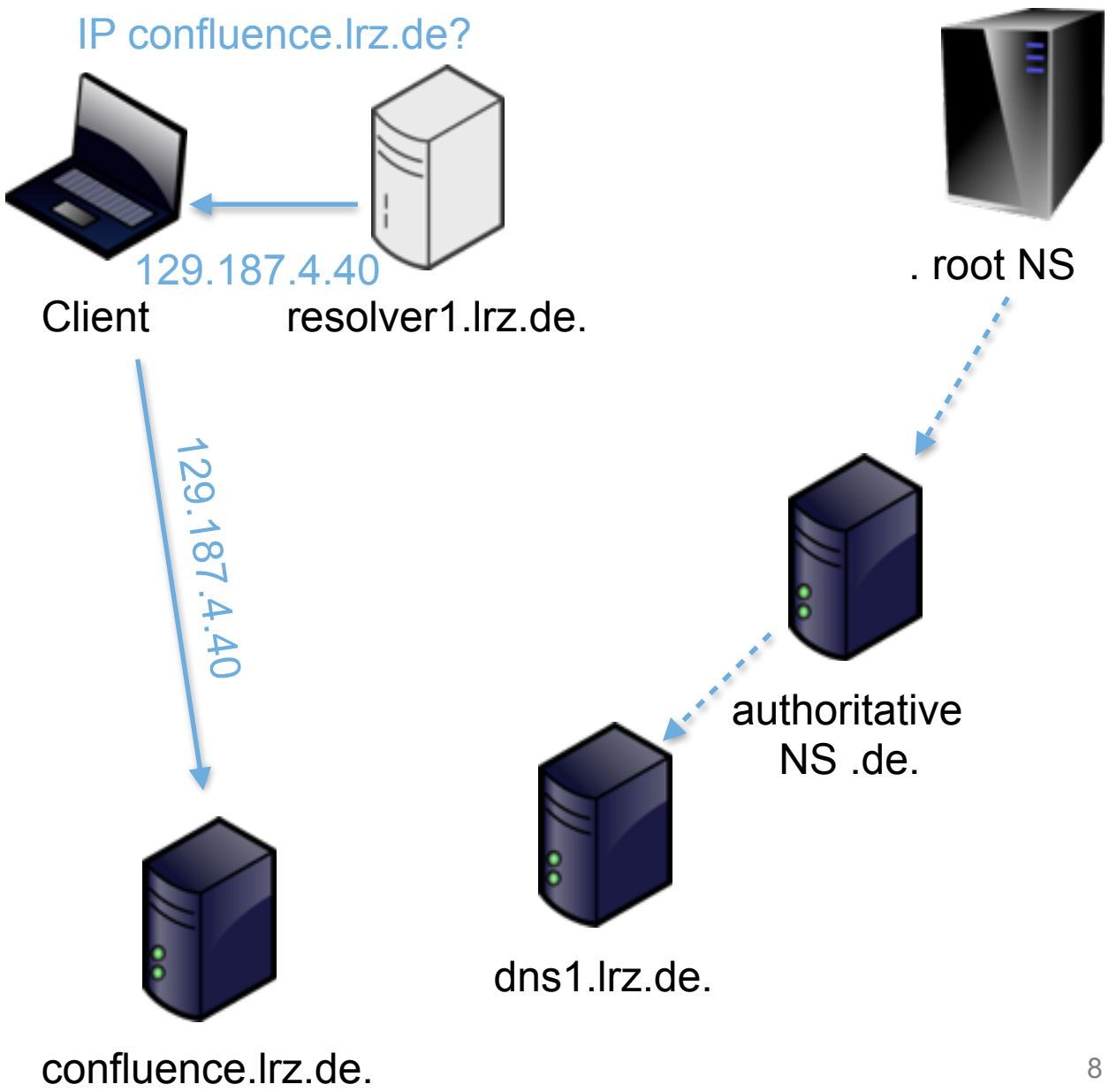
DNS Abfrage-Beispiel: confluence.lrz.de

1. Benutzer will auf: confluence.lrz.de
2. Browser fragt 10.156.33.53, resolver1.lrz.de
3. Resolving DNS fragt root Nameserver
4. Verweis auf .de. TLD authoritative Nameserver
5. Resolver frage .de. TLD NS
6. Verweis auf LRZ NS dns1.lrz.de
7. Resolver fragt dns1.lrz.de.
8. IP zu confluence.lrz.de.
   129.187.4.40
   2001:4ca0:0:305::40



DNS Abfrage-Beispiel: confluence.lrz.de

1. Benutzer will auf: confluence.lrz.de
2. Browser fragt 10.156.33.53, resolver1.lrz.de
3. Resolving DNS fragt root Nameserver
4. Verweis auf .de. TLD authoritative Nameserver
5. Resolver frage .de. TLD NS
6. Verweis auf LRZ NS dns1.lrz.de
7. Resolver fragt dns1.lrz.de.
8. IP zu confluence.lrz.de.
   129.187.4.40
   2001:4ca0:0:305::40
9. IP Antwort an Client



DNS - Grundsätzliche Probleme

- DNS ist plain text
- UDP keine sessions
- Baumstruktur mit Weiterreichen der Verantwortlichkeit

  → jede Instanz ist für ihre Zone zuständig
- Basiert auf gegenseitigem Vertrauen
- Resolving Nameserver sind Opfer von Fehlern, Übernahmen und Angriffen

DNS Nameserver - Angriffspunkte

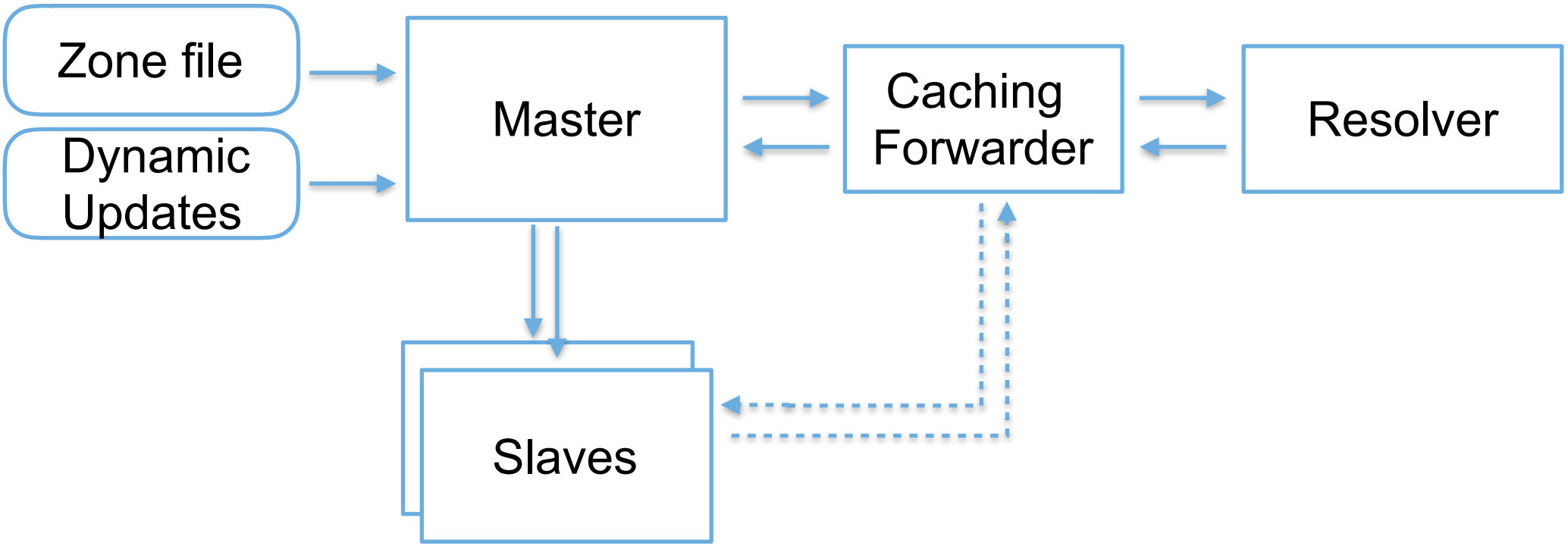


Bild: RIPE NCC

DNS Nameserver - Angriffspunkte

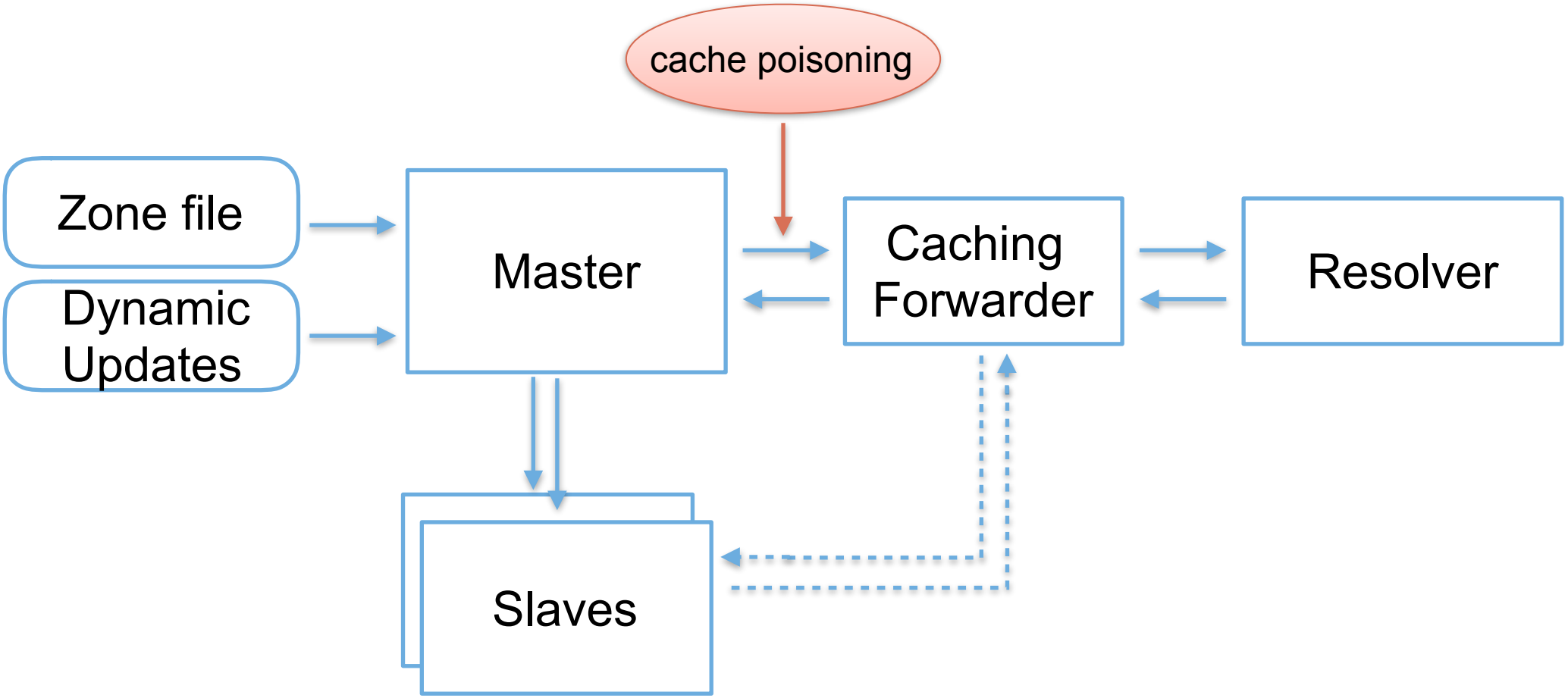


Bild: RIPE NCC

DNS Nameserver - Angriffspunkte

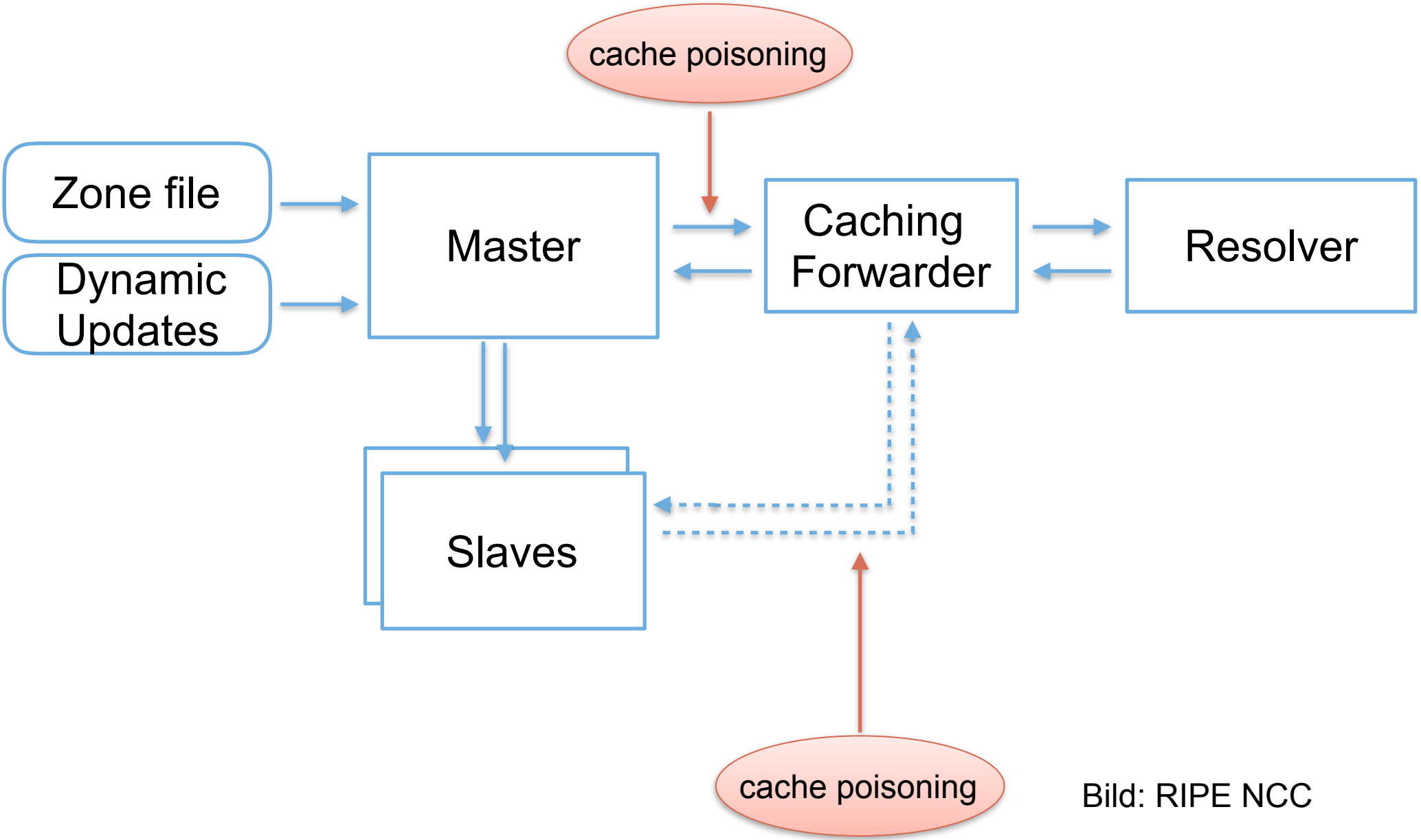


Bild: RIPE NCC

DNS Nameserver - Angriffspunkte

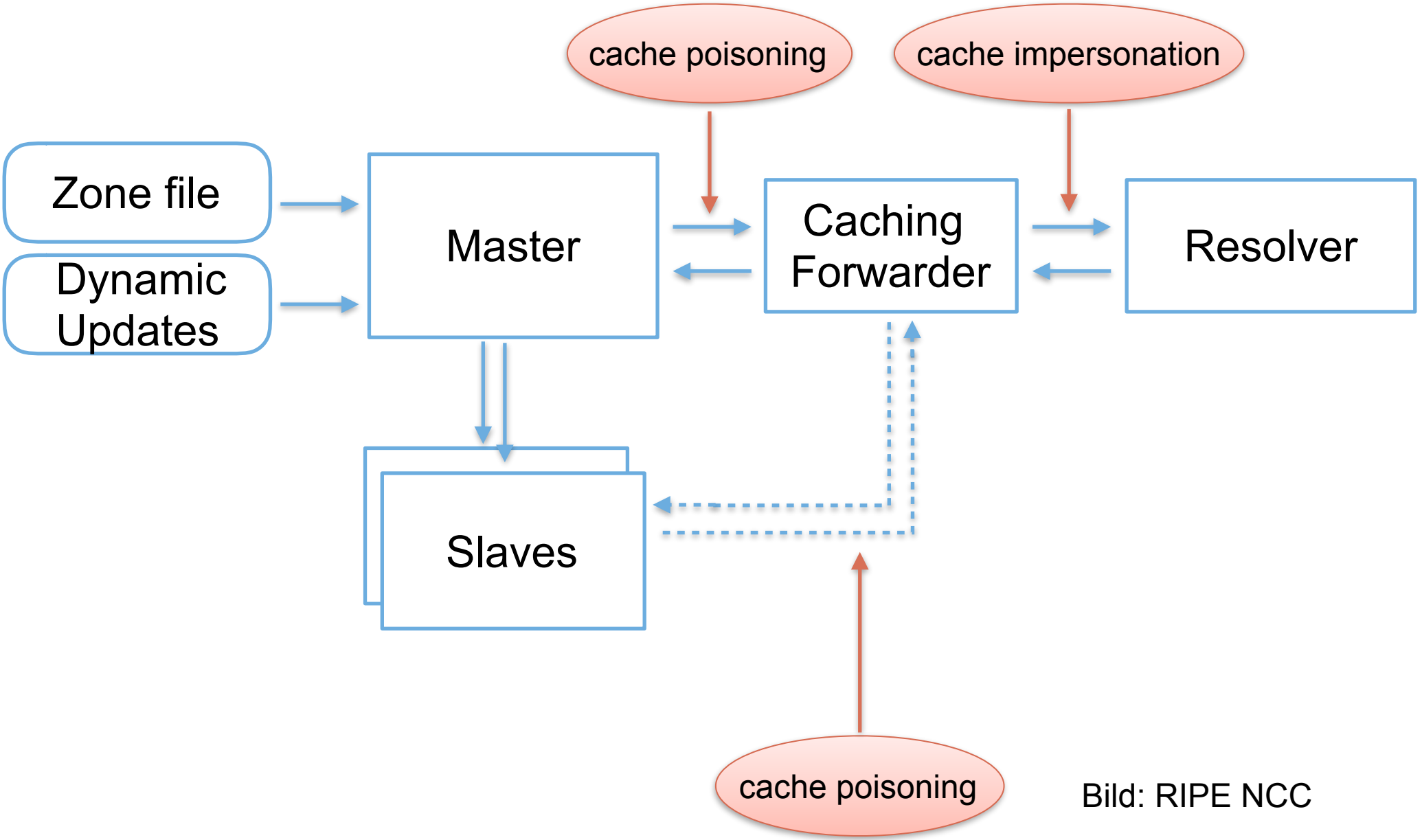


Bild: RIPE NCC

DNS Nameserver - Angriffspunkte

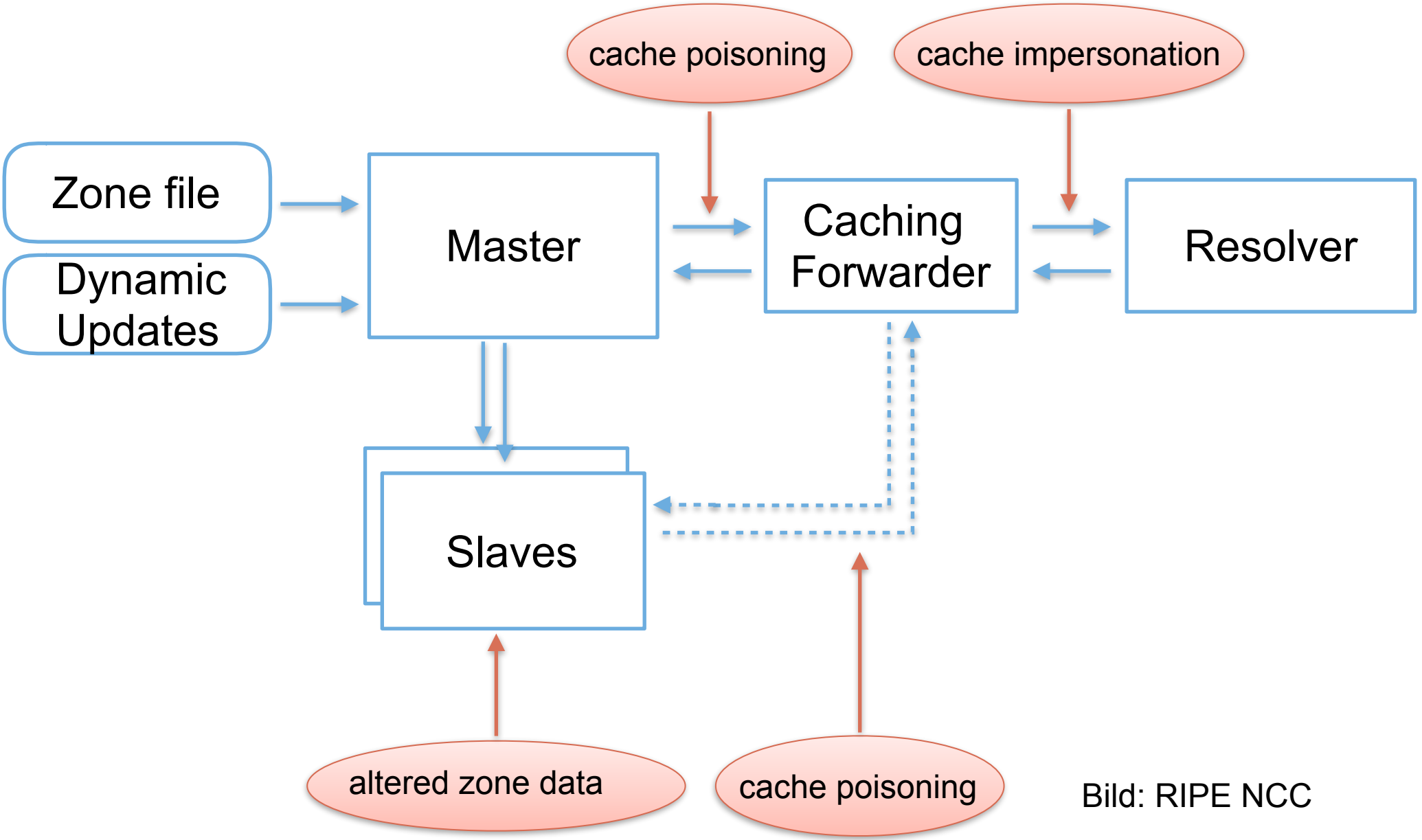


Bild: RIPE NCC

Angriffsvektoren auf DNS Abfragen/Server

- Cache impersonation
  = (Spoofed Address attack / Man-in-the-middle attack)
- Cache poisoning
- Geänderte Zonendateien auf Slaves
- TCP SYN Flood Attacks / UDP Flood Attack
- Zone Walking

DNS Spoofing - wie?

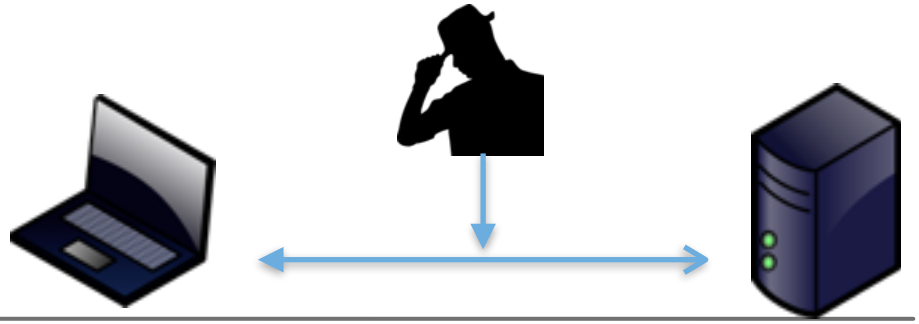
- DNS hauptsächlich über (verbindungsloses) UDP
- Zuordnung von Anfrage und Antwort über Matching verschiedener Protokollfelder
  - IP-Adresse beider Seiten (2-3 Bit Zufall)
  - Quellport Anfrage bzw. Zielport Antwort (15-16 Bit Zufall)
  - Anfrage (kein Zufall)
  - DNS Transaction-ID (16 Bit Zufall)

  im besten Fall etwa 2^{34} Möglichkeiten
- mit heutigen Bandbreiten ist Brute-Force Spoofing realistisch
- Implementierungs- und Konfigurationsfehler reduzieren Zufall oder ermöglichen neue Angriffe (Kaminsky-Attacke, out-of-bailiwick)
- wenn ein Angreifer Anfragen (passiv) mithört kann er fast immer schneller antworten als der richtige Server

DNS Spoofing - warum?

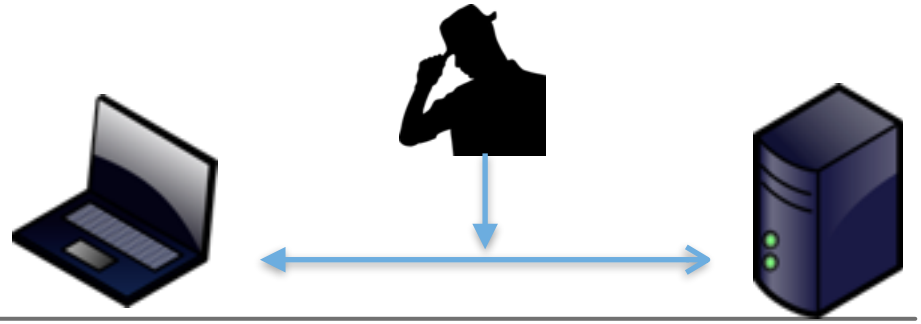
- Platzieren falscher Einträge in Resolvercache, zum Teil für ganze TLDs
- kein Selbstzweck, sondern Vorbereitung für
  - Man-in-the-Middle Angriffen (Lenovo Superfish, älter: Apple Updates)
  - Advertisement Fraud
  - Drive-by Angriffe von Malware
  - Hijacking von SMTP, dann Abgreifen von Passwortresetmails
  - ...
- schwer zu erkennen, noch schwerer zu verhindern (BCP-38?)
- betrifft unter Umständen tausende von Nutzern auf einen Schlag

Man-in-middle attack



- Angreifer sendet ein IP Paket mit der Adresse des DNS-Servers als Absender
- Muss die korrekte DNS-ID-Nummer verwenden
- ID der Anfrage sniffen, muß aber schneller als der echte DNS NS antworten

Man-in-middle attack



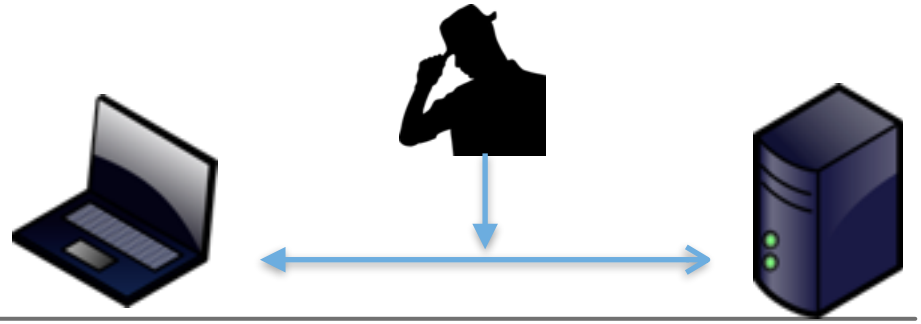
- Angreifer sendet ein IP Paket mit der Adresse des DNS-Servers als Absender
- Muss die korrekte DNS-ID-Nummer verwenden
- ID der Anfrage sniffen, muß aber schneller als der echte DNS NS antworten

Client

echter
DNS NS

Lokales Netzwerk

Man-in-middle attack



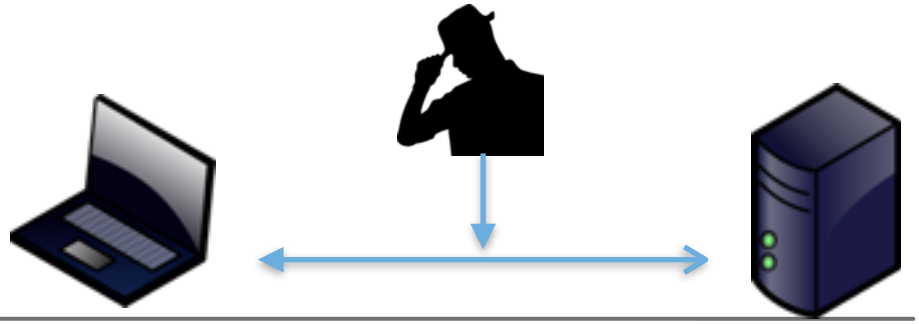
- Angreifer sendet ein IP Paket mit der Adresse des DNS-Servers als Absender
- Muss die korrekte DNS-ID-Nummer verwenden
- ID der Anfrage sniffen, muß aber schneller als der echte DNS NS antworten

Client

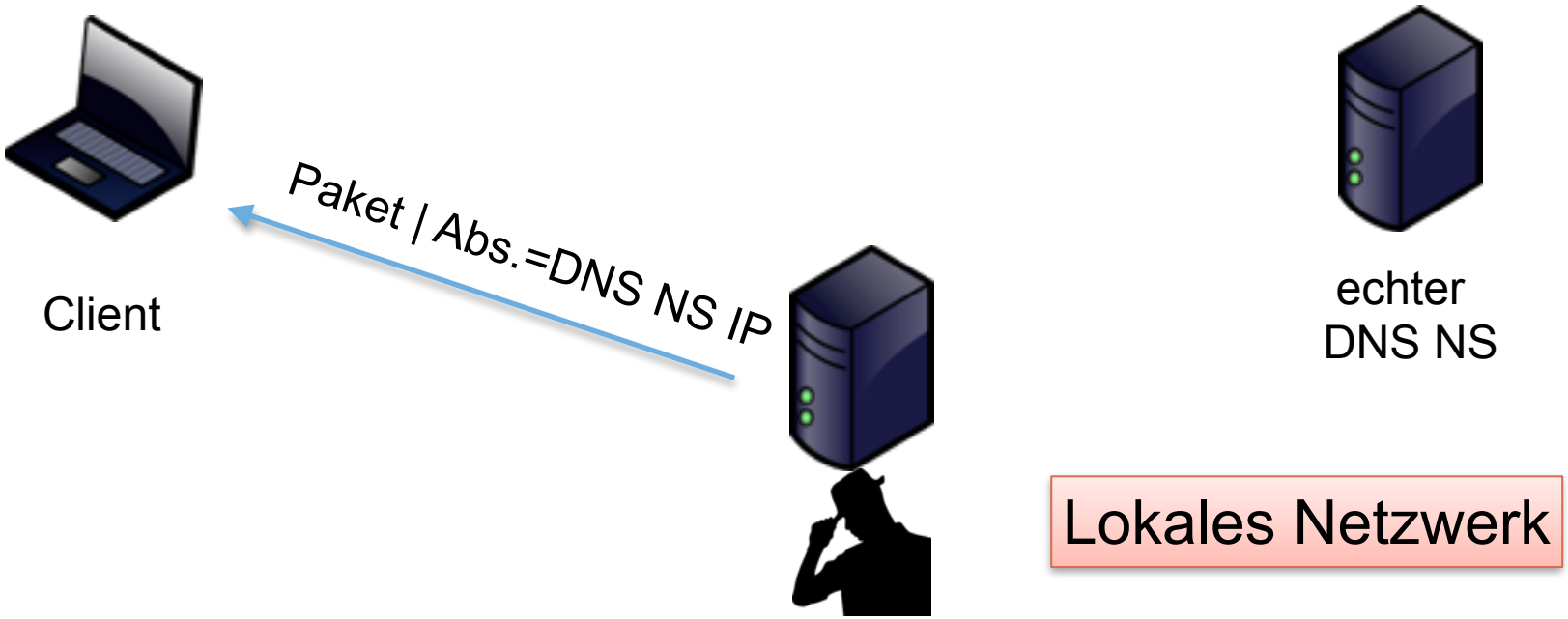
echter
DNS NS

Lokales Netzwerk

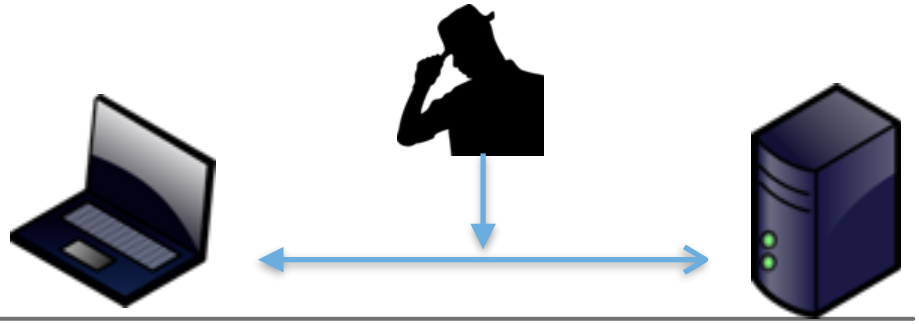
Man-in-middle attack



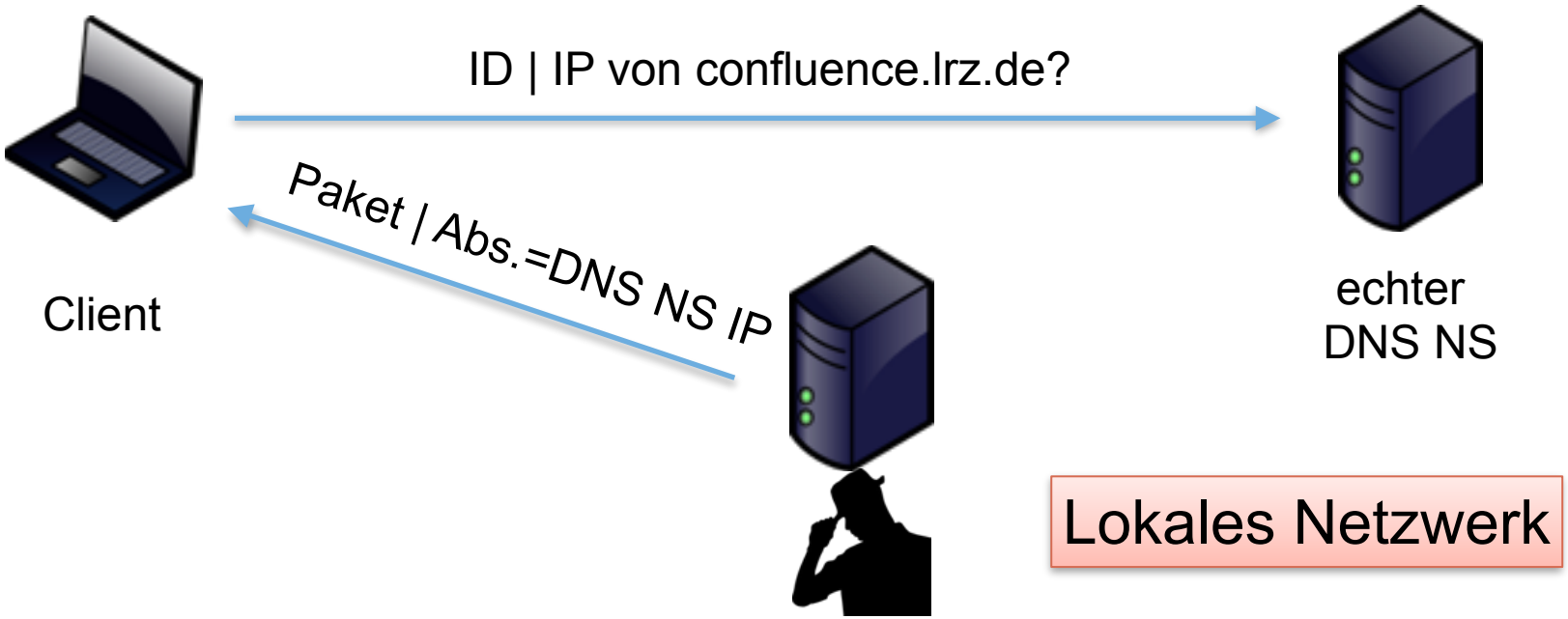
- Angreifer sendet ein IP Paket mit der Adresse des DNS-Servers als Absender
- Muss die korrekte DNS-ID-Nummer verwenden
- ID der Anfrage sniffen, muß aber schneller als der echte DNS NS antworten



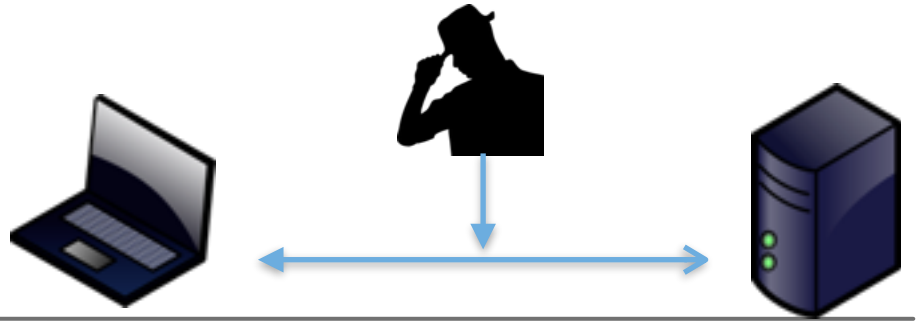
Man-in-middle attack



- Angreifer sendet ein IP Paket mit der Adresse des DNS-Servers als Absender
- Muss die korrekte DNS-ID-Nummer verwenden
- ID der Anfrage sniffen, muß aber schneller als der echte DNS NS antworten

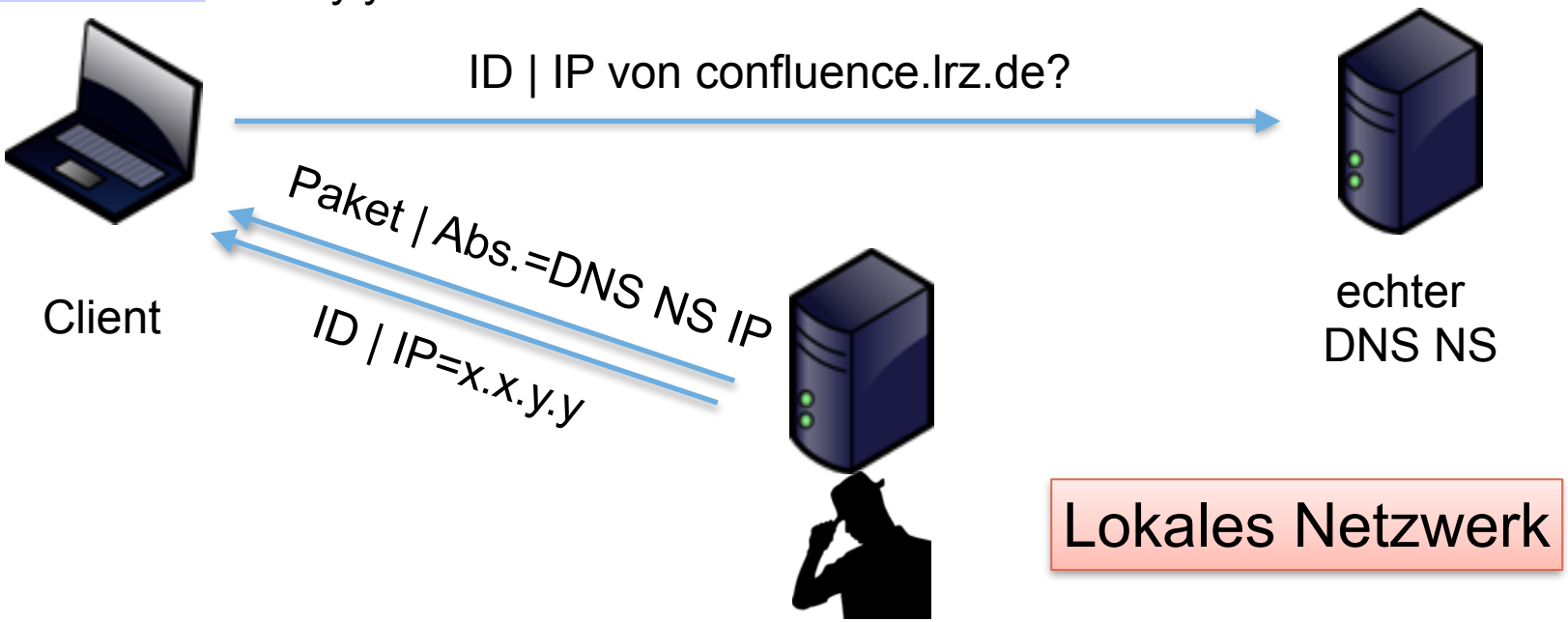


Man-in-middle attack

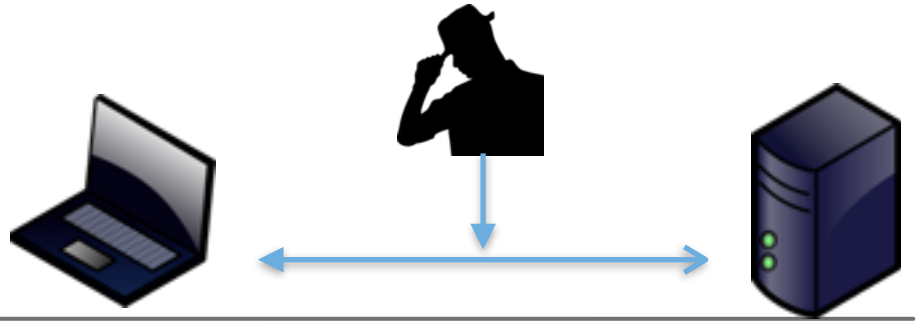


- Angreifer sendet ein IP Paket mit der Adresse des DNS-Servers als Absender
- Muss die korrekte DNS-ID-Nummer verwenden
- ID der Anfrage sniffen, muß aber schneller als der echte DNS NS antworten

confluence.lrz.de = x.x.y.y

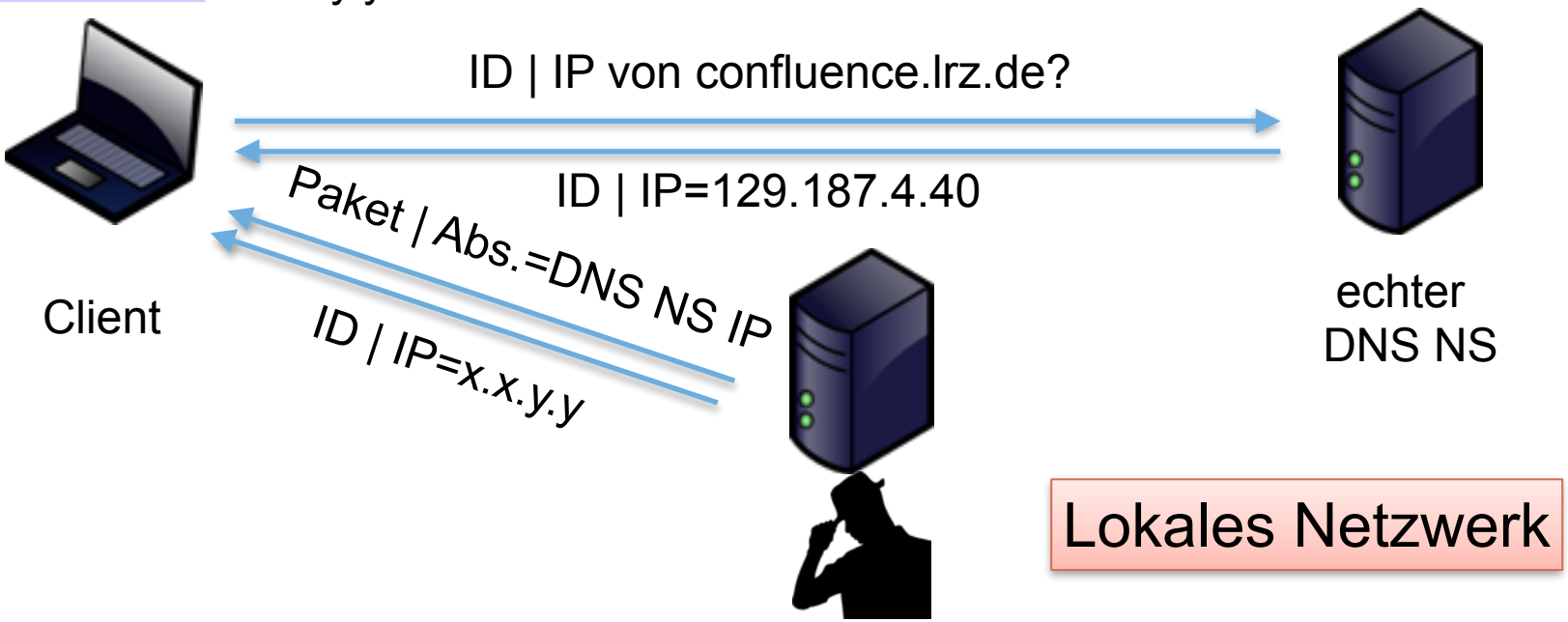


Man-in-middle attack

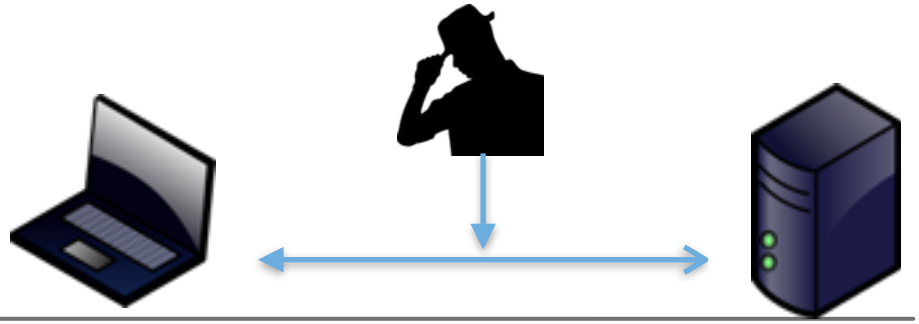


- Angreifer sendet ein IP Paket mit der Adresse des DNS-Servers als Absender
- Muss die korrekte DNS-ID-Nummer verwenden
- ID der Anfrage sniffen, muß aber schneller als der echte DNS NS antworten

confluence.lrz.de = x.x.y.y

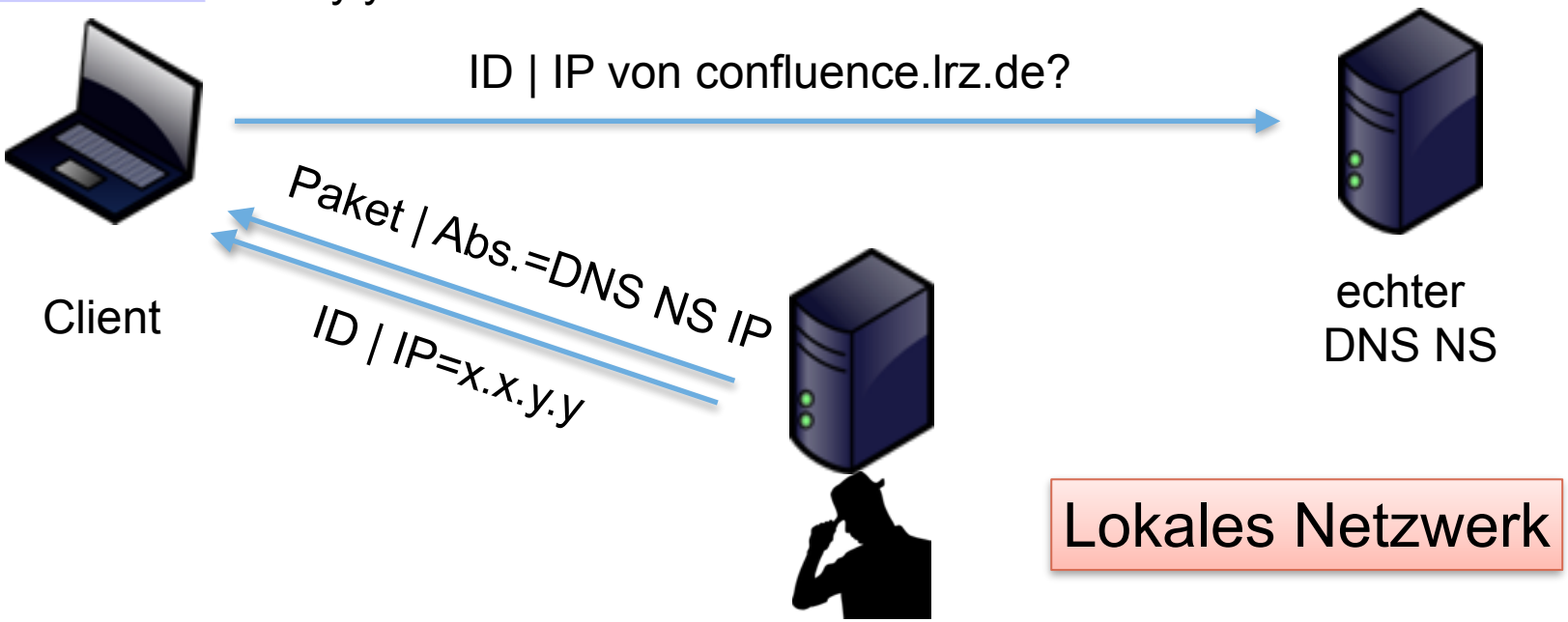


Man-in-middle attack

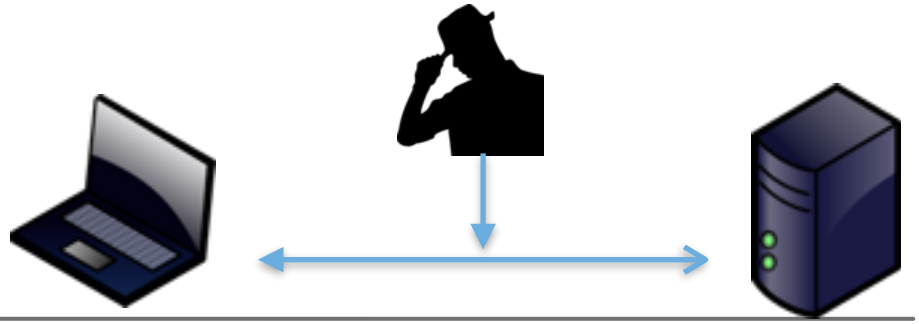


- Angreifer sendet ein IP Paket mit der Adresse des DNS-Servers als Absender
- Muss die korrekte DNS-ID-Nummer verwenden
- ID der Anfrage sniffen, muß aber schneller als der echte DNS NS antworten

confluence.lrz.de = x.x.y.y

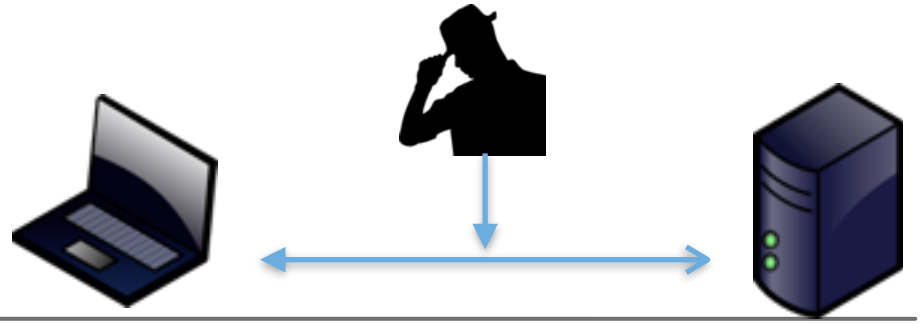


Man-in-middle attack



- Pakete mit zufälligen (allen) möglichen IDs in Paketen
- ~N x 100 Antworten senden, um die ID mit hoher Wahrscheinlichkeit zu treffen
- Einige Nameserver erhöhen einfach die ID um 1 von aufeinander folgenden Antworten auf Anfragen

Man-in-middle attack



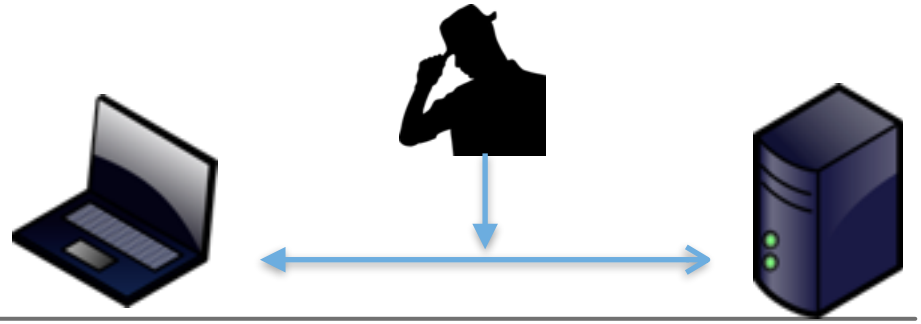
- Pakete mit zufälligen (allen) möglichen IDs in Paketen
- ~N x 100 Antworten senden, um die ID mit hoher Wahrscheinlichkeit zu treffen
- Einige Nameserver erhöhen einfach die ID um 1 von aufeinander folgenden Antworten auf Anfragen

Client

echter
DNS NS

Internet

Man-in-middle attack



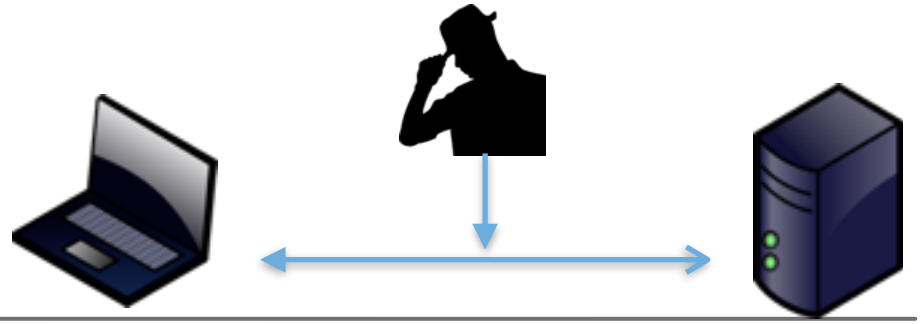
- Pakete mit zufälligen (allen) möglichen IDs in Paketen
- ~N x 100 Antworten senden, um die ID mit hoher Wahrscheinlichkeit zu treffen
- Einige Nameserver erhöhen einfach die ID um 1 von aufeinander folgenden Antworten auf Anfragen

Client

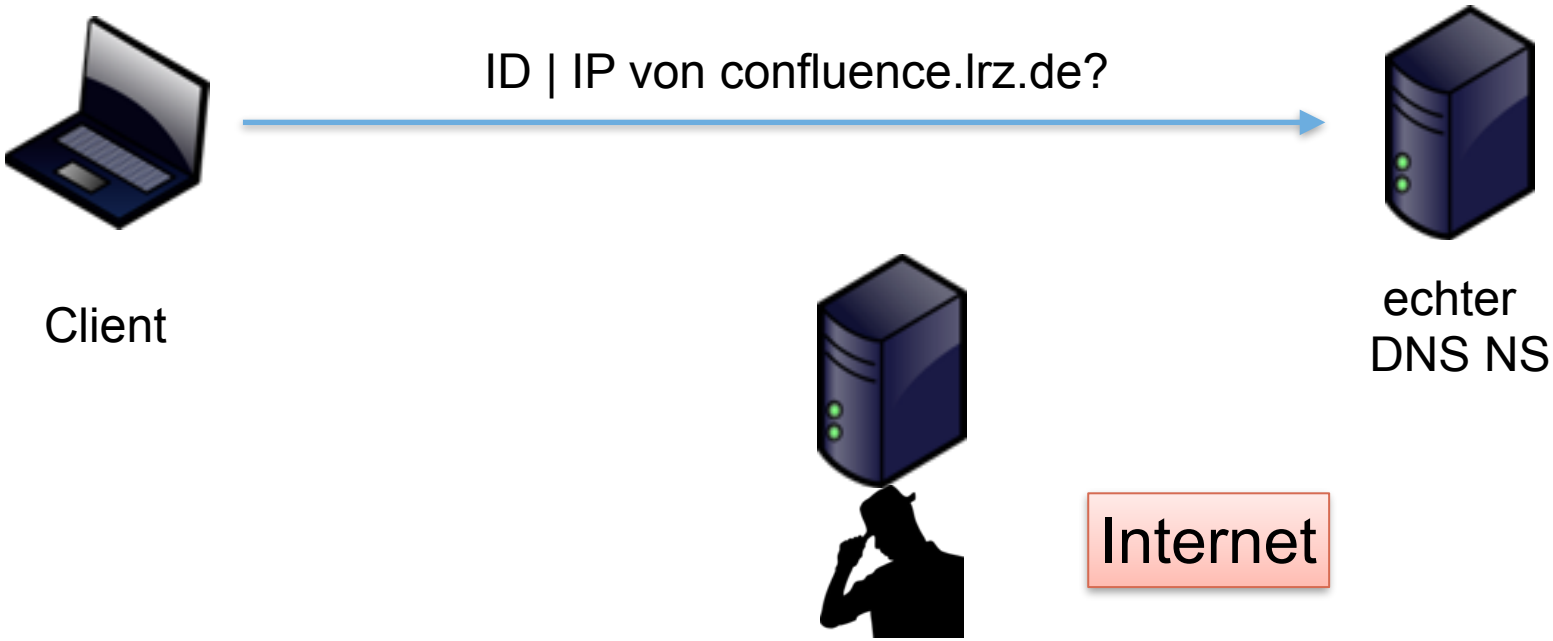
echter
DNS NS

Internet

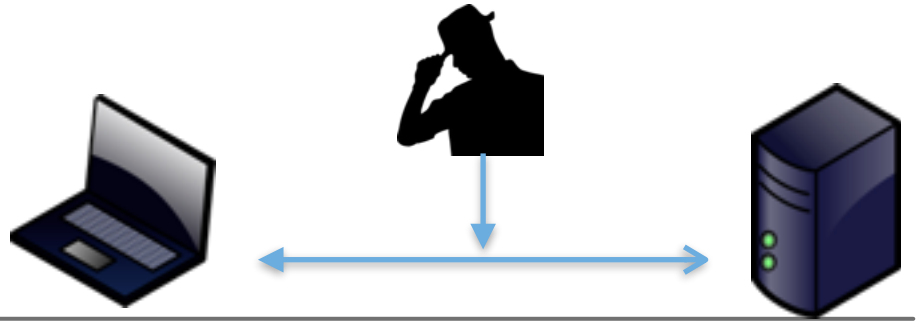
Man-in-middle attack



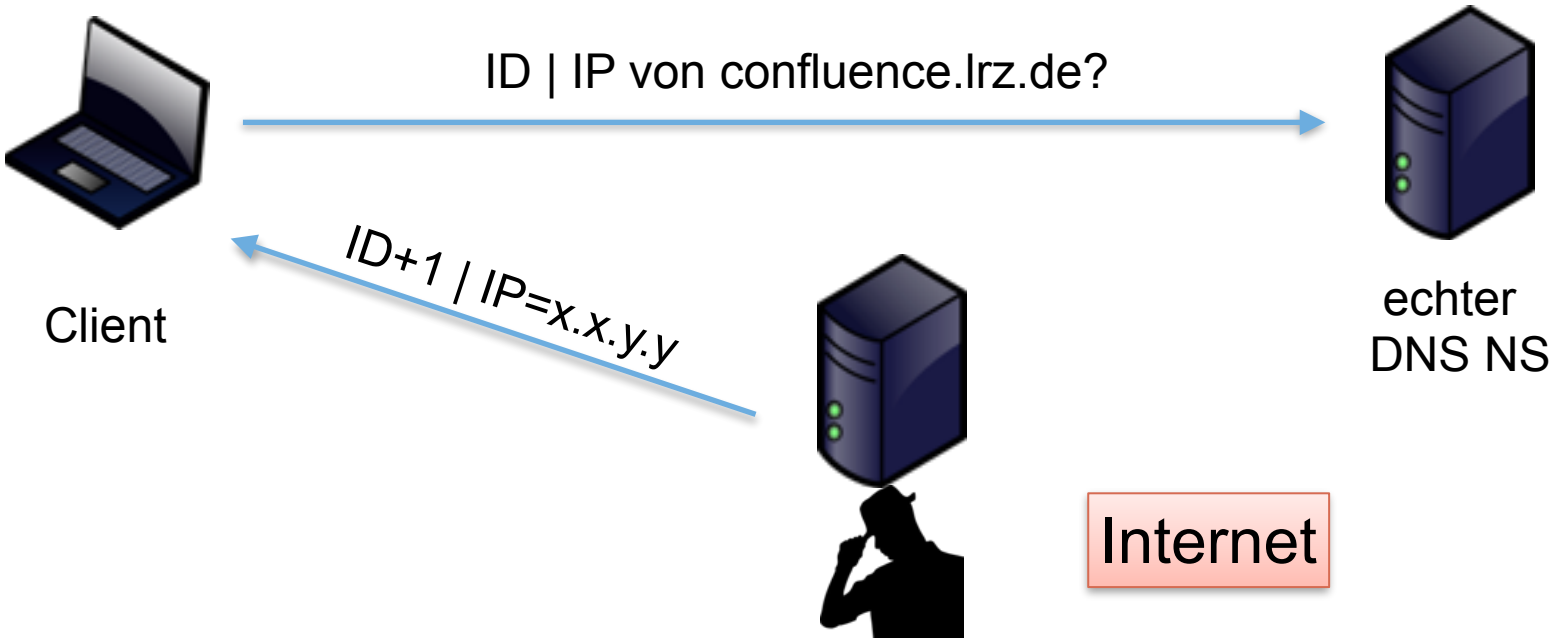
- Pakete mit zufälligen (allen) möglichen IDs in Paketen
- ~N x 100 Antworten senden, um die ID mit hoher Wahrscheinlichkeit zu treffen
- Einige Nameserver erhöhen einfach die ID um 1 von aufeinander folgenden Antworten auf Anfragen



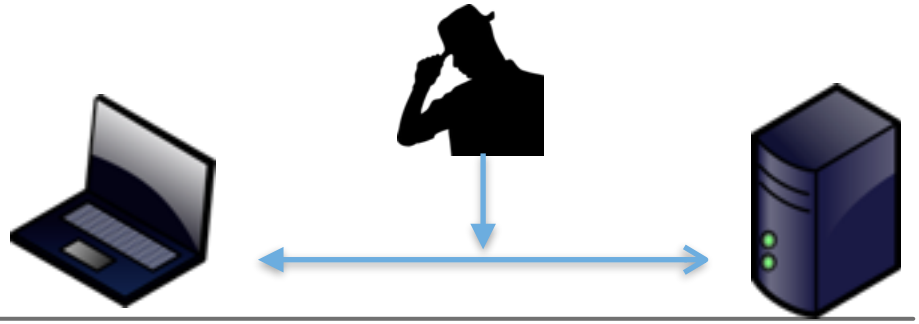
Man-in-middle attack



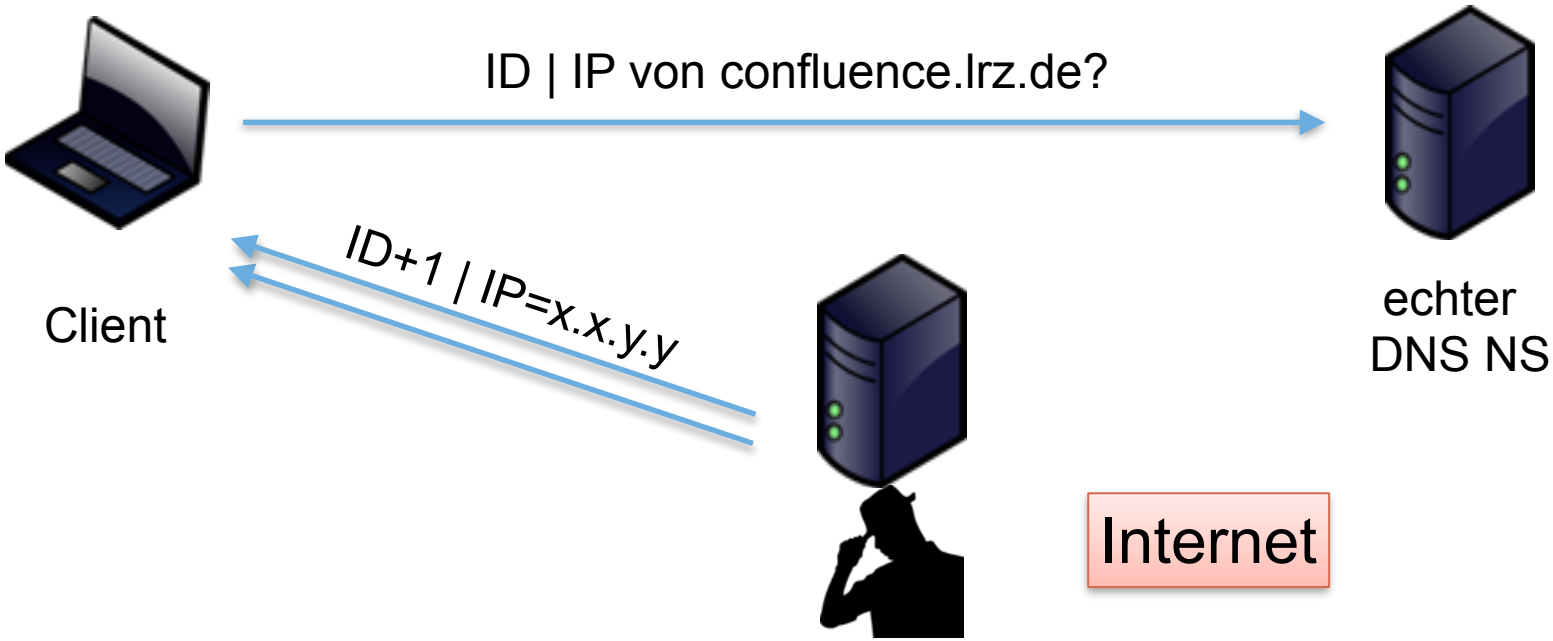
- Pakete mit zufälligen (allen) möglichen IDs in Paketen
- ~N x 100 Antworten senden, um die ID mit hoher Wahrscheinlichkeit zu treffen
- Einige Nameserver erhöhen einfach die ID um 1 von aufeinander folgenden Antworten auf Anfragen



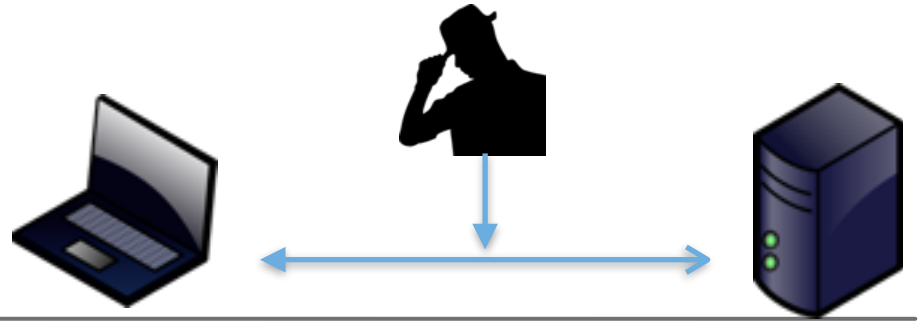
Man-in-middle attack



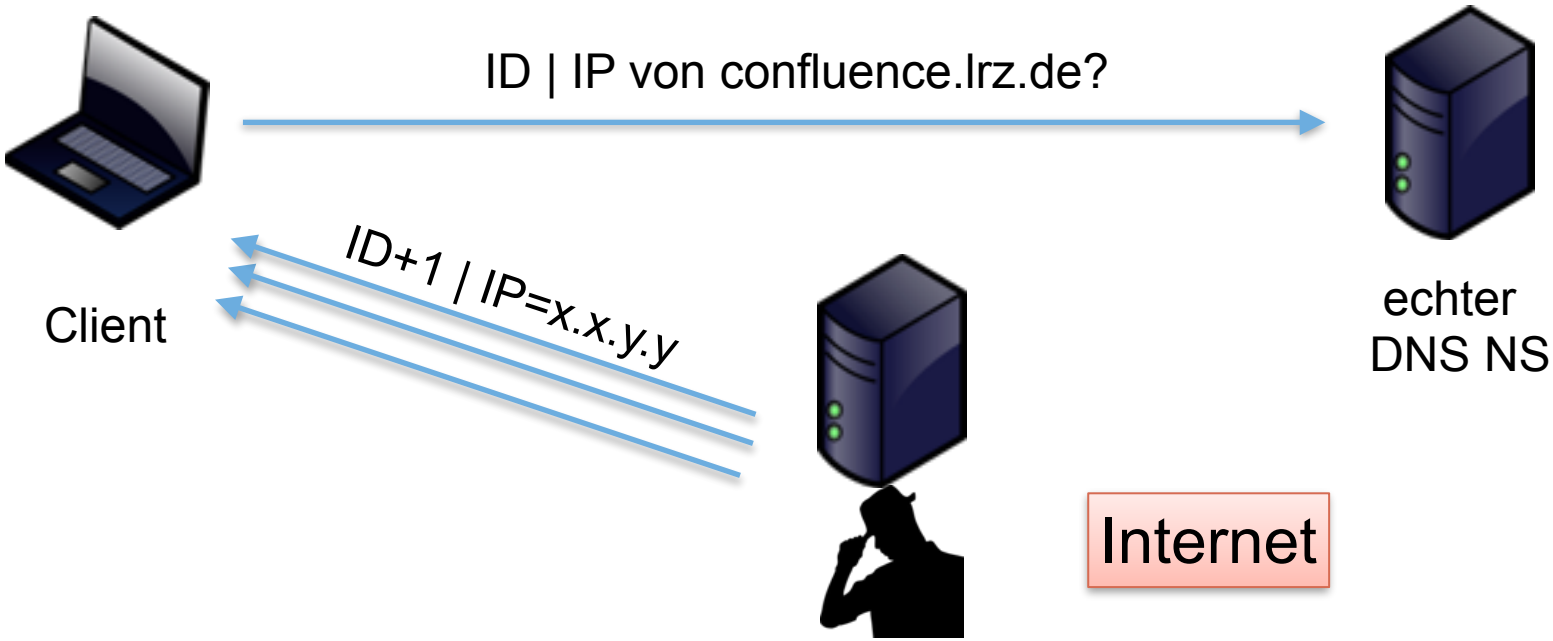
- Pakete mit zufälligen (allen) möglichen IDs in Paketen
- ~N x 100 Antworten senden, um die ID mit hoher Wahrscheinlichkeit zu treffen
- Einige Nameserver erhöhen einfach die ID um 1 von aufeinander folgenden Antworten auf Anfragen



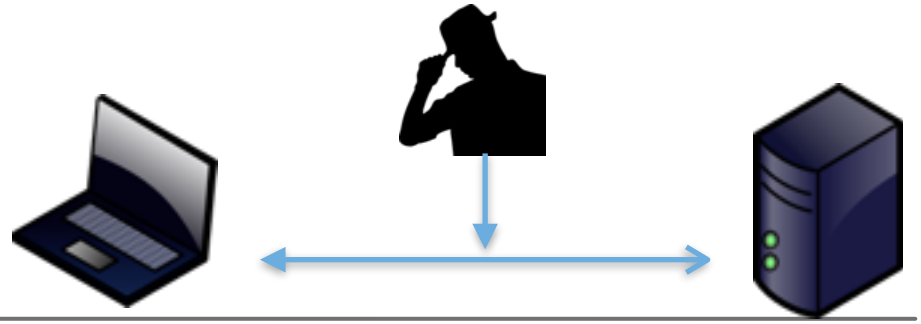
Man-in-middle attack



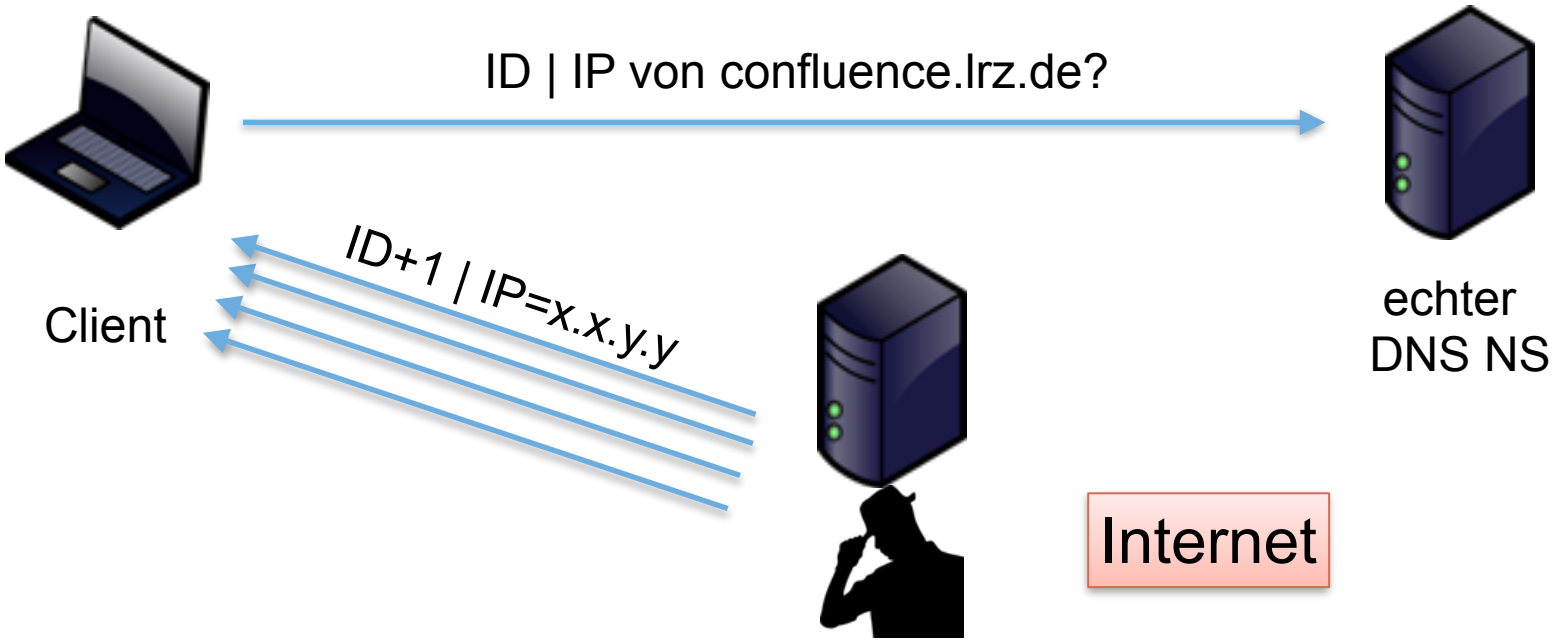
- Pakete mit zufälligen (allen) möglichen IDs in Paketen
- ~N x 100 Antworten senden, um die ID mit hoher Wahrscheinlichkeit zu treffen
- Einige Nameserver erhöhen einfach die ID um 1 von aufeinander folgenden Antworten auf Anfragen



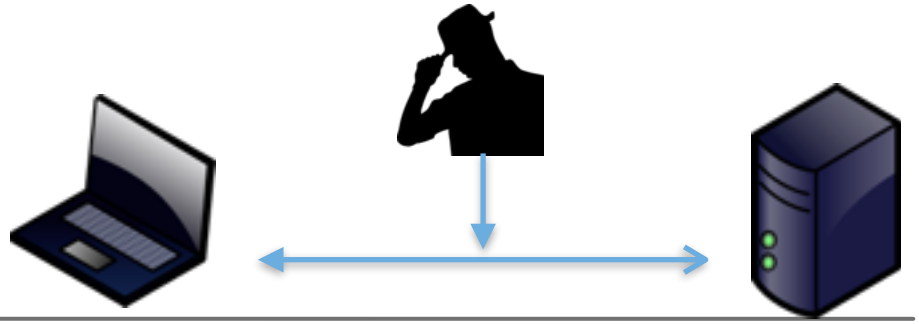
Man-in-middle attack



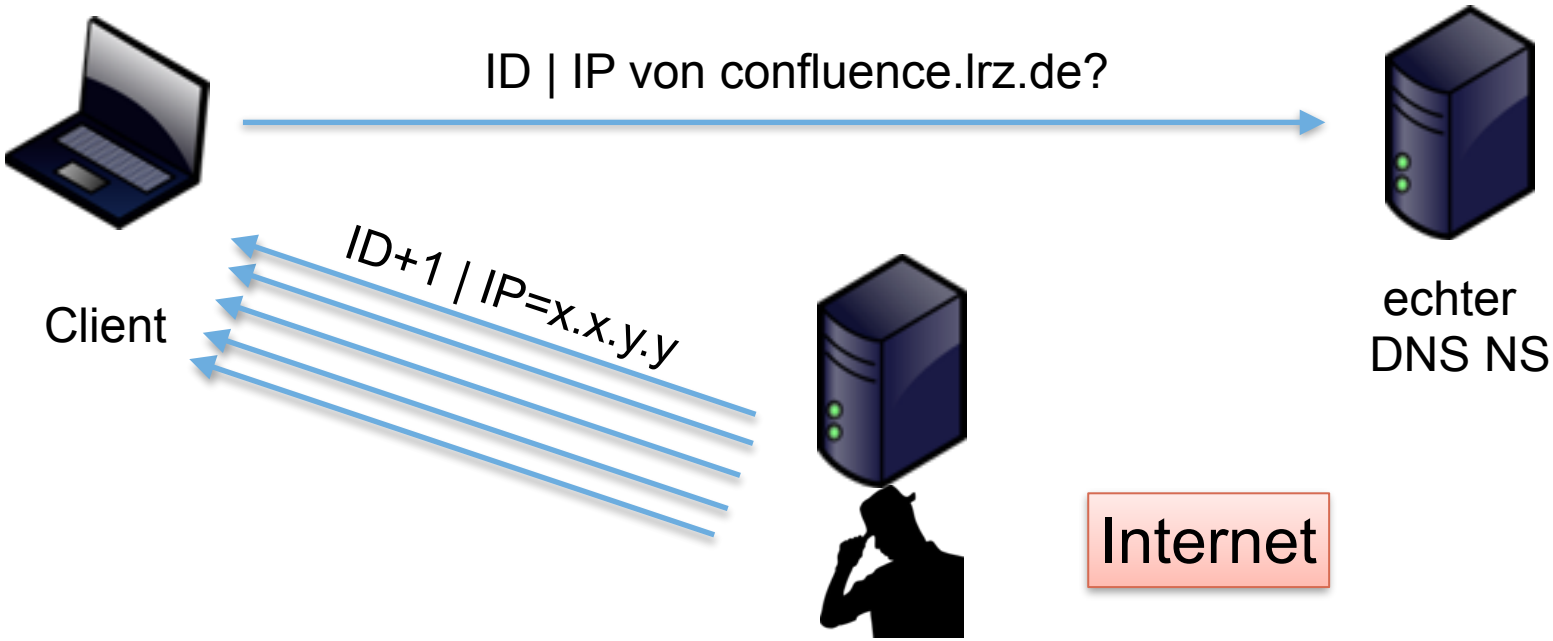
- Pakete mit zufälligen (allen) möglichen IDs in Paketen
- ~N x 100 Antworten senden, um die ID mit hoher Wahrscheinlichkeit zu treffen
- Einige Nameserver erhöhen einfach die ID um 1 von aufeinander folgenden Antworten auf Anfragen



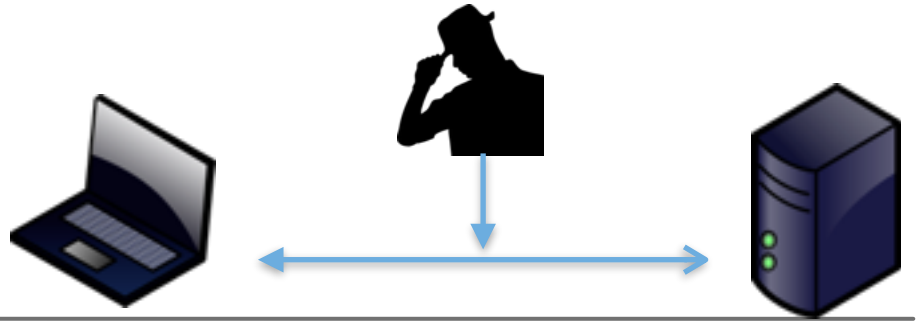
Man-in-middle attack



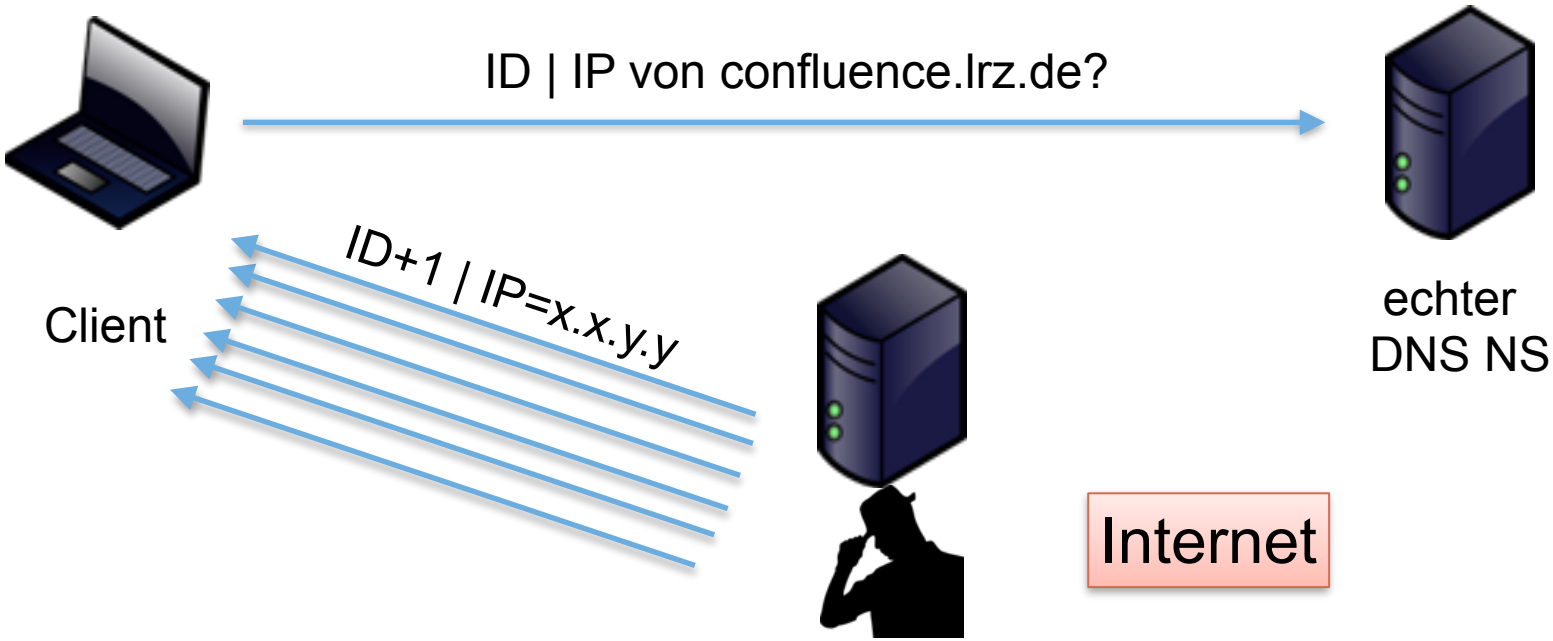
- Pakete mit zufälligen (allen) möglichen IDs in Paketen
- ~N x 100 Antworten senden, um die ID mit hoher Wahrscheinlichkeit zu treffen
- Einige Nameserver erhöhen einfach die ID um 1 von aufeinander folgenden Antworten auf Anfragen



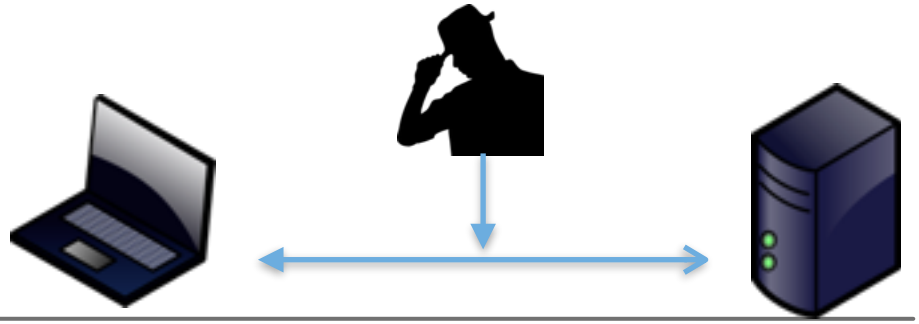
Man-in-middle attack



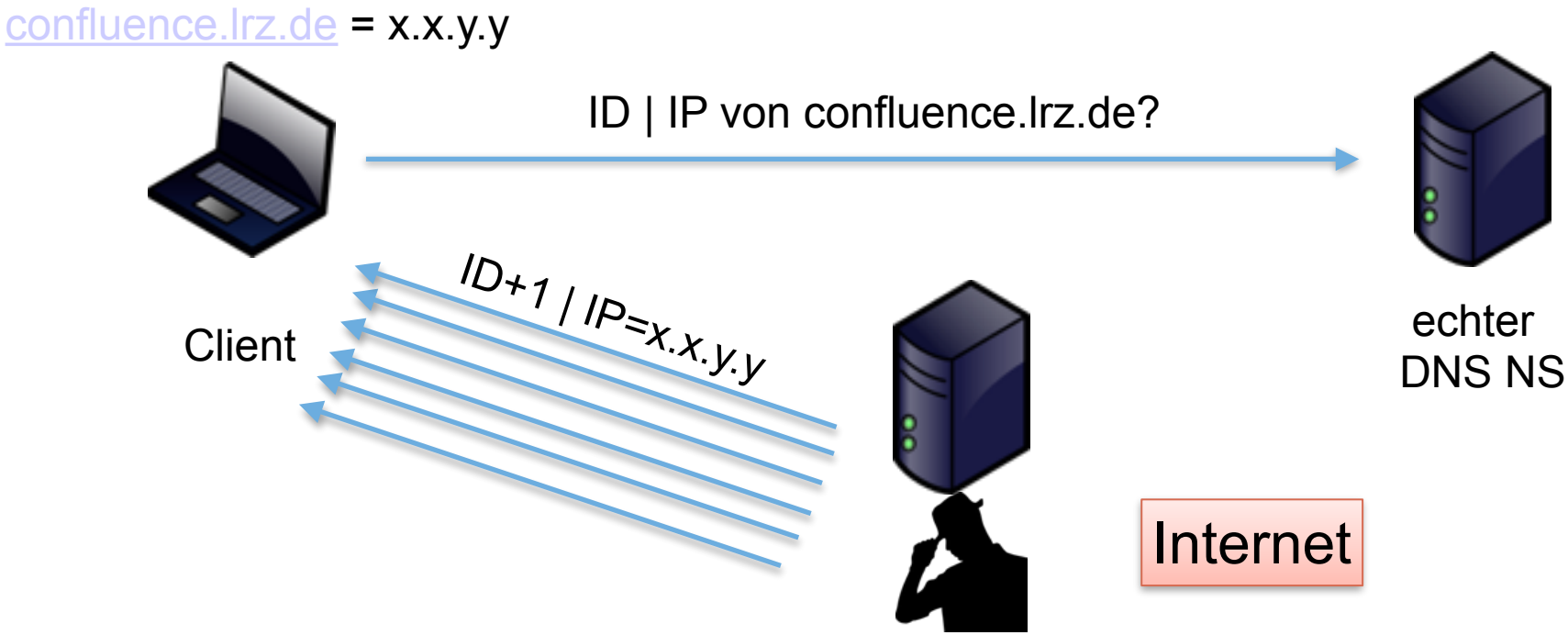
- Pakete mit zufälligen (allen) möglichen IDs in Paketen
- ~N x 100 Antworten senden, um die ID mit hoher Wahrscheinlichkeit zu treffen
- Einige Nameserver erhöhen einfach die ID um 1 von aufeinander folgenden Antworten auf Anfragen



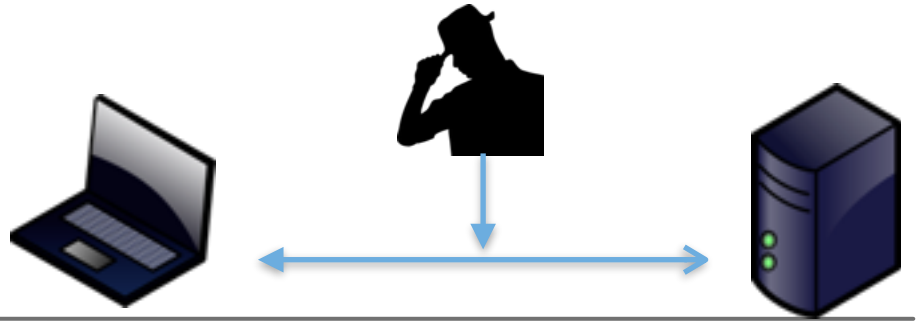
Man-in-middle attack



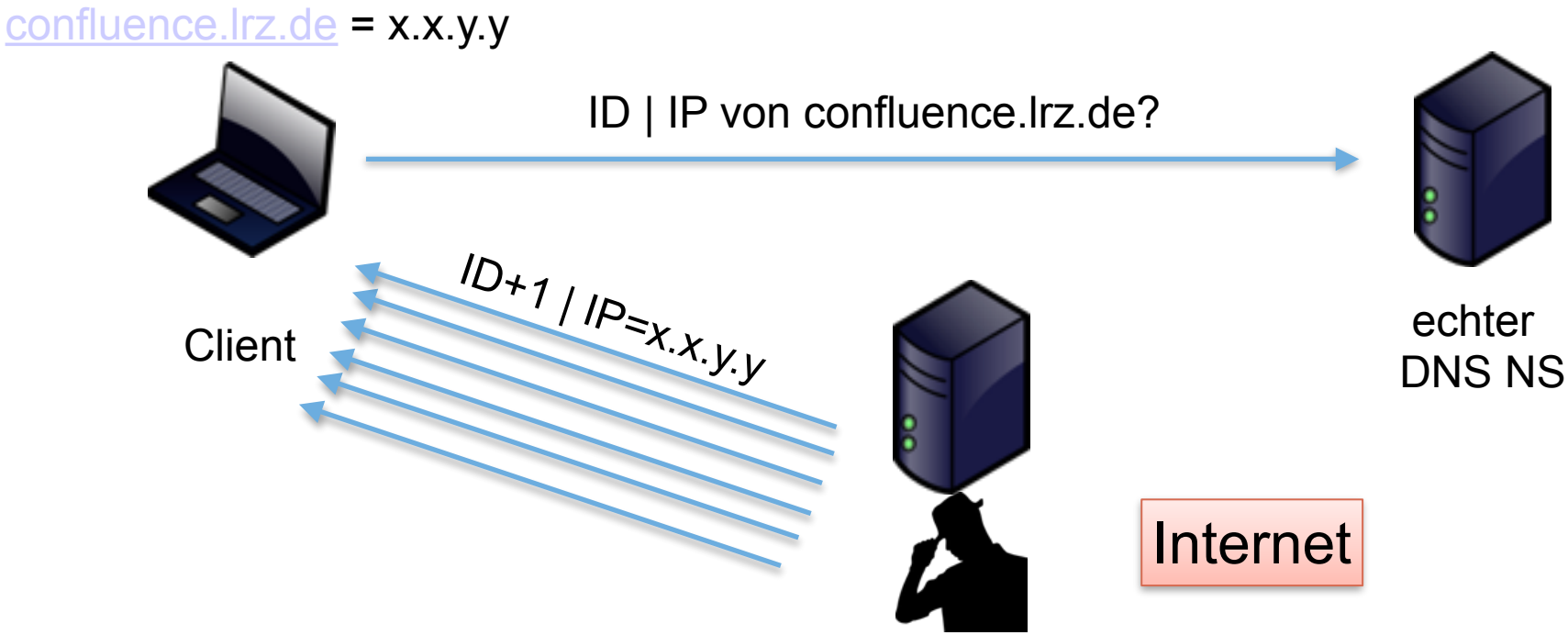
- Pakete mit zufälligen (allen) möglichen IDs in Paketen
- ~N x 100 Antworten senden, um die ID mit hoher Wahrscheinlichkeit zu treffen
- Einige Nameserver erhöhen einfach die ID um 1 von aufeinander folgenden Antworten auf Anfragen



Man-in-middle attack



- Pakete mit zufälligen (allen) möglichen IDs in Paketen
- ~N x 100 Antworten senden, um die ID mit hoher Wahrscheinlichkeit zu treffen
- Einige Nameserver erhöhen einfach die ID um 1 von aufeinander folgenden Antworten auf Anfragen



DNS Caching

- DNS server caches IP-Domänenname-Zuordnung zur Optimierung zukünftiger Anfragen
- Angriffspunkt durch Cache-Poisoning
  - RR-Eintrag
  - DNS ID-Vorhersage
- DNS Cache enthält IP-Verweis auf den Rechner des Angreifers

DNS cache poisoning

confluence.lrz.de. IN A 129.187.4.40

Wenn es der Angreifer durch Spoofing (MTM) schafft, unautorisierte Antworten in den Cache eines Resolvers zu bringen, spricht man von “Cache poisoning”.

DNS cache poisoning

confluence.lrz.de. IN A 129.187.4.40

Wenn es der Angreifer durch Spoofing (MTM) schafft, unautorisierte Antworten in den Cache eines Resolvers zu bringen, spricht man von “Cache poisoning”.

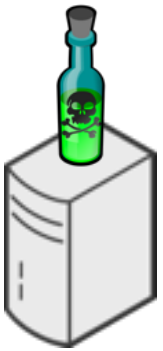
DNS cache poisoning

confluence.lrz.de. IN A 129.187.4.40

Wenn es der Angreifer durch Spoofing (MTM) schafft, unautorisierte Antworten in den Cache eines Resolvers zu bringen, spricht man von “Cache poisoning”.

DNS cache poisoning

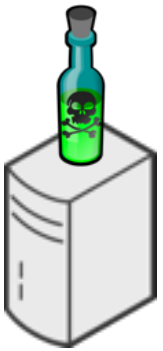
confluence.lrz.de. IN A x.y.w.z



Wenn es der Angreifer durch Spoofing (MTM) schafft, unautorisierte Antworten in den Cache eines Resolvers zu bringen, spricht man von “Cache poisoning”.

DNS cache poisoning

confluence.lrz.de. IN A x.y.w.z

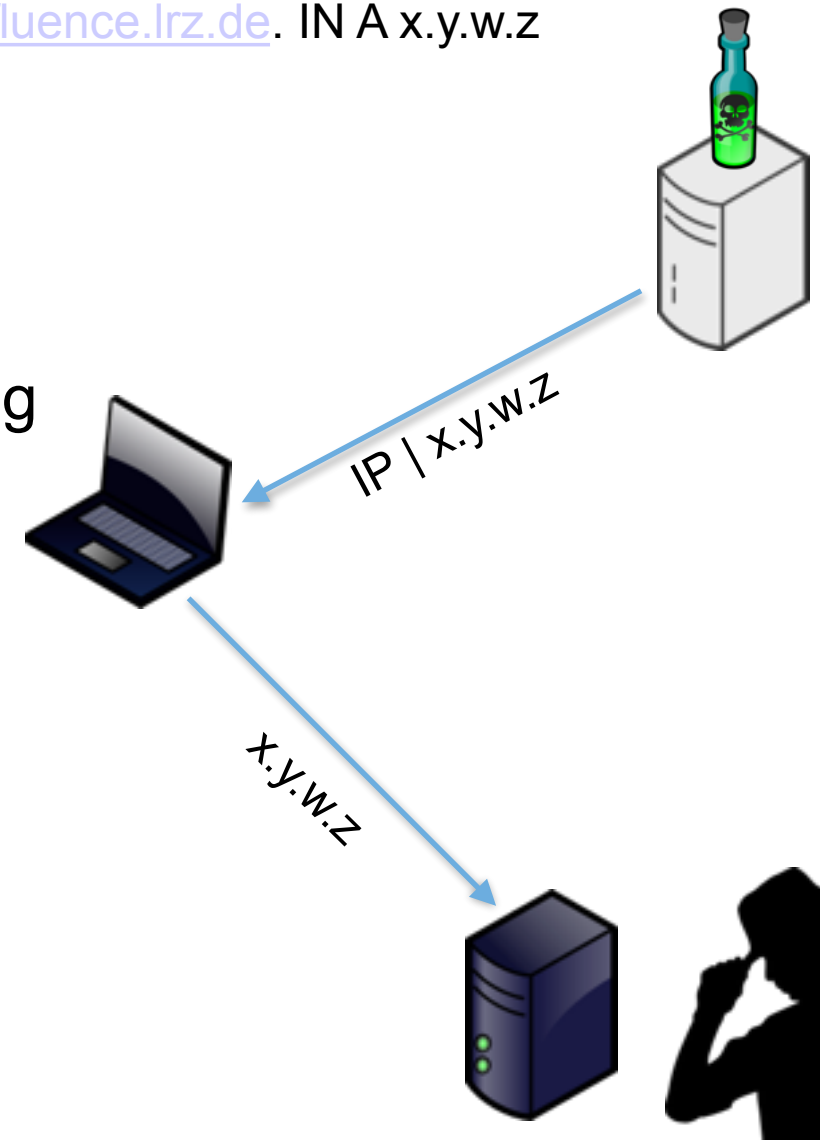


confluence.lrz.de.?

Wenn es der Angreifer durch Spoofing (MTM) schafft, unautorisierte Antworten in den Cache eines Resolvers zu bringen, spricht man von “Cache poisoning”.

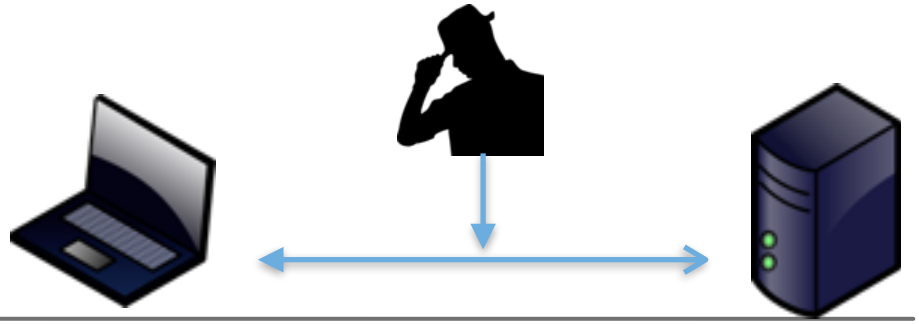
DNS cache poisoning

confluence.lrz.de. IN A x.y.w.z

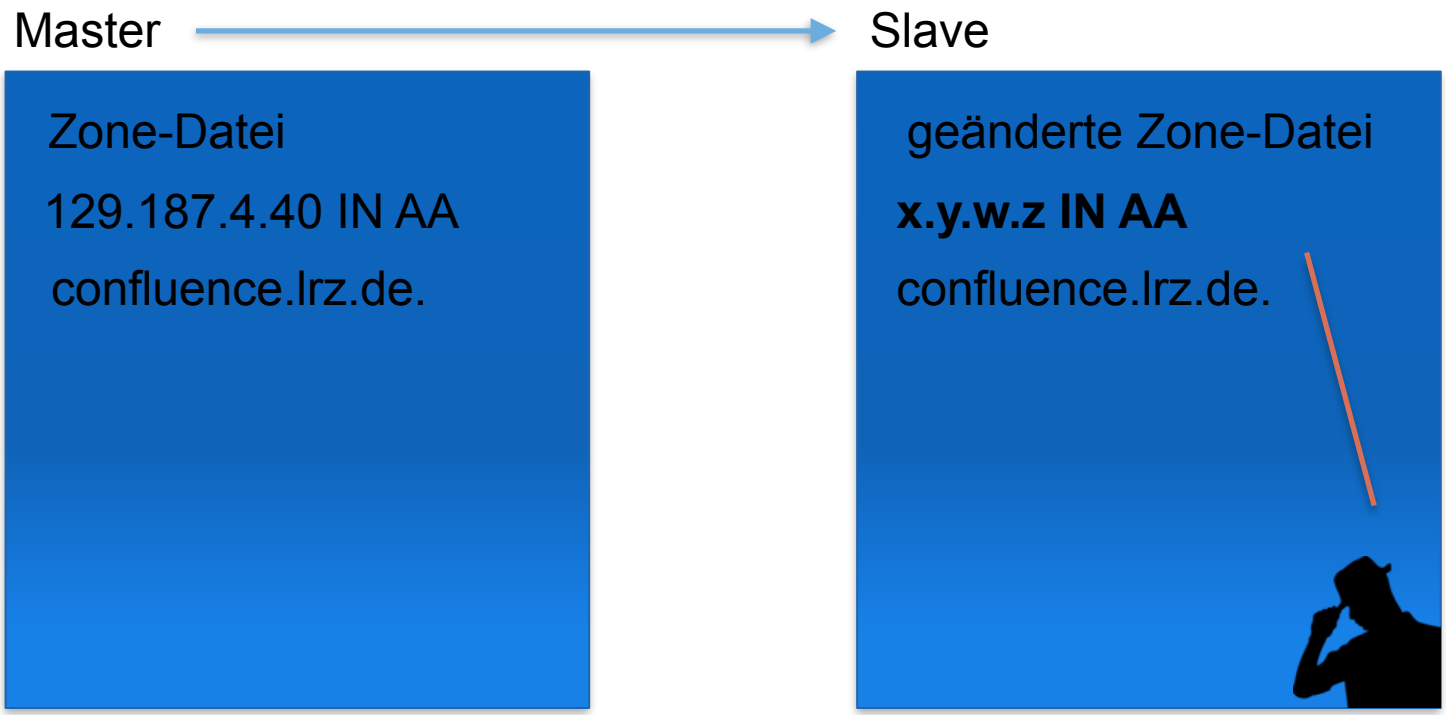


Wenn es der Angreifer durch Spoofing (MTM) schafft, unautorisierte Antworten in den Cache eines Resolvers zu bringen, spricht man von “Cache poisoning”.

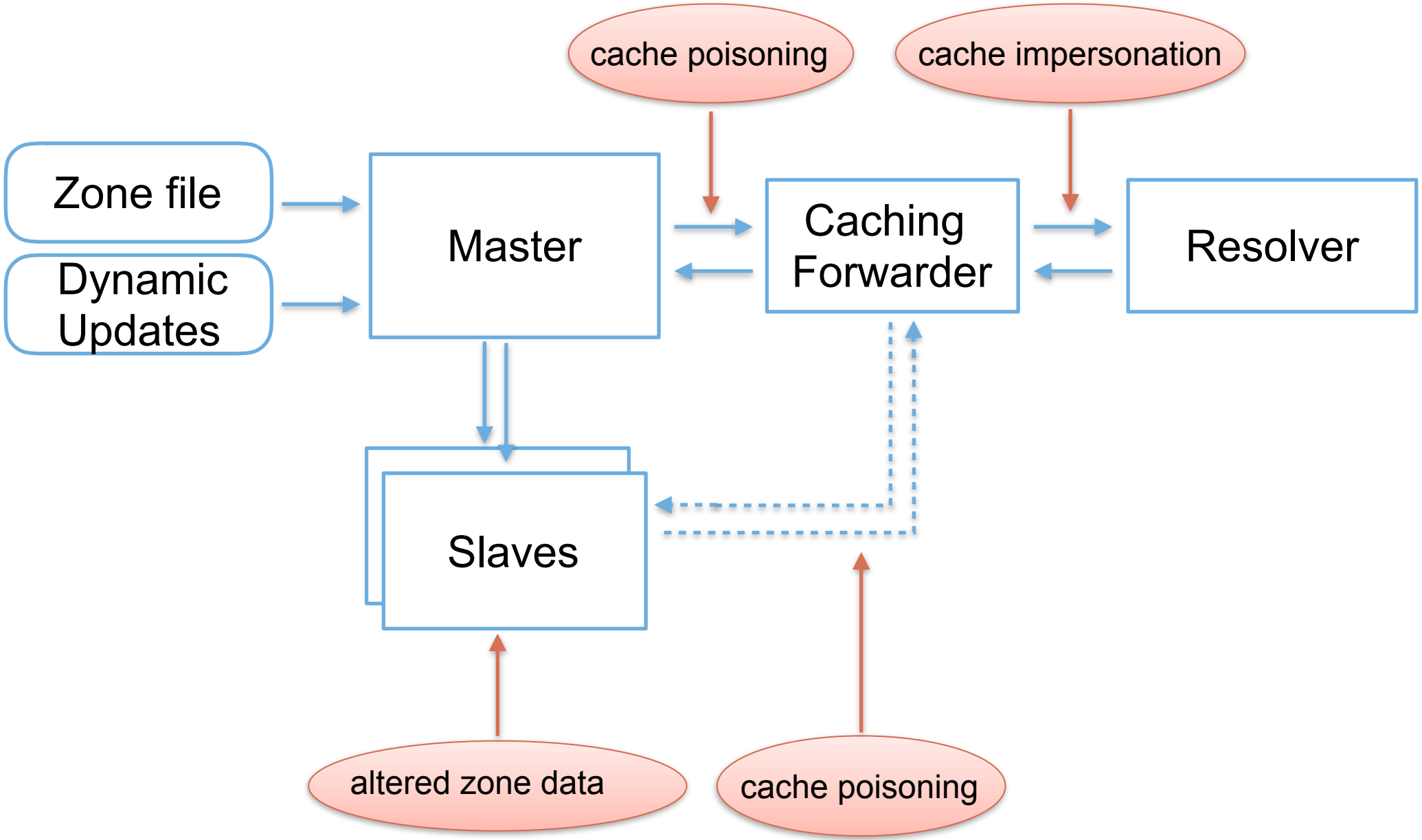
Altered Zone data



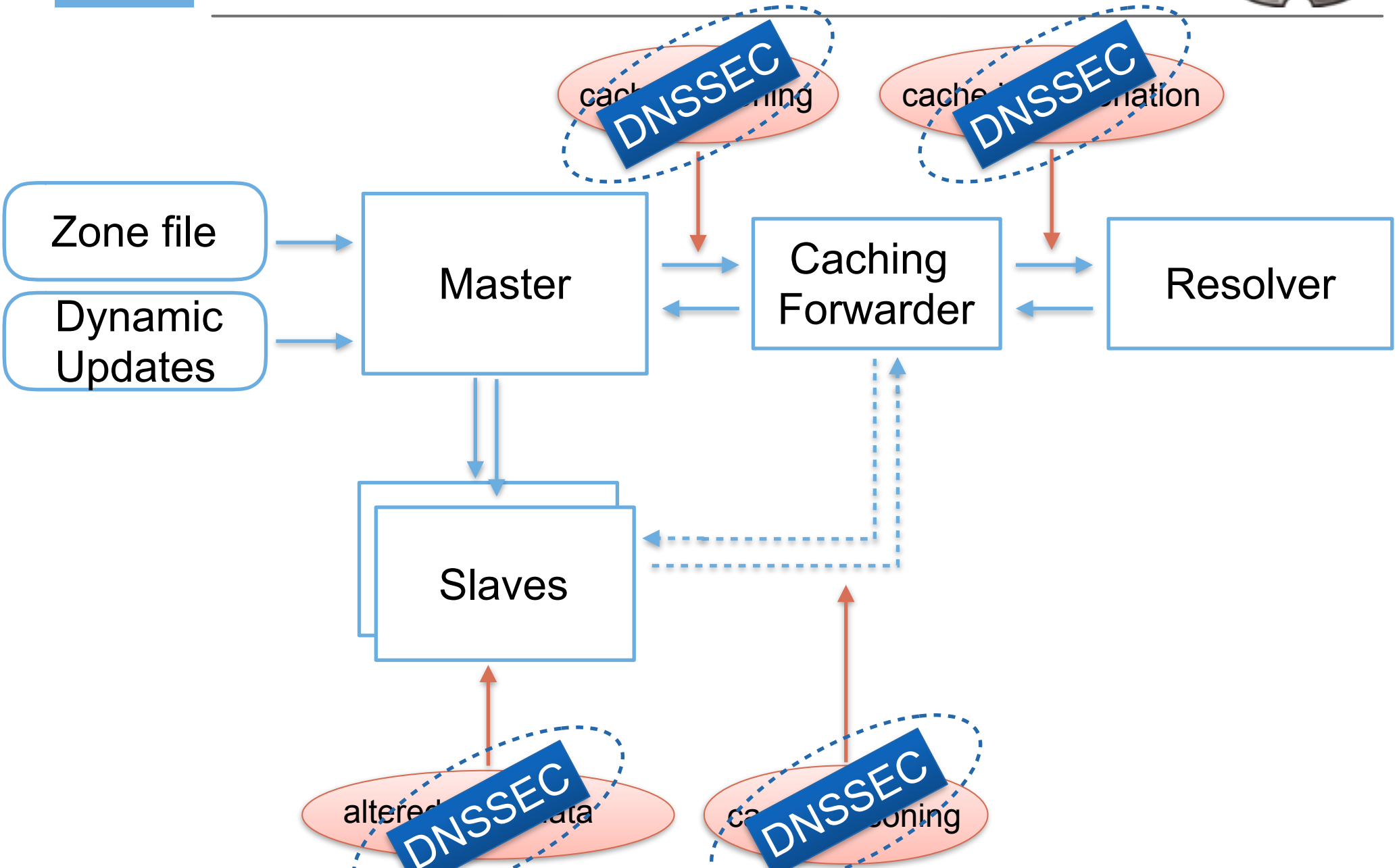
- Zone-Datei auf DNS Slave wird vom Angreifer geändert
- Response leitet auf einen Server des Angreifers um



DNSSEC schützt vor diesen Attacken



DNSSEC schützt vor diesen Attacken



Public Key Infrastructure (PKI) Grundlagen

Public Key Infrastructure - Hash erzeugen

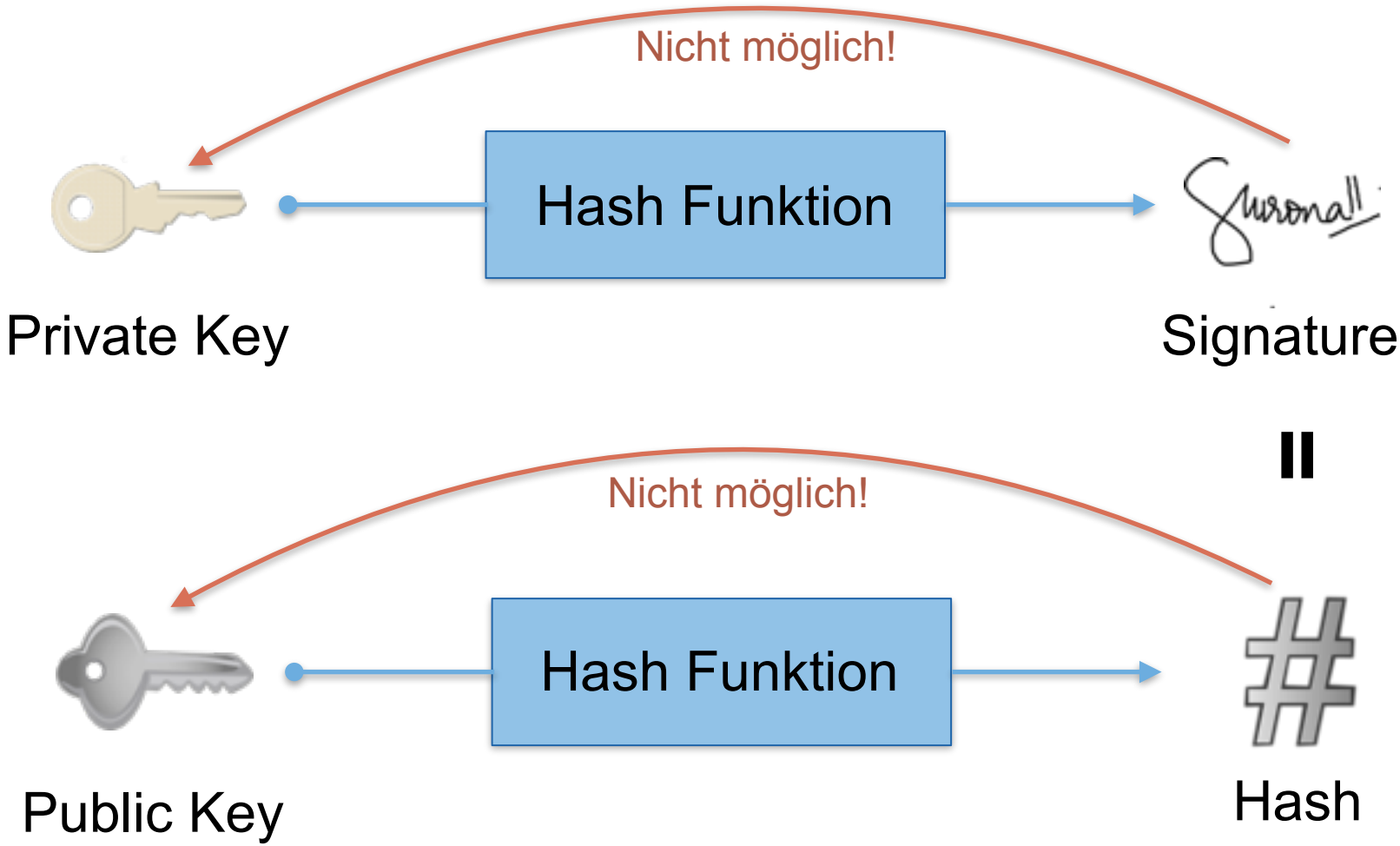
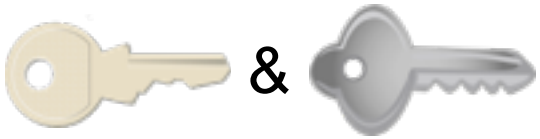


Bild: RIPE NCC

DNSSEC - Funktionsweise

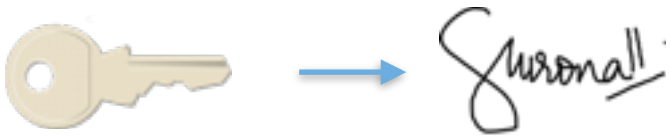
Authoritative Nameserver

- PKI basiert: **privater** und **öffentlicher** Schlüssel



- Antworten sind NICHT verschlüsselt

- Antworten mit **geheimen** Schlüssel signiert



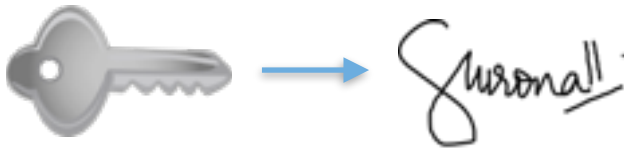
- **Signatur** wird mit DNS Antwort übertragen

IP +

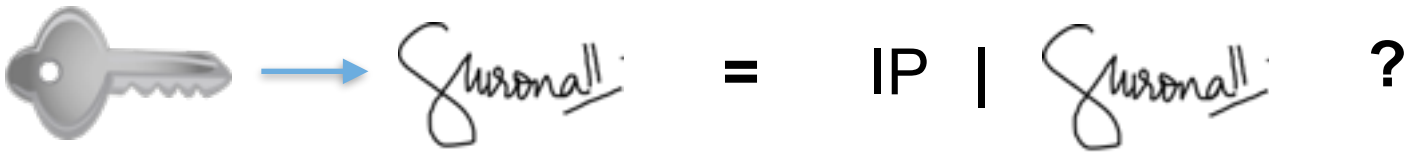
DNSSEC - Funktionsweise

Resolving Nameserver

- Resolving NS überprüfen Signatur anhand des **Hashes** (errechnet aus **öffentlichem** Schlüssel)



- Errechneter Hash = Signatur der DNS Antwort?



Ja ! → DNS Antwort authentisch von diesem DNS-Server

DNSSEC-Abfrage im Detail

resolving
nameserver

authoritative
nameserver

Zone lrz.de.

129.187.4.40

DNSSEC-Abfrage im Detail

public key

1. author. NS erzeugt Keys

resolving nameserver

authoritative nameserver

Zone lrz.de.

129.187.4.40

private key

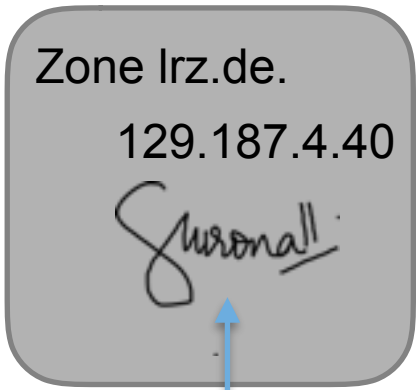
DNSSEC-Abfrage im Detail

1. author. NS erzeugt Keys
2. author. NS signiert Zone

resolving nameserver

public key

authoritative nameserver



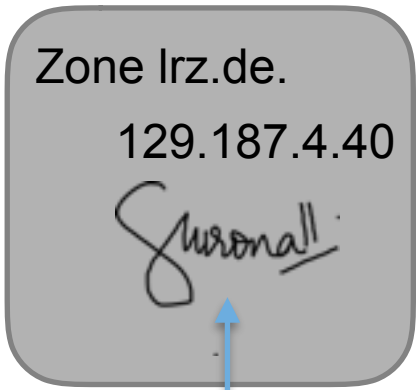
private key

DNSSEC-Abfrage im Detail

1. author. NS erzeugt Keys
2. author. NS signiert Zone
3. resolv. NS empfängt PK

resolving nameserver

authoritative nameserver



public key

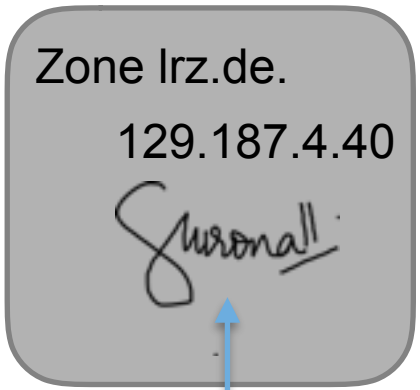
private key

DNSSEC-Abfrage im Detail

1. author. NS erzeugt Keys
2. author. NS signiert Zone
3. resolv. NS empfängt PK
4. confluence.lrz.de?

resolving nameserver

authoritative nameserver



public key

private key

DNSSEC-Abfrage im Detail

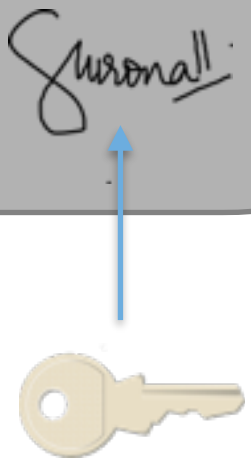
1. author. NS erzeugt Keys
2. author. NS signiert Zone
3. resolv. NS empfängt PK
4. confluence.lrz.de?
5. resolv. NS empfängt DNSSEC Paket

resolving nameserver

authoritative nameserver

Zone lrz.de.

129.187.4.40



private key

public key

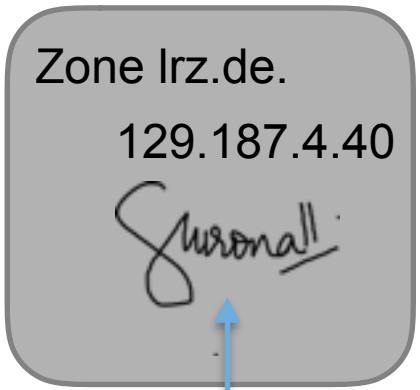
IP

DNSSEC-Abfrage im Detail

1. author. NS erzeugt Keys
2. author. NS signiert Zone
3. resolv. NS empfängt PK
4. confluence.lrz.de?
5. resolv. NS empfängt DNSSEC Paket
6. resolv. NS errechnet Hash aus public key

resolving nameserver

authoritative nameserver



public key

private key

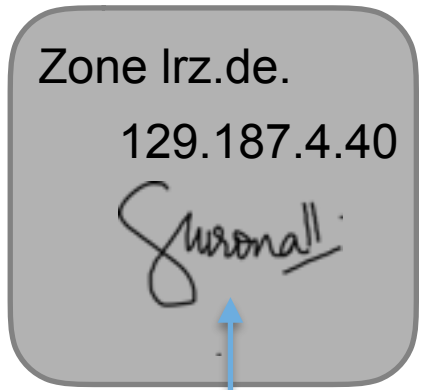
IP

DNSSEC-Abfrage im Detail

1. author. NS erzeugt Keys
2. author. NS signiert Zone
3. resolv. NS empfängt PK
4. confluence.lrz.de?
5. resolv. NS empfängt DNSSEC Paket
6. resolv. NS errechnet Hash aus public key
7. Hash = RRSIG?

resolving nameserver

authoritative nameserver



public key

IP

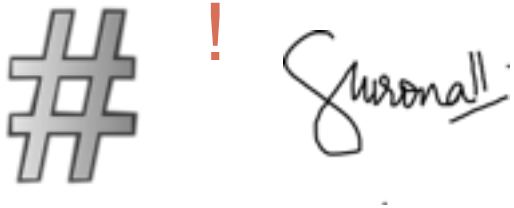
private key

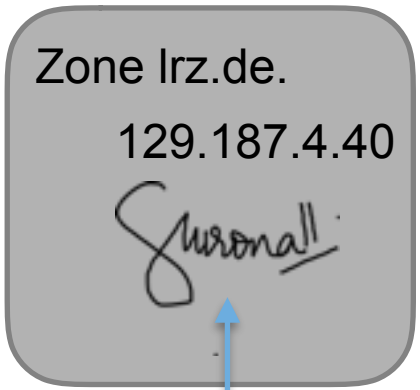
DNSSEC-Abfrage im Detail

1. author. NS erzeugt Keys
2. author. NS signiert Zone
3. resolv. NS empfängt PK
4. confluence.lrz.de?
5. resolv. NS empfängt DNSSEC Paket
6. resolv. NS errechnet Hash aus public key
7. Hash = RRSIG?
8. IP von authoritative NS ist authentisch!

resolving nameserver

authoritative nameserver





public key

private key

IP

authentisch

DNSSEC Records und Zusammenhänge

Neue Resource Records für DNSSEC

- DNSSKEY - DNSSEC Schlüssel
- RRSIG - Signatur über RR
- DS - Delegated Signer
- NSEC - Next Secure, nächster sicherer Eintrag
- NSEC3 - Next Secure rehashed

DNSKEY

- DNS Operator erstellt Public/Private-Schlüsselpaar
- Mögliche Algorithmen:
  - RSAMD5
  - DH
  - RSASHA1
  - DSA-NSEC3-SHA1
  - RSASHA1-NSEC3-SHA1
  - RSASHA256
  - RSASHA512
  - ECC-GOST
  - ECDSAP256SHA256
  - ECDSAP384SHA384

(Quelle: http://www.iana.org/assignments/dns-sec-alg-numbers/dns-sec-alg-numbers.xml)

- Übliche Wahl: RSHASHA256 mit >= 2048 Bit Länge
- Privater Schlüssel wird sicher verwahrt, öffentlicher Schlüssel auf Zonenebene in DNSKEY-Record veröffentlicht
- Verwendung eines Schlüsselpaars in mehreren Zonen möglich, aber nicht empfohlen

DNSKEY

```
<zone>  DNSKEY  <flag> 3 <algo> <pubkey|BASE64>
```

Flag = 256 (Zone Key Bit) oder 257 (Zone Key Bit + SEP)

Algo = Algorithmus

Beispiel:

```
dnssec.bayern.  21597   IN      DNSKEY  256 3 8
        AwEAAcmJ2TpL3K6oW12WAnXBMR/cvbU3CZEKihIOOx0xcZXbTPjqUsOMhBBhHyh
        […]
        RcYNsaZlMJsBhsV6H4czfVuuvgcSPKSE=
```

- Key-ID = 16bit-weise Addition des PubKey ohne Übertrag
  - Optimierung, muss nicht eindeutig sein
  - wird beim Schlüsselhandling oft als Identifier benutzt
  - Anzeige z.B. mit ldns-read-zone von ldnsutils

DNSKEY

- Schlüssel haben **keine** weiteren Informationen
  - keine Gültigkeitsdauer (!)
  - keine Informationen über Besitzer, Erstellungsdatum etc
  - keine Signatur

- Eine Zone kann mehrere DNSKEY-Einträge haben

RRSIG

- RRSIG Records enthalten die Signatur über ein RRSET
- RRSET
  - alle Einträge zu einem bestimmten Namen (Label) mit einem bestimmten RRTYPE
- Es gibt pro aktivem Schlüssel einen RRSIG-Record
- Signatur – und damit die Korrektheit des Resource Records - kann mit Hilfe des öffentlichen Schlüssels (DNSKEY) in der gleichen Zone validiert werden

RRSIG

```
<label> RRSIG <type> <algo> <nlabels> <ttl>
        <expiration> <inception>
        <keytag> <signer> <signature>
```

Beispiel:

```
dnssec.bayern.          21599   IN      SOA     dns1.lrz.de.
        hostmaster.lrz.de. 2016110430 10800 3600 1814400 60
dnssec.bayern.          21599   IN      RRSIG   SOA 8 2 86400
        20170126202339 20161103192339 36675 dnssec.bayern.
        IKF0WM9iN5OnoOKT+4TnzyjQ/DuE8q1b6nzYiZfElS2udT3ZSHJgI9LN
        […]
```

RRSIG

Beispiel 2 (RRSET):

```
test.ws01.ws.dnssec.bayern. 300 IN      A       1.2.3.4
test.ws01.ws.dnssec.bayern. 300 IN      A       1.2.3.5

test.ws01.ws.dnssec.bayern. 300 IN      RRSIG   A 8 5 300
         20161206204740 20161106194740 8088 ws01.ws.dnssec.bayern.
```

Beispiel 3 (Wildcard):

```
*.wildcard.ws01.ws.dnssec.bayern. 300 IN A      8.8.8.8

*.wildcard.ws01.ws.dnssec.bayern. 300 IN RRSIG  A 8 5 300
         20161206204758 20161106194758 8088 ws01.ws.dnssec.bayern.

bla.wildcard.ws01.ws.dnssec.bayern. 300 IN A    8.8.8.8
bla.wildcard.ws01.ws.dnssec.bayern. 300 IN RRSIG A 8 5 300
         20161206204758 20161106194758 8088 ws01.ws.dnssec.bayern.
```

RRSIG

Beispiel 4 (mehrere DNSKEY):

```
ws01.ws.dnssec.bayern. 300  MX     100 dnssec-ws01.ws01.ws.dnssec.bayern.
ws01.ws.dnssec.bayern. 300  RRSIG  MX 8 4 300
         20161206195642 20161106192642 8088 ws01.ws.dnssec.bayern.
ws01.ws.dnssec.bayern. 300  RRSIG  MX 8 4 300
         20161206204353 20161106195621 23526 ws01.ws.dnssec.bayern.
```

DS

- Schlüssel wird vom Zoneninhaber generiert, veröffentlicht und für Signaturen verwendet
  - Signaturen mit öffentlichen Informationen verifizierbar
  - Sicherheitsniveau wie selbstsigniertes Zertifikat

- Jeder vertrauenswürdige Schlüssel müsste in jedem Resolver als trusted-key hinterlegt werden

DS

Delegiertes Vertrauen

- Alle Resolver vertrauen einem bekannten Schlüssel der Root-Zone
- Root-Zone delegiert Vertrauen für Unterzonen (TLDs) an bestimmte Schlüssel
- Unterzonen delegieren Vertrauen für SLDs an bestimmte Schlüssel

DS

- Delegation Signer Record enthält den SHA-1 und/oder SHA-256 Hash des DNSKEY der delegierten Zone
- Bilden von Schlüssel-Ketten möglich
- Beispiel einer Delegation:

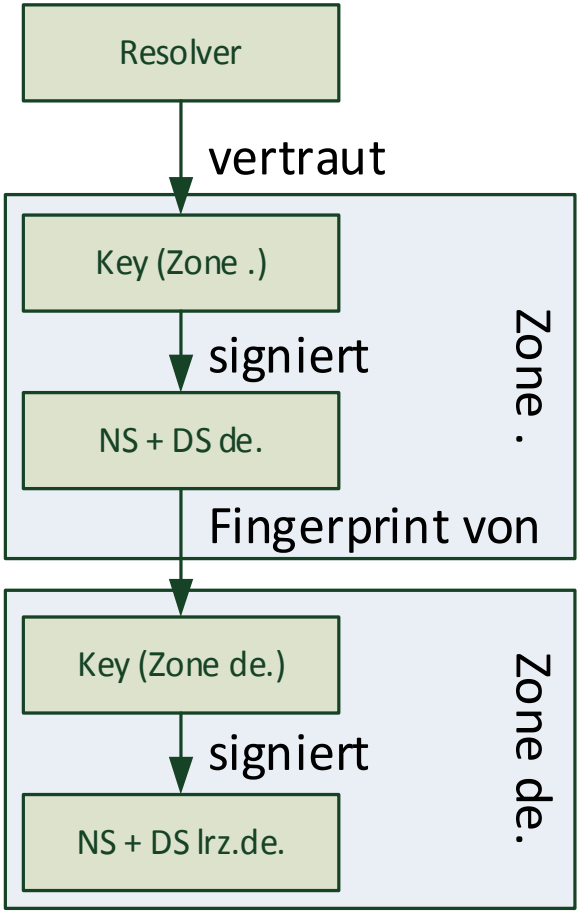
```
;; AUTHORITY SECTION:
lrz.de.                 86400   IN      NS      dns1.lrz.de.
lrz.de.                 86400   IN      NS      dns2.lrz.bayern.
lrz.de.                 86400   IN      NS      dns3.lrz.eu.
lrz.de.                 86400   IN      DS      42814 8 2
        C2DB28705793D9DB9DC5210F70A7DBC84
        6A2B056C6A3C358C2CB284E AA5B0F6A
lrz.de.                 86400   IN      RRSIG   DS 8 2 86400
        20141116150000 20141109150000 56395 de.
```

- Authoritativ für den DS-Record ist die Parentzone!

DS

- Update des DS-Keys in der Parentzone passiert üblicherweise Out-of-Band
  - Robot-Schnittstelle des Registrars und der Registry
  - Mail an Poldi
- In-Band Schnittstelle (CDS+CDNSKEY, RFC 7344) hat sich bisher nicht durchgesetzt

KSK und ZSK


Resolver
vertraut
Key (Zone .)
signiert
NS + DS de.
Zone .
Fingerprint von
Key (Zone de.)
signiert
NS + DS lrz.de.
Zone de.

KSK und ZSK

- Schlüsselpaar wird bei jeder Änderung, aber auch im automatischen Zugriff (Resigning) benötigt
- Erhöhtes Risiko des Diebstahls → schneller Schlüsselwechsel muss möglich sein

  vs.

  Schlüssel kann nicht ausgetauscht werden, ohne die DS-Records in der Parentzone anzupassen
- Split in zwei (oder mehr) Schlüsselpaare
  - Key-Signing-Key (im DS-Record, wird nur benutzt um den/die gerade aktiven DNSKEYs zu signieren),
  - Zone-Signing-Key (signiert den Inhalt der Zone)

KSK und ZSK

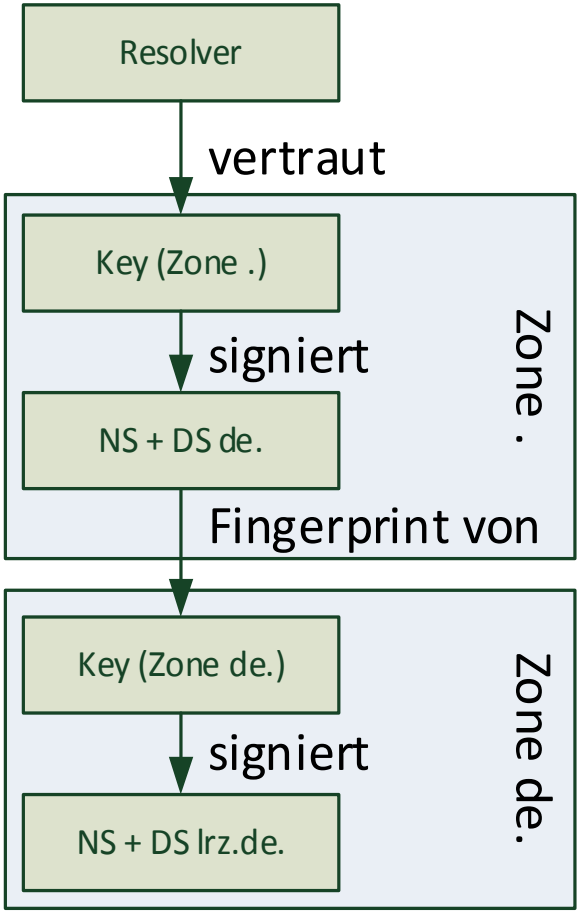
- KSK könnte sicher verschlossen werden, ZSK ist im täglichen Zugriff und kann ohne Registry rotiert werden
  - Modell Rootzone
  - Wird nicht von allen Tools unterstützt
- Oft empfohlen: kleinerer ZSK (1024 vs. 2048 Bit)
  - Dafür häufiger Wechsel des ZSK
  - Begründung: Rechenzeit / Speicherplatz

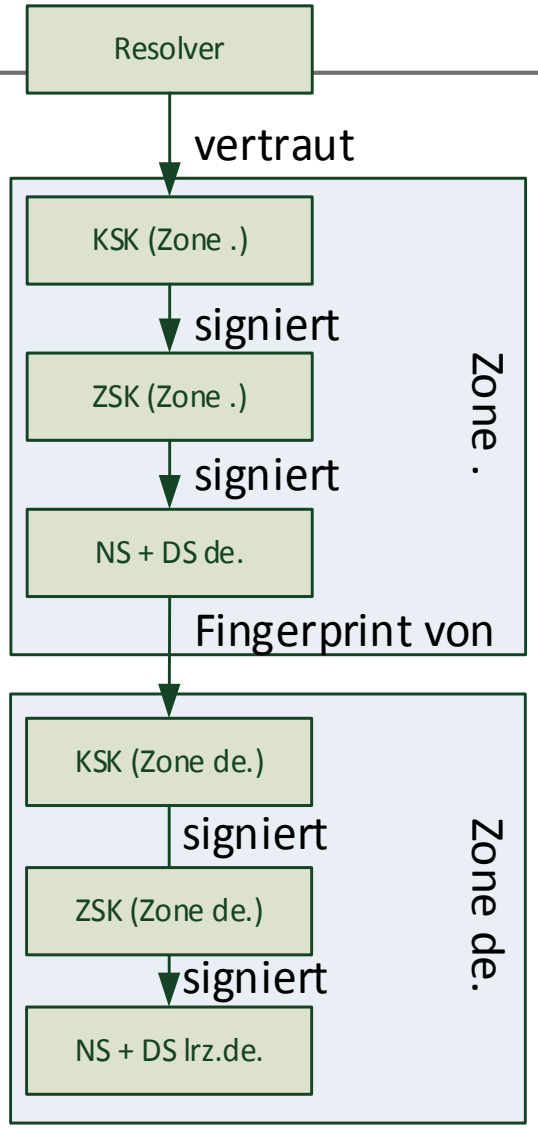
KSK und ZSK

Meine Meinung:

- Rechenzeit / Speicherplatz kein so großes Problem
  - Größte Zone am LRZ 100k Einträge mit 2k Schlüssel
- Wenn es darauf wirklich ankommt eher anderen Algorithmus
  - z.B. Algo 13 (ECDSAP256SHA256) bei Cloudflare
- Keys haben keine vordefinierte Lebenszeit, ein Rollen nach Kalender ist unnötig
  - Rollover sollte aber geübt werden, um im Bedardsfall nicht ins Schwimmen zu geraten (Key Compromise)
- KSK/ZSK-Trennung trotzdem Best-Practise

KSK und ZSK


Resolver
vertraut
Key (Zone .)
signiert
NS + DS de.
Zone .
Fingerprint von
Key (Zone de.)
signiert
NS + DS lrz.de.
Zone de.


Resolver
vertraut
KSK (Zone .)
signiert
ZSK (Zone .)
signiert
NS + DS de.
Zone .
Fingerprint von
KSK (Zone de.)
signiert
ZSK (Zone de.)
signiert
NS + DS lrz.de.
Zone de.

KSK und ZSK, Beispiel

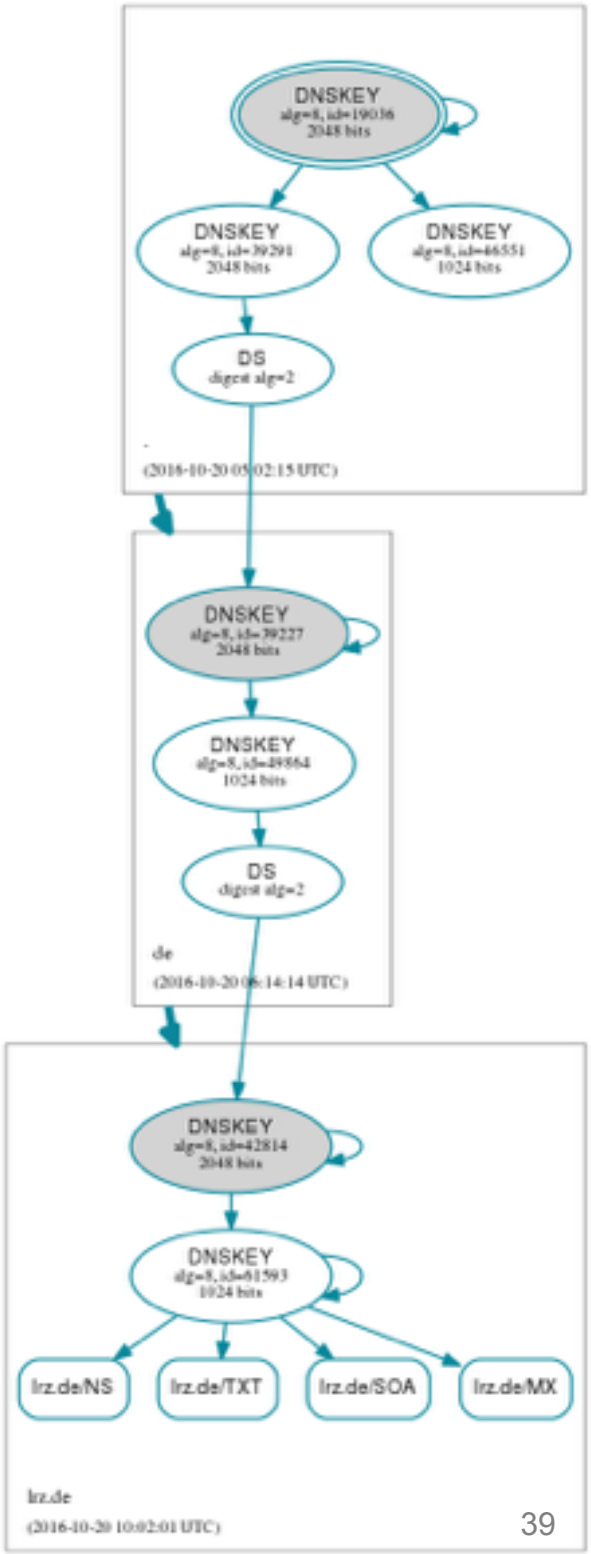
```
mhn.de              DS        26325 10 2 […]

mhn.de.             DNSKEY    257 3 10 […] ;{id = 26325}
mhn.de.             DNSKEY    256 3 10 […] ;{id = 1916}
mhn.de.             RRSIG     DNSKEY 10 2 86400 20151109000000
                              20110622150447 26325 mhn.de. […]
mhn.de.             RRSIG     DNSKEY 10 2 86400 20151109000000
                              20110622150449 1916 mhn.de. […]

mhn.de.             SOA       dns1.lrz.de. hostmaster.lrz.de.
                              2012022410 21600 […]
mhn.de.             RRSIG     SOA 10 2 86400 20151109000000
                              20120224090634 1916 mhn.de. […]
```

DNSSEC-Test mit DNSViz & Debugger Tools

DNSViz Tool - dnsviz.net

- Web tool zur Überprüfung von DNSSEC
- DNSKEYs und RRsets
- Durchlauf der „chain of trust“
- Delegated Signer graphische Darstellung

Demonstration in der folgenden Übung



DNSSEC Debugger

- Detaillierte Information welcher Teil fehlschlägt:

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

- Erlaubt Fehleranalyse für pro Zone in „chain-of-trust“

  DNSKey (DNS Zone signing key)

  DS (Delegated Signer)

  RRSIG (Resource Record Signature)

Beispiel:

Analyzing DNSSEC problems for mpe.mpg.de

.	✓ Found 2 DNSKEY records for . ✓ DS=19036/SHA-1 verifies DNSKEY=19036/SEP ✓ Found 1 RRSIGs over DNSKEY RRset ✓ RRSIG=19036 and DNSKEY=19036/SEP verifies the DNSKEY RRset
de	✓ Found 1 DS records for de in the . zone ✓ Found 1 RRSIGs over DS RRset ✓ RRSIG=39291 and DNSKEY=39291 verifies the DS RRset ✓ Found 3 DNSKEY records for de ✓ DS=39227/SHA-256 verifies DNSKEY=39227/SEP ✓ Found 1 RRSIGs over DNSKEY RRset ✓ RRSIG=39227 and DNSKEY=39227/SEP verifies the DNSKEY RRset
mpg.de	✓ Found 2 DS records for mpg.de in the de zone ✓ Found 1 RRSIGs over DS RRset ✓ RRSIG=44919 and DNSKEY=44919 verifies the DS RRset ✓ Found 2 DNSKEY records for mpg.de ✓ DS=40326/SHA-256 verifies DNSKEY=40326/SEP ✓ Found 2 RRSIGs over DNSKEY RRset ✓ RRSIG=13895 and DNSKEY=13895 verifies the DNSKEY RRset
mpe.mpg.de	✗ No DS records found for mpe.mpg.de in the mpg.de zone ✗ No DNSKEY records found ✓ mpe.mpg.de A RR has value 134.76.31.205 ✗ No RRSIGs found

Beispiel:

.root DNSSEC verifiziert

Analyzing DNSSEC problems for mpe.mpg.de

.	✓ Found 2 DNSKEY records for . ✓ DS=19036/SHA-1 verifies DNSKEY=19036/SEP ✓ Found 1 RRSIGs over DNSKEY RRset ✓ RRSIG=19036 and DNSKEY=19036/SEP verifies the DNSKEY RRset
de	✓ Found 1 DS records for de in the . zone ✓ Found 1 RRSIGs over DS RRset ✓ RRSIG=39291 and DNSKEY=39291 verifies the DS RRset ✓ Found 3 DNSKEY records for de ✓ DS=39227/SHA-256 verifies DNSKEY=39227/SEP ✓ Found 1 RRSIGs over DNSKEY RRset ✓ RRSIG=39227 and DNSKEY=39227/SEP verifies the DNSKEY RRset
mpg.de	✓ Found 2 DS records for mpg.de in the de zone ✓ Found 1 RRSIGs over DS RRset ✓ RRSIG=44919 and DNSKEY=44919 verifies the DS RRset ✓ Found 2 DNSKEY records for mpg.de ✓ DS=40326/SHA-256 verifies DNSKEY=40326/SEP ✓ Found 2 RRSIGs over DNSKEY RRset ✓ RRSIG=13895 and DNSKEY=13895 verifies the DNSKEY RRset
mpe.mpg.de	✗ No DS records found for mpe.mpg.de in the mpg.de zone ✗ No DNSKEY records found ✓ mpe.mpg.de A RR has value 134.76.31.205 ✗ No RRSIGs found

Beispiel:

.root DNSSEC verifiziert

Analyzing DNSSEC problems for mpe.mpg.de

.	✓ Found 2 DNSKEY records for . ✓ DS=19036/SHA-1 verifies DNSKEY=19036/SEP ✓ Found 1 RRSIGs over DNSKEY RRset ✓ RRSIG=19036 and DNSKEY=19036/SEP verifies the DNSKEY RRset
de	✓ Found 1 DS records for de in the . zone ✓ Found 1 RRSIGs over DS RRset ✓ RRSIG=39291 and DNSKEY=39291 verifies the DS RRset ✓ Found 3 DNSKEY records for de ✓ DS=39227/SHA-256 verifies DNSKEY=39227/SEP ✓ Found 1 RRSIGs over DNSKEY RRset ✓ RRSIG=39227 and DNSKEY=39227/SEP verifies the DNSKEY RRset
mpg.de	✓ Found 2 DS records for mpg.de in the de zone ✓ Found 1 RRSIGs over DS RRset ✓ RRSIG=44919 and DNSKEY=44919 verifies the DS RRset ✓ Found 2 DNSKEY records for mpg.de ✓ DS=40326/SHA-256 verifies DNSKEY=40326/SEP ✓ Found 2 RRSIGs over DNSKEY RRset ✓ RRSIG=13895 and DNSKEY=13895 verifies the DNSKEY RRset
mpe.mpg.de	✗ No DS records found for mpe.mpg.de in the mpg.de zone ✗ No DNSKEY records found ✓ mpe.mpg.de A RR has value 134.76.31.205 ✗ No RRSIGs found

Beispiel:

.root DNSSEC verifiziert

.de TLD Domäne ist korrekt signiert und authentifiziert. DNSKEY, DS und RRSIG verifiziert:

Analyzing DNSSEC problems for mpe.mpg.de

.	✓ Found 2 DNSKEY records for . ✓ DS=19036/SHA-1 verifies DNSKEY=19036/SEP ✓ Found 1 RRSIGs over DNSKEY RRset ✓ RRSIG=19036 and DNSKEY=19036/SEP verifies the DNSKEY RRset
de	✓ Found 1 DS records for de in the . zone ✓ Found 1 RRSIGs over DS RRset ✓ RRSIG=39291 and DNSKEY=39291 verifies the DS RRset ✓ Found 3 DNSKEY records for de ✓ DS=39227/SHA-256 verifies DNSKEY=39227/SEP ✓ Found 1 RRSIGs over DNSKEY RRset ✓ RRSIG=39227 and DNSKEY=39227/SEP verifies the DNSKEY RRset
mpg.de	✓ Found 2 DS records for mpg.de in the de zone ✓ Found 1 RRSIGs over DS RRset ✓ RRSIG=44919 and DNSKEY=44919 verifies the DS RRset ✓ Found 2 DNSKEY records for mpg.de ✓ DS=40326/SHA-256 verifies DNSKEY=40326/SEP ✓ Found 2 RRSIGs over DNSKEY RRset ✓ RRSIG=13895 and DNSKEY=13895 verifies the DNSKEY RRset
mpe.mpg.de	✗ No DS records found for mpe.mpg.de in the mpg.de zone ✗ No DNSKEY records found ✓ mpe.mpg.de A RR has value 134.76.31.205 ✗ No RRSIGs found

Beispiel:

.root DNSSEC verifiziert

Analyzing DNSSEC problems for mpe.mpg.de

.	✓ Found 2 DNSKEY records for . ✓ DS=19036/SHA-1 verifies DNSKEY=19036/SEP ✓ Found 1 RRSIGs over DNSKEY RRset ✓ RRSIG=19036 and DNSKEY=19036/SEP verifies the DNSKEY RRset
de	✓ Found 1 DS records for de in the . zone ✓ Found 1 RRSIGs over DS RRset ✓ RRSIG=39291 and DNSKEY=39291 verifies the DS RRset ✓ Found 3 DNSKEY records for de ✓ DS=39227/SHA-256 verifies DNSKEY=39227/SEP ✓ Found 1 RRSIGs over DNSKEY RRset ✓ RRSIG=39227 and DNSKEY=39227/SEP verifies the DNSKEY RRset
mpg.de	✓ Found 2 DS records for mpg.de in the de zone ✓ Found 1 RRSIGs over DS RRset ✓ RRSIG=44919 and DNSKEY=44919 verifies the DS RRset ✓ Found 2 DNSKEY records for mpg.de ✓ DS=40326/SHA-256 verifies DNSKEY=40326/SEP ✓ Found 2 RRSIGs over DNSKEY RRset ✓ RRSIG=13895 and DNSKEY=13895 verifies the DNSKEY RRset
mpe.mpg.de	✗ No DS records found for mpe.mpg.de in the mpg.de zone ✗ No DNSKEY records found ✓ mpe.mpg.de A RR has value 134.76.31.205 ✗ No RRSIGs found

Beispiel:

.root DNSSEC verifiziert

.de TLD Domäne ist korrekt signiert und authentifiziert. DNSKEY, DS und RRSIG verifiziert:

Max-Planck Gesellschaft Domäne ist korrekt signiert und authentifiziert.

Analyzing DNSSEC problems for mpe.mpg.de

.	✓ Found 2 DNSKEY records for . ✓ DS=19036/SHA-1 verifies DNSKEY=19036/SEP ✓ Found 1 RRSIGs over DNSKEY RRset ✓ RRSIG=19036 and DNSKEY=19036/SEP verifies the DNSKEY RRset
de	✓ Found 1 DS records for de in the . zone ✓ Found 1 RRSIGs over DS RRset ✓ RRSIG=39291 and DNSKEY=39291 verifies the DS RRset ✓ Found 3 DNSKEY records for de ✓ DS=39227/SHA-256 verifies DNSKEY=39227/SEP ✓ Found 1 RRSIGs over DNSKEY RRset ✓ RRSIG=39227 and DNSKEY=39227/SEP verifies the DNSKEY RRset
mpg.de	✓ Found 2 DS records for mpg.de in the de zone ✓ Found 1 RRSIGs over DS RRset ✓ RRSIG=44919 and DNSKEY=44919 verifies the DS RRset ✓ Found 2 DNSKEY records for mpg.de ✓ DS=40326/SHA-256 verifies DNSKEY=40326/SEP ✓ Found 2 RRSIGs over DNSKEY RRset ✓ RRSIG=13895 and DNSKEY=13895 verifies the DNSKEY RRset
mpe.mpg.de	✗ No DS records found for mpe.mpg.de in the mpg.de zone ✗ No DNSKEY records found ✓ mpe.mpg.de A RR has value 134.76.31.205 ✗ No RRSIGs found

Beispiel:

.root DNSSEC verifiziert

MPE-Domäne (Max-Planck Institut für Extraterrestrische Physik) hat keinen DNSKey oder DS, damit auch keine RRSIGs.

Analyzing DNSSEC problems for mpe.mpg.de

.	✓ Found 2 DNSKEY records for . ✓ DS=19036/SHA-1 verifies DNSKEY=19036/SEP ✓ Found 1 RRSIGs over DNSKEY RRset ✓ RRSIG=19036 and DNSKEY=19036/SEP verifies the DNSKEY RRset
de	✓ Found 1 DS records for de in the . zone ✓ Found 1 RRSIGs over DS RRset ✓ RRSIG=39291 and DNSKEY=39291 verifies the DS RRset ✓ Found 3 DNSKEY records for de ✓ DS=39227/SHA-256 verifies DNSKEY=39227/SEP ✓ Found 1 RRSIGs over DNSKEY RRset ✓ RRSIG=39227 and DNSKEY=39227/SEP verifies the DNSKEY RRset
mpg.de	✓ Found 2 DS records for mpg.de in the de zone ✓ Found 1 RRSIGs over DS RRset ✓ RRSIG=44919 and DNSKEY=44919 verifies the DS RRset ✓ Found 2 DNSKEY records for mpg.de ✓ DS=40326/SHA-256 verifies DNSKEY=40326/SEP ✓ Found 2 RRSIGs over DNSKEY RRset ✓ RRSIG=13895 and DNSKEY=13895 verifies the DNSKEY RRset
mpe.mpg.de	✗ No DS records found for mpe.mpg.de in the mpg.de zone ✗ No DNSKEY records found ✓ mpe.mpg.de A RR has value 134.76.31.205 ✗ No RRSIGs found

Beispiel:

.root DNSSEC verifiziert

.de TLD Domäne ist korrekt signiert und authentifiziert. DNSKEY, DS und RRSIG verifiziert:

Max-Planck Gesellschaft Domäne ist korrekt signiert und authentifiziert.

MPE-Domäne (Max-Planck Institut für Extraterrestrische Physik) hat keinen DNSKey oder DS, damit auch keine RRSIGs.

Analyzing DNSSEC problems for mpe.mpg.de

.	✓ Found 2 DNSKEY records for . ✓ DS=19036/SHA-1 verifies DNSKEY=19036/SEP ✓ Found 1 RRSIGs over DNSKEY RRset ✓ RRSIG=19036 and DNSKEY=19036/SEP verifies the DNSKEY RRset
de	✓ Found 1 DS records for de in the . zone ✓ Found 1 RRSIGs over DS RRset ✓ RRSIG=39291 and DNSKEY=39291 verifies the DS RRset ✓ Found 3 DNSKEY records for de ✓ DS=39227/SHA-256 verifies DNSKEY=39227/SEP ✓ Found 1 RRSIGs over DNSKEY RRset ✓ RRSIG=39227 and DNSKEY=39227/SEP verifies the DNSKEY RRset
mpg.de	✓ Found 2 DS records for mpg.de in the de zone ✓ Found 1 RRSIGs over DS RRset ✓ RRSIG=44919 and DNSKEY=44919 verifies the DS RRset ✓ Found 2 DNSKEY records for mpg.de ✓ DS=40326/SHA-256 verifies DNSKEY=40326/SEP ✓ Found 2 RRSIGs over DNSKEY RRset ✓ RRSIG=13895 and DNSKEY=13895 verifies the DNSKEY RRset
mpe.mpg.de	✗ No DS records found for mpe.mpg.de in the mpg.de zone ✗ No DNSKEY records found ✓ mpe.mpg.de A RR has value 134.76.31.205 ✗ No RRSIGs found

Übung - DNSViz/Debugger DNSSEC-Überprüfung

Testen Sie die folgenden Domains

Mit dnsviz.net und finden Sie mit dnssec-debugger.verisign.com heraus, wo DNSSEC gegebenenfalls fehl schlägt:

- tu-muenchen.de
- physik.uni-muenchen.de
- mpa-garching.mpg.de

DNSSEC am Beispiel von Bind 9.9

DNSSEC mit BIND - Schritt für Schritt

1. Schlüsselpaar erzeugen (ZSK und KSK)
2. Zonen signieren und veröffentlichen
3. DS Record im Parent setzen

Zone Signing - Zone Signing Key

Zone Signing - Zone Signing Key

Zone Signing Key erzeugen

Zone Signing - Zone Signing Key

Zone Signing Key erzeugen

$ dnssec-keygen -a RSASHA256 -b 2048 -K — /var/named/keys -n ZONE example.org

Zone Signing - Zone Signing Key

Zone Signing Key erzeugen

$ dnssec-keygen -a RSASHA256 -b 2048 -K — /var/named/keys -n ZONE example.org

Typ: RSA

Zone Signing - Zone Signing Key

Zone Signing Key erzeugen

```
$ dnssec-keygen -a RSASHA256 -b 2048 -K — /var/named/keys -n ZONE example.org
```

No. Bit

Zone Signing - Zone Signing Key

Zone Signing Key erzeugen

```
$ dnssec-keygen -a RSASHA256 -b 2048 -K — /var/named/keys -n ZONE example.org
```

Verzeichnis für Keys

Zone Signing - Zone Signing Key

Zone Signing Key erzeugen

```
$ dnssec-keygen -a RSASHA256 -b 2048 -K — /var/named/keys -n ZONE example.org
```

Name der Zone

Zone Signing - Zone Signing Key

Zone Signing Key erzeugen

$ dnssec-keygen -a RSASHA256 -b 2048 -K — /var/named/keys -n ZONE example.org

Zone Signing - Key Signing Key

Zone Signing - Key Signing Key

Key Signing Key erzeugen

Zone Signing - Key Signing Key

Key Signing Key erzeugen

$ dnssec-keygen -a RSASHA256 -b 2048 -f KSK -K — /var/named/keys -n ZONE example.org

Zone Signing - Key Signing Key

Key Signing Key erzeugen

```
$ dnssec-keygen -a RSASHA256 -b 2048 -f KSK -K — /var/named/keys -n ZONE example.org
```

Typ: RSA

Zone Signing - Key Signing Key

Key Signing Key erzeugen

```
$ dnssec-keygen -a RSASHA256 -b 2048 -f KSK -K — /var/named/keys -n ZONE example.org
```

No. Bit

Zone Signing - Key Signing Key

Key Signing Key erzeugen

```
$ dnssec-keygen -a RSASHA256 -b 2048 -f KSK -K — /var/named/keys -n ZONE example.org
```

KSK!

Zone Signing - Key Signing Key

Key Signing Key erzeugen

```
$ dnssec-keygen -a RSASHA256 -b 2048 -f KSK -K — /var/named/keys -n ZONE example.org
```

Verzeichnis für Keys

Zone Signing - Key Signing Key

Key Signing Key erzeugen

```
$ dnssec-keygen -a RSASHA256 -b 2048 -f KSK -K — /var/named/keys -n ZONE example.org
```

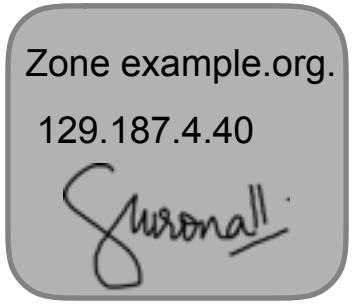
Name der Zone

Zone Signing - Key Signing Key

Key Signing Key erzeugen

$ dnssec-keygen -a RSASHA256 -b 2048 -f KSK -K — /var/named/keys -n ZONE example.org

Zone signieren



Zone signieren, signierte Zone wird als wsXX.ws.dnssec.bayern.signed gespeichert

```
dnssec-signzone -K /var/named/keys -S -o wsXX.ws.dnssec.bayern \
/etc/bind/wsXX.ws.dnssec.bayern.zone
```

Wichtige Parameter im Zusammenhang mit dem Schlüssel-Management

-K directory
Key repository: Specify a directory to search for DNSSEC keys. If not specified, defaults to the current directory.

-o origin
The zone origin. If not specified, the name of the zone file is assumed to be the origin.

-S
Smart signing: Instructs dnssec-signzone to search the key repository for keys that match the zone being signed, and to include them in the zone if appropriate.

Veröffentlichen der Zone

Verweis auf Zone-Datei in /etc/bind/named.conf anpassen.

nsupdate wird verwendet, um die Zone zu veröffentlichen:

```
$ nsupdate <ENTER>
> server 10.156.8.23
> zone ws.dnssec.bayern
> update add wsXX.ws.dnssec.bayern. 300 IN DS <DS RECORD>
> send
```

Übung - DNSSEC in Bind 9.9 konfigurieren

Virtual Machine - dnssec-wsXX.dnssec.bayern

- Für jeden Teilnehmer existiert eine Virtuelle Maschine **dnssec-wsXX.dnssec.bayern** (**XX** die Teilnehmernummer **01-20**)

- Login auf jeder VM mit
  Benutzer: ***dnssecadmin***
  Passwort: ***DNSSEC@bayern***

  dnssecadmin@dnssec-wsXX:~$ ***sudo -s***

- BIND 9.9.5 installiert, einfaches Zonen-Beispiel in

  /etc/bind/wsXX.ws.dnssec.bayern.zone

- VMs sind aus dem MWN/Eduroam erreichbar

  ssh dnssecadmin@dnssec-wsXX.dnssec.bayern

Zone Datei auf dnssec-wsXX.dnssec.bayern

Eine einfache Zone ist in der Datei */etc/bind/ws01.ws.dnssec.bayern.zone* definiert:

```
$TTL 300
$ORIGIN ws01.ws.dnssec.bayern.

@    IN    SOA      dnssec-ws01.dnssec.bayern. root.dnssec-ws01.dnssec.bayern. (
      1     ; serial
      4h    ; refresh
      1h    ; retry
      2w    ; expire
      300 ; negative TTL
      )

      IN    NS  dnssec-ws01.dnssec.bayern.

dnssec-ws01 IN    A     138.246.99.206
dnssec-ws01 IN    AAAA     2001:4ca0:800:2:250:56ff:fe8f:5584
```

Übung - ZSK und KSK, Zone Signing

DNSSEC – Schritt für Schritt mit BIND 9.9

1. Zone Signing Key (ZSK) erzeugen
2. Key Signing Key (KSK) erzeugen
3. DS Resource Record erstellen
4. DS RR im Parent updaten
5. Zone signieren
6. Zone veröffentlichen

Zone Signing Key (ZSK) erzeugen

Zum Signieren der Resource Records wird ein ZSK mit 2048 Bit für die ZONE wsXX.ws.dnssec.bayern erzeugt, wobei XX die Nummer der ws-VM ist

dnssec-keygen -b 2048 -a RSASHA256 wsXX.ws.dnssec.bayern

Generating key pair.....+++ .....+++

Kws20.ws.dnssec.bayern.+008+44274

KEY-ID

Der ZSK wird später durch diese Key-ID in der Zone referenziert.

Key Signing Key (KSK) erzeugen

Zum Signieren des Zone Signing Keys wird ein KSK mit 2048 Bit für die ZONE wsXX.ws.dnssec.bayern erzeugt, angezeigt durch die Option „-f KSK“

dnssec-keygen -b 2048 -a RSASHA256 -f KSK wsXX.ws.dnssec.bayern

Generating key pair.....+++ .....+++

KwsXX.ws.dnssec.bayern.+008+13384

KEY-ID

Der KSK wird später durch diese Key-ID in der Zone referenziert.

Zone mit den ZSK und KSK signieren

dnssec-signzone signiert die RRsets mit dem ZSK und den ZSK RR mit dem KSK. „Smart Signing“ (Option –S) kümmert sich um das Finden der Schlüssel.

dnssec-signzone -S -o wsXX.ws.dnssec.bayern /etc/bind/wsXX.ws.dnssec.bayern.zone

Fetching KSK 1612/RSASHA256 from key repository.
Fetching ZSK 37338/RSASHA256 from key repository.
Verifying the zone using the following algorithms: RSASHA256.
Zone fully signed:
Algorithm: RSASHA256: KSKs: 1 active, 0 stand-by, 0 revoked
ZSKs: 1 active, 0 stand-by, 0 revoked
/etc/bind/ws20.ws.dnssec.bayern.zone.signed

NB: Smart Signing schaut nach allen gültigen ZSKs und KSK im lokalen Verzeichnis und verwendet diese. Option: –K /var/named/keys gibt ein anderes Verzeichnis an.

Delegated Signer RR (DS) aus KSK erzeugen

Um für DNSSEC einen Delegated Signer Resource Record an die Parent Zone übergeben zu können, muss der **DS Resource Record** dort eingetragen werden. **dnssec-signzone** erzeugt eine **Datei** „dsset-<zonename>“ **automatisch**.

Key ID des KSK

wsXX.ws.dnssec.bayern. IN DS 13384 8 1 9A623A8B2382B8802BDC9CDDEF490BAAB36A6F13

wsXX.ws.dnssec.bayern. IN DS 13384 8 2
4A30BB99C097CA1FFC056083FF27F4EC5D9600CC9057A0BA03283DC049B9C0C5

NB: Wie Sie sehen, werden DS RR immer für beide Hashing-Algorithmen (1=SHA-1 u. 2=SHA-256) erzeugt.

Mindestens einer muss, aber beide sollten an die Parent-Zone übergeben werden (siehe nächster Schritt).

Delegated Signer RR (DS) aus KSK erzeugen

Ein Delegated Signer Resource Record kann auch aus dem **public KSK (...+008+KSK-ID.key-Datei)** alternativ per Hand erzeugt werden, der an die Parent Zone übergeben werden kann. Führen Sie dafür folgenden Befehl aus:

dnssec-dsfromkey KwsXX.ws.dnssec.bayern.+008+01612.key

KEY-ID des vorher erzeugten KSK

wsXX.ws.dnssec.bayern. IN DS 1612 8 1 9A623A8B2382B8802BDC9CDDEF490BAAB36A6F13

wsXX.ws.dnssec.bayern. IN DS 1612 8 2
4A30BB99C097CA1FFC056083FF27F4EC5D9600CC9057A0BA03283DC049B9C0C5

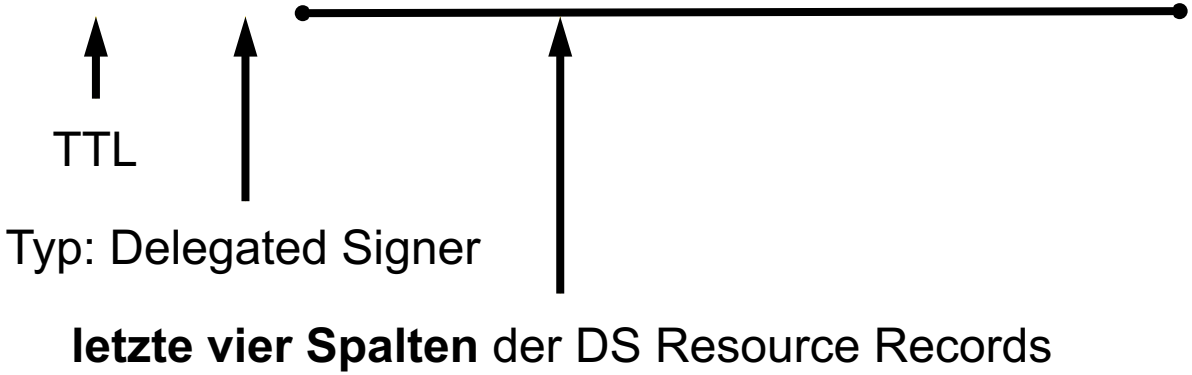
NB: Wie Sie sehen werden DS RR immer für beide Hashing-Algorithmen (1=SHA-1 u. 2=SHA-256) erzeugt.

Mindestens einer muss, aber beide sollten an die Parent-Zone übergeben werden (siehe nächster Schritt).

DS Record an die Parent-Zone übergeben

Die übergeordnete Domän („ws.dnssec.bayern“) nur dynamisch updaten lässt, muss der DS RR Eintrag über ein dynamisches **nsupdate** auf den NS 10.156.8.23 erfolgen:

nsupdate <ENTER>
> server 10.156.8.23
> zone ws.dnssec.bayern
> update add wsXX.ws.dnssec.bayern 300 IN DS 1612 8 1 9A623A8B2382B8802BD…
> update add wsXX.ws.dnssec.bayern 300 IN DS 1612 8 2 4A30BB99C097CA1FFC056...
> send
> quit



NB: Beim Signieren mit einem neuen KSK müssen die DS RR der Parent-Zone mit nsupdate aktualisiert werden.

Zone veröffentlichen

dnssec-signzone erzeugt eine neue Zonen-Datei „.signed“ als Ausgabe.

Darum muss **/etc/bind/named.conf** muss auf die **signierte Zone** verweisen!

```
options {
        directory "/var/cache/bind";
        dnssec-validation auto;
        listen-on-v6 { any; };
};

zone wsXX.ws.dnssec.bayern {
        type master;
        file "/etc/bind/wsXX.ws.dnssec.bayern.zone.signed";
};
```

Dann die Zone veröffentlichen, indem die Zonen-Konfiguration neu geladen wird.

rndc reload

server reload successful

Erfahrungen aus der DNSSEC-Praxis

DNSSEC am LRZ

- resolver1.lrz.de validierend seit 2008
- Teilnahme am DENIC-DNSSEC-Testbed, DLV, IANA ITAR
- resolver2.lrz.de validierend seit März 2014
- seitdem keine Auflösung von Domains mit kaputtem DNSSEC mehr!
- <5 Incidents die periphär durch DNSSEC hervorgerufen wurde

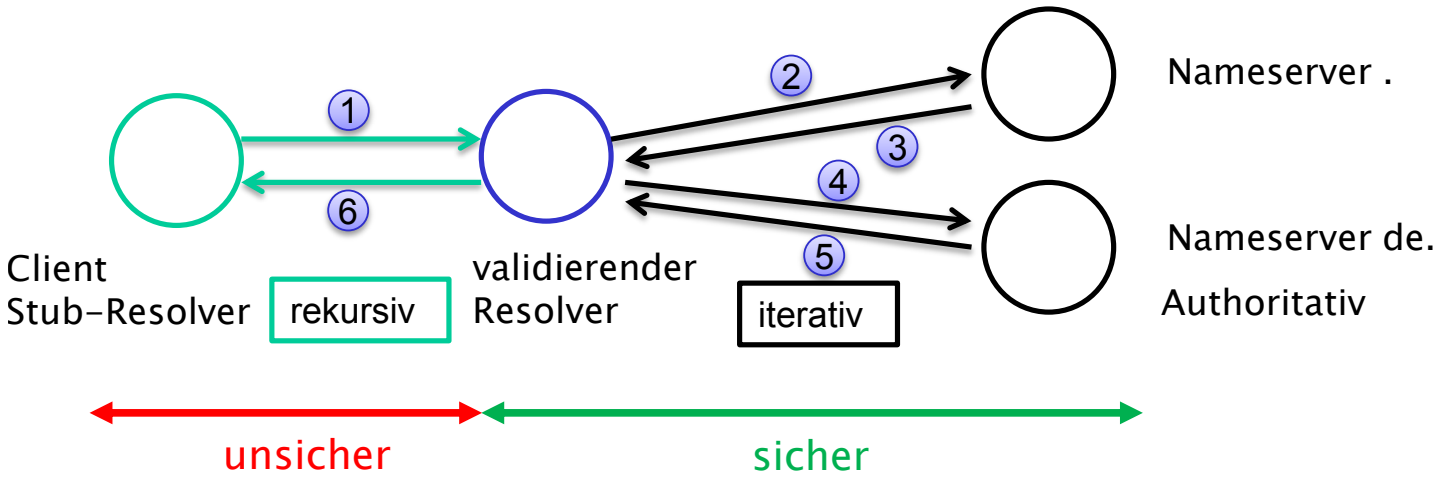
DNSSEC am LRZ

Autoritativer Nameserver

- Implementierung am LRZ als Signing-Proxy
- BIND 9.9 Inline-Signing auf Debian-VM
- kein Eingriff in bestehende Infrastruktur für !DNSSEC-Zonen

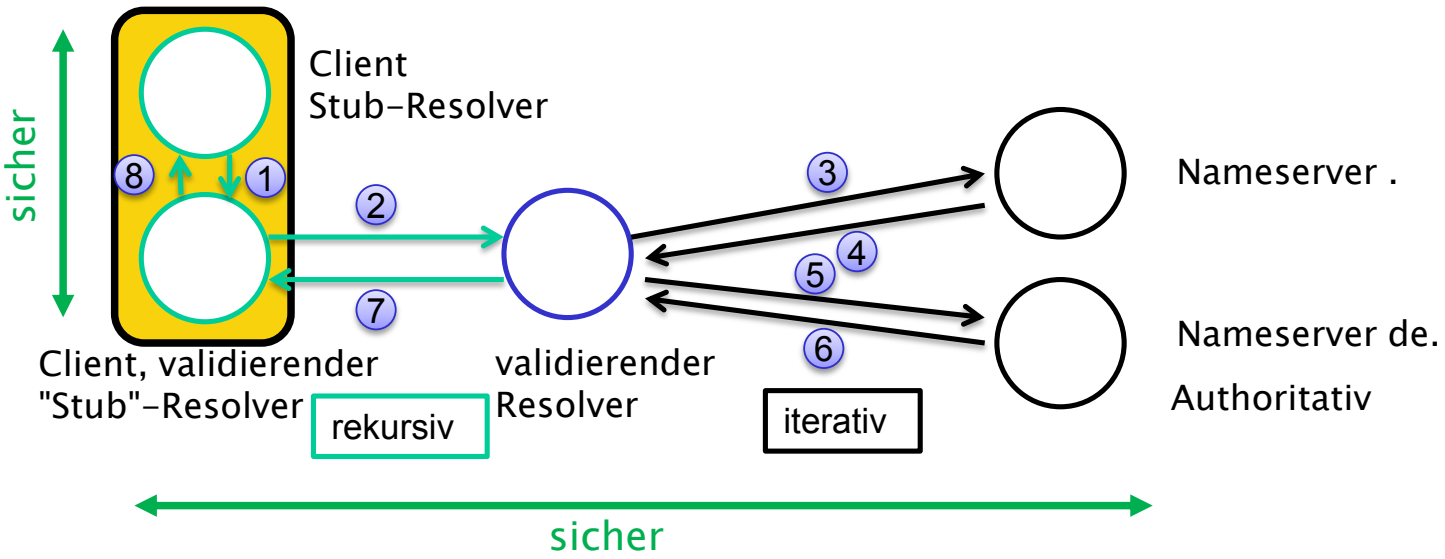
Rekursive Seite

- relativ trivial
- in „modernen“ Versionen von BIND und Unbound standardmäßig an
- DNSSEC wird validiert, Domains mit DNSSEC-Fehlern können nicht aufgelöst werden (SERVFAIL)
- Resolver gibt Validierungsstatus im *ad*-Flag zurück
- Zwei Probleme
  - BIND validiert keine Slave-Zonen
  - **keine** Authentifizierung zwischen Client und Resolver, Spoofing möglich



Rekursive Seite

- Lösung: lokal validierende Resolver (z.B. Unbound)



- am LRZ im Aufbau (Mailsystem, ausgewählte Mitarbeiter, ...)

Rekursive Seite

- resolver1.lrz.de validierend seit 2008
- Teilnahme am DENIC-DNSSEC-Testbed, DLV, IANA ITAR
- resolver2.lrz.de validierend seit März 2014
- seitdem **keine** Auflösung von Domains mit kaputtem DNSSEC mehr!
- <5 Incidents die periphär durch DNSSEC hervorgerufen wurden
  - **immer** durch Gegenseite verursacht

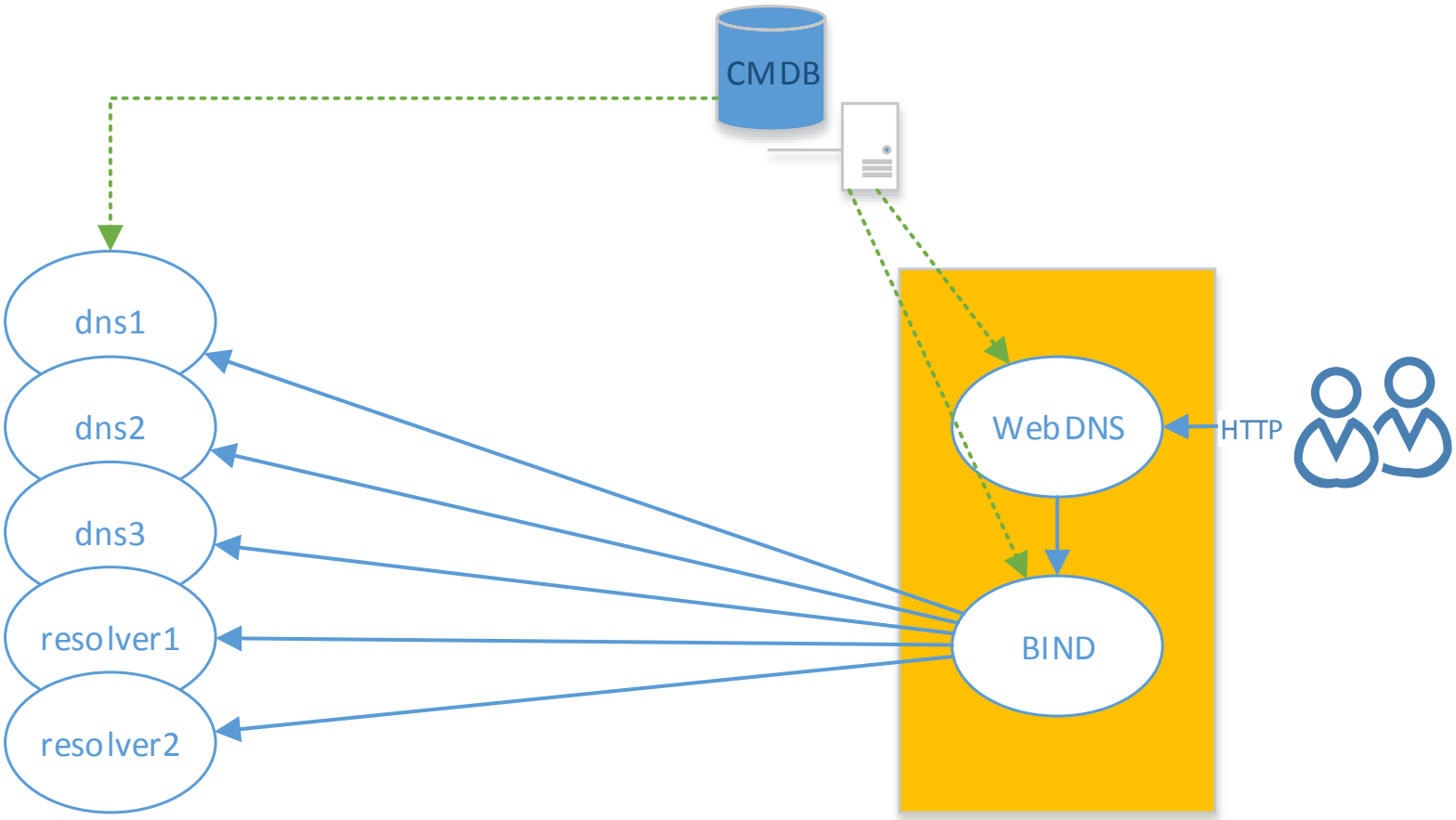
Authoritative Seite

- stellt signierte Zone für eigene Domains bereit
- grober Ablauf:
  - Generieren von Key-Signing-Key (KSK) und Zone-Signing-Key (ZSK)
  - Signieren der Zone mit dem ZSK
  - Signieren des ZSK mit dem KSK
  - Einfügen des Hashs des KSK in die Parent-Zone (DS-Record)
- alle authoritativen Server (auch Slaves) müssen DNSSEC-fähig sein
- Signieren muss bei jeder Änderung der Zone gemacht werden
- manueller Aufruf von `dnssec-signzone`
  - fehleranfällig!
  - muss auch ohne Änderung der Zone periodisch durchgeführt werden (RRSIG Validity Period)

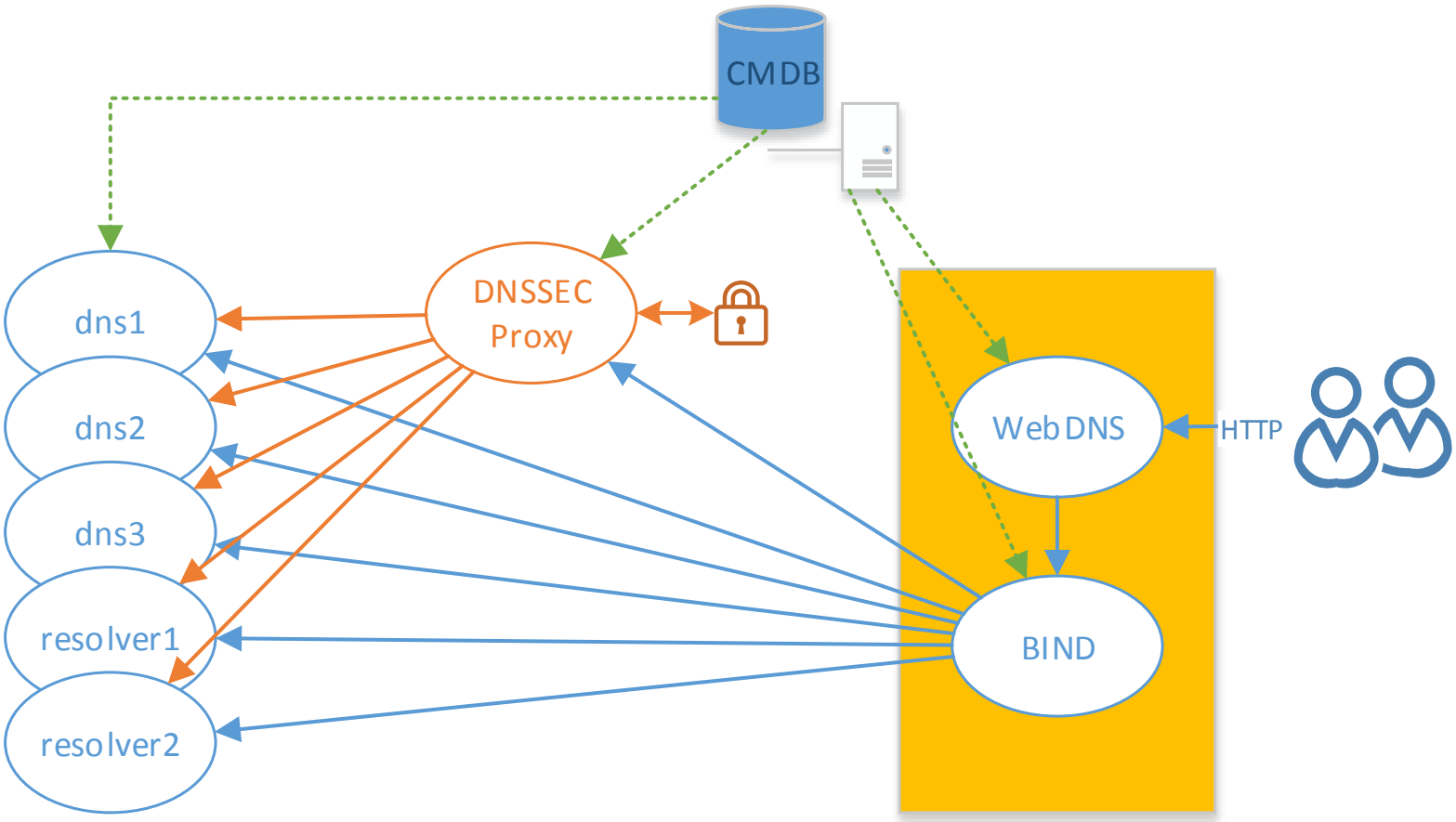
Authoritative Seite

- BIND 9.8 - „auto-dnssec maintain“ Feature
  - Key in Key-Directory anlegen
  - BIND kümmert sich automatisch um Resigning
  - Änderungen an der Zone mit rndc freeze/thaw oder nsupdate
- BIND 9.9 - „inline-signing“ Feature
  - siehe oben
  - Input nicht mehr Zonenfile, sondern AXFR
  - ermöglicht die Einrichtung eines Signing Proxies
- OpenDNSSEC
  - automatisiert mehr (z.B. Keyrollover)
  - komplexere Installation (benötigt MySQL und SoftHSM)
  - unterstützt ebenfalls Inline Signing

Authoritative Seite am LRZ



Authoritative Seite am LRZ



Authoritative Seite am LRZ

- BIND 9.9 Inline Signing-Proxy auf Debian Wheezy+Backports
- RSASHA256 mit 2048 Bit KSK/ZSK
- DNSSEC-fähige Registrars DFN und InternetX
- Nebenschauplatz: Aufteilung Nameserver auf drei TLDs

- Große Hauptdomains mit Infrastruktur signiert (wenn's kracht dann richtig)
  - lrz.de - 28.10.2014
  - tum.de - 10.12.2014
  - lmu.de - 12.01.2015
- einige kleinere Domains ebenfalls, zum Teil schon länger
  - badw-muenchen.de - Dezember 2010

Monitoring

- DNSSEC-Signierungsfehler sind tödlich
- fallen je nach lokaler Resolverstruktur gar nicht auf

- Tägliche Prüfung am LRZ für jede DNSSEC-Zone
  - Prüfen der signierten Zone durch ldns-verify-zone (ldns[1])
    - prüft NSEC-Chaining, Keys, RRSIG-Validity (>30 Tage)
  - Prüfen der signierten Zone durch dnssec-verify (BIND)
    - prüft NSEC-Chaining, Keys
  - Abfrage des SOA-Records bei Google DNS, DNS-OARC ODVR[2]
    - prüft sichere Delegation (ad-Flag)
    - prüft Funktionsfähigkeit aus Nutzersicht
- Nutzung lokaler validierender Resolver auf wichtigen Servern

[1] http://www.nlnetlabs.nl/projects/ldns/
[2] https://www.dns-oarc.net/oarc/services/odvr

Status weltweit

Rekursiv:

- weltweit etwa 12% der DNS-Anfragen mit DNSSEC gesichert[3]
- Comcast (größter Privatkundenprovider der Welt) seit Anfang 2012
- Google DNS (8.8.8.8 und 8.8.4.4) seit März 2013

Authoritativ:

- bund.de, bayern.de
- ruhr-uni-bochum.de, tuhh.de, uni-kl.de, hs-owl.de, uni-rostock.de, uni-stuttgart.de, mpg.de, ...
- >500.000 in .com/.net/.edu [4]
- „Pflicht“ für US Regierungsstellen (.gov)

[3] http://stats.labs.apnic.net/dnssec/XA

[4] http://scoreboard.verisignlabs.com/

Voraussetzungen für DNSSEC

Voraussetzungen für DNSSEC

- Software
- Netzwerk
- Hardware
- Sicherheitsanforderungen

Voraussetzungen für DNSSEC - Software

- BIND Version mindestens 9.7 (ab 9.8 managed key rollover)
- OpenDNSSEC
- Unbound (1.4): resolving Nameserver mit DNSSEC Unterstützung
- Windows Server 2012 (empfohlen) ab Windows Server 2008 R2 limitierter Support

Voraussetzungen für DNSSEC - Netzwerk

- Switche/Router müssen Pakete >512 Byte unterstützen
- Firewalls TCP Port 53 offen
- MTU-Discovery freigeschaltet
- Firewalls DNSSEC-Pakete >512 Byte durchlassen

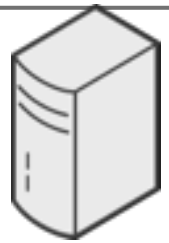
Voraussetzungen für DNSSEC - Hardware

Autoritative Nameserver

- Public key encryption ist rechenintensiv aber mit moderner Serverhardware kein Problem
- Zone-Dateien mindestens 3x so groß (-> RAM-Anforderung größer)

Voraussetzungen für DNSSEC - Hardware

Resolving Nameserver



- Public key encryption ist rechenintensiv aber mit moderner Serverhardware (>2005) kein Problem
- Zone-Dateien mindestens 3x so groß (-> RAM-Anforderung größer)

Voraussetzungen für DNSSEC - Sicherheit

- ZSK/KSK Paar pro Zone ist zu empfehlen
- … oder zumindest für wichtige Zonen
- Private Key Signing Keys müssen an sicherem Ort aufbewahrt werden:

  Hardware Security Module **oder** Datenträger im Tresor
- Konsequentes Key management / rollover (Wer ist verantwortlich?)

DNSSEC - Zusammenfassung

DNSSEC - Vorteile für DNS

- Vor Cache impersonation Angriffen geschützt
- DNS Spoofing / Cache-poisoning Angriffe werden verhindert
- Zone-Veränderung auf Slave-Nameservern nicht mehr möglich
- Nutzer vor Umleitungen auf Servern von Hackern gewarnt

DNSSEC - Nachteile der Nutzung

- Implementation der Konfiguration etwas aufwändig
- Zonen und DNS-Antworten werden größer
- Signatur-Validierung kostet CPU-Ressourcen
- Key rollover muß gemanaged werden
- DDOS-Attacken werden verstärkt durch die größeren Pakete
- Aggressive NSEC3 caching bietet (bald) Entlastung von DDOS auf autoritative NS

NSEC/NSEC3 - Nichtexistenz von Objekten

NSEC - Next Secure Record

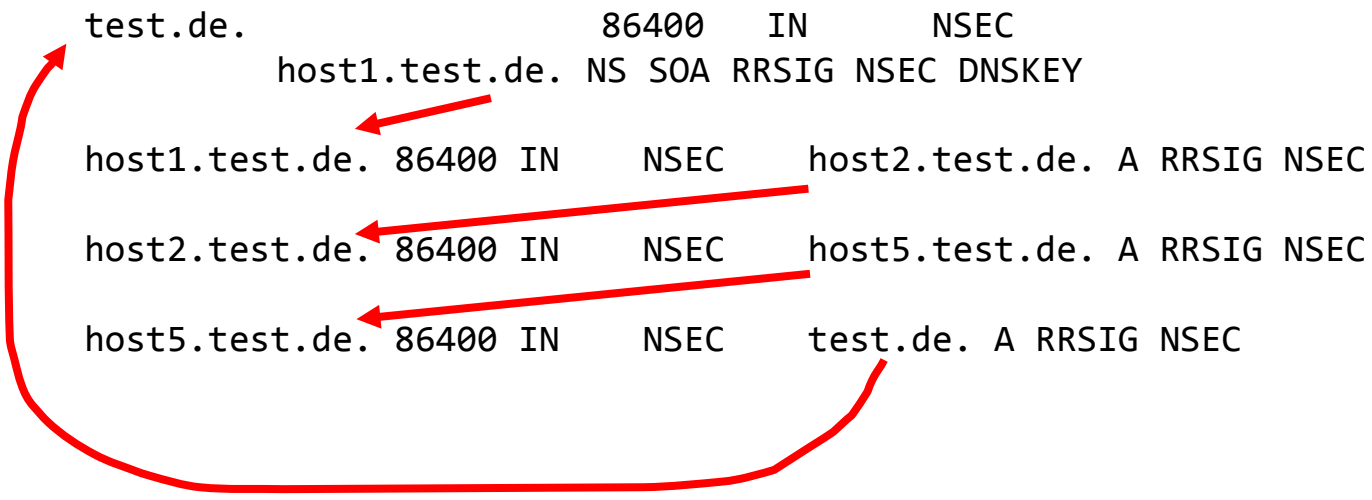
- DNSSEC sichert die Authentizität von DNS-Einträgen mit Signaturen
- NSEC gibt den nächsten Eintrag einer Zone und welche Typen für einen erfragten Namen existieren
- Authentische Nicht-Existenz von Einträgen
- Dynamisches Erstellen von „xyz.lrz.de existiert nicht“ ist zu rechenaufwändig und verstößt gegen Sicherheitsrichtlinien
- NSEC stellt sicher, dass sich zwischen zwei Einträgen kein weiterer befindet

NSEC

- RRSIG können nur die Korrektheit eines Eintrags beweisen, aber nicht deren (Nicht-)Existenz
- dynamisches Erstellen von signierten Antworten ("xyz.lrz.de existiert nicht") zu CPU-intensiv und verstößt gegen Sicherheitsmodell
- Lösung NSEC-Records (Next SECure):
  - Sortiert aufeinanderfolgende Namen werden beim Signieren mit Hilfe von NSEC Records verpointert, so dass eine geschlossene Kette entsteht
  - Eine Anfrage nach einem nicht existierenden Namen wird mit dem NSEC Record beantwortet, der den nicht existierenden RR umklammert
- Nichtvorhandensein von Namen kann bewiesen werden

NSEC

- NSEC-Records enthalten auch alle vorhandenen RRTYPE des Namens → (Nicht-)Existenz eines bestimmten RRTYPE bewiesen
- Beispiel:



NSEC - Beispiel 1

Zone-Datei

ant.lrz.de	NSEC	baby.lrz.de	A	AAAA	NSEC	RRSIG
baby.lrz.de	NSEC	cat.lrz.de	A	AAAA	NSEC	RRSIG
cat.lrz.de	NSEC	dodo.lrz.de	A	AAAA	NSEC	RRSIG
dodo.lrz.de	NSEC	mouse.lrz.de	A	AAAA	NSEC	RRSIG
mouse.lrz.de	NSEC	lrz.de	A	AAAA	NSEC	RRSIG
lrz.de	NSEC	www.lrz.de	A	AAAA	NSEC	RRSIG
www.lrz.de	NSEC	ant.lrz.de	A	AAAA	NSEC	RRSIG

A für fruit.lrz.de?

Existiert nicht! Kein Eintrag zwischen dodo und mouse

dodo.lrz.de	NSEC	mouse.lrz.de	A	AAAA	NSEC	RRSIG

Signatur über NSEC Eintrag

NSEC - Beispiel 2

Zone-Datei

ant.lrz.de	NSEC	baby.lrz.de	A	AAAA	NSEC	RRSIG
baby.lrz.de	NSEC	cat.lrz.de	A	AAAA	NSEC	RRSIG
cat.lrz.de	NSEC	dodo.lrz.de	A	AAAA	NSEC	RRSIG
dodo.lrz.de	NSEC	mouse.lrz.de	A	AAAA	NSEC	RRSIG
mouse.lrz.de	NSEC	lrz.de	A	AAAA	NSEC	RRSIG
lrz.de	NSEC	www.lrz.de	A	AAAA	NSEC	RRSIG
www.lrz.de	NSEC	ant.lrz.de	A	AAAA	NSEC	RRSIG

AAAA für baby.lrz.de?

Existiert nicht! Es ist nicht in der Liste im NSEC Record

baby.lrz.de	NSEC	cat.lrz.de	A	NSEC RRSIG

Signatur über NSEC Eintrag

NSEC Record - Detail und Zone Walking

- Zeigt auf den nächsten Eintrag in der Zone
- Zeigt auch alle existierenden RRs für einen Domain-Besitzer
- Letzter NSEC Eintrag zeigt wieder auf den ersten (“Ring”)
- Erlaubt immer noch Entdecken der Zone (“Zone walking”)

```
www.lrz.de. 3600 IN    NSEC ant.lrz.de.    A        RRSIG NSEC
```

NSEC Record - Detail und Zone Walking

- Zeigt auf den nächsten Eintrag in der Zone
- Zeigt auch alle existierenden RRs für einen Domain-Besitzer
- Letzter NSEC Eintrag zeigt wieder auf den ersten (“Ring”)
- Erlaubt immer noch Entdecken der Zone (“Zone walking”)

Besitzer

```
www.lrz.de.  3600 IN  NSEC ant.lrz.de.  A  RRSIG NSEC
```

NSEC Record - Detail und Zone Walking

- Zeigt auf den nächsten Eintrag in der Zone
- Zeigt auch alle existierenden RRs für einen Domain-Besitzer
- Letzter NSEC Eintrag zeigt wieder auf den ersten (“Ring”)
- Erlaubt immer noch Entdecken der Zone (“Zone walking”)

NSEC Record - Detail und Zone Walking

- Zeigt auf den nächsten Eintrag in der Zone
- Zeigt auch alle existierenden RRs für einen Domain-Besitzer
- Letzter NSEC Eintrag zeigt wieder auf den ersten (“Ring”)
- Erlaubt immer noch Entdecken der Zone (“Zone walking”)

Existierende Resource Record Typen für www.lrz.de

```
www.lrz.de. 3600 IN  NSEC ant.lrz.de.  A  RRSIG NSEC
```

NSEC - Problem “Zone Walking”

- Erlaubt die Rekonstruktion eines Zone-Files
- Datenschutz-Probleme
- Informationen können für Angriffe verwertet werden

Lösung: NSEC3 Resource Record

- Wie NSEC
- Aber die Namen sind hashed, um Zone-Entdeckung zu verhindern
- Hashed Namen sind geordnet
- Letzter Eintrag verweist auf ersten (“Ring”)

NSEC3

- Verpointerung durch NSEC Records erlaubt das Auslesen vollständiger Namespaces (Zone Walking) (ldns-walk)
- Abhilfe: Hashes statt Namen verketten, Salt und Hashrundenzahl pro Zone festgelegt

```
test.de.                                    NSEC3PARAM      1 0 1 000102

ang2tj60t4i0loeoffdjtq28arbnug4l.test.de. NSEC3         1 0 1 000102
                                gcn5e441oqahljld7m8jg0lff5ccnss2 A AAAA RRSIG
; host2.test.de -> host5.test.de

gcn5e441oqahljld7m8jg0lff5ccnss2.test.de. NSEC3         1 0 1 000102
                                oe8oml2vcngi1i0gktjs6q80t4jjrdl2 TXT RRSIG
; host5.test.de -> test.de

oe8oml2vcngi1i0gktjs6q80t4jjrdl2.test.de. NSEC3         1 0 1 000102
                                pbh1l8qs3j1iuiocnkmp8igfci1pgjcf NS SOA […]
; test.de -> host1.test.de

pbh1l8qs3j1iuiocnkmp8igfci1pgjcf.test.de. NSEC3         1 0 1 000102
                                ang2tj60t4i0loeoffdjtq28arbnug4l A RRSIG
; host1.test.de -> host2.test.de
```

NSEC3 - Beispiel

Zone-Datei

```
df67wer9x1    NSEC3 ed5g8rt69v     A  AAAA    NSEC3 RRSIG
ed5g8rt69v    NSEC3 ftyro47f75     A          NSEC3 RRSIG
ftyro47f75    NSEC3 h3aq475y76q    A  AAAA    NSEC3 RRSIG
h3aq475y76q   NSEC3 iz45wt6P3d     A          NSEC3 RRSIG
iz45wt6P3d    NSEC3  jf8r8yt64j    A  AAAA    NSEC3 RRSIG
jf8r8yt64j    NSEC3 kt8y0gur9a     A  AAAA MX NSEC3 RRSIG
kt8y0gur9a    NSEC3 df67wer9x1     A  AAAA    NSEC3 RRSIG
```

A für fruit.lrz.de?

Existiert nicht! Es gibt keinen Eintrag zwischen h3aq475y76q und iz45wt6P3d!

```
h3aq475y76q NSEC3 iz45wt6P3d    A          NSEC3 RRSIG
```

RRSIG über NSEC

Übung - NSEC authentische Nichtexistenz

Zone-Walking mit ldns-walk

ldns-walk aus den ldns Tools ermöglicht bei **nicht NSEC3-gesicherten** Zonen aufgrund der durch NSEC zurück gelieferten „nächsten sicheren Einträgen“ Zone-Walking, d.h. das **Auslesen aller Einträge** einer Zone.

ldns-walk wsXX.ws.dnssec.bayern

ws20.ws.dnssec.bayern. ws20.ws.dnssec.bayern. NS SOA MX RRSIG NSEC DNSKEY
dnssec-ws20.ws20.ws.dnssec.bayern. A AAAA RRSIG NSEC

Lösung:

NSEC3 hashed die Zoneneinträge in der signierten Zonen-Datei. Siehe nächste Folie...

NSEC3-Signieren einer Zone

dnssec-signzone wird der zusätzliche Parameter „**-3**“ (d.h. NSEC3 statt NSEC verwendet) zusammen mit einem zufälligen „Salt“-Wert (hier: 674f1f001cbed616**)** übergeben:

dnssec-signzone -S -3 674f1f001cbed616 -o wsXX.ws.dnssec.bayern /etc/bind/wsXX.ws.dnssec.bayern.zone

Fetching KSK 1612/RSASHA256 from key repository.
Fetching ZSK 37338/RSASHA256 from key repository.
Verifying the zone using the following algorithms: RSASHA256.
Zone fully signed:
Algorithm: RSASHA256: KSKs: 1 active, 0 stand-by, 0 revoked
ZSKs: 1 active, 0 stand-by, 0 revoked
/etc/bind/ws20.ws.dnssec.bayern.zone.signed

NB: Um einen guten Zufallswert für NSEC3 zu erhalten, empfiehlt sich ihn vorher aus /dev/random zu genieren:

head -c 512 /dev/random | sha1sum | cut -b 1-16

674f1f001cbed616

ldns-walk einer NSEC3-signierten Zone

Wird ldns-walk nun mit denselben Parameter (Zonen-Name), aber nach NSEC3-Signieren der Zone aufgerufen, ergibt das Zone-Walking nur noch gehashte Ergebnisse:

ldns-walk wsXX.ws.dnssec.bayern.zone

1IPIHLB0MCSA9F5UTO5JPUP1RM37IKCE.zone. 0 IN RRSIG NSEC3 8 2 86400 20161205212900 20161105211620 25890 zone. f2eFGG98goHcSB2kbziZsKUYF4sTgXMUfGlMl3cnGCSpmFZgAx6LkrflO2o pUjGRjV7HiJhmzRdb7bmMdRo3vVfvSr1KN7rrsU5jX32WE+UsL/mMU9UrnrG7BkqI1qqgVF/sD/UY7VBTHTA1dMnVckQG 892eOtnPzsZzMwSl1LuWeTvemKZrKfSScPvNKlYfQrQK14gGGpGQj/3nZECgVA== 1IPIHLB0MCSA9F5UTO5JPUP1RM37IKCE.zone. 0 IN NSEC3 1 1 1 5853dde5 1rvm140hcqmc1jpak9sjjapr6pvvkc81 NS DS RRSIG 8B6458CBGSDSSOA6DQ3VF6L4LLQP9LRC.zone. 0 IN RRSIG NSEC3 8 2 86400 20161208130655 20161108120655 25890 zone. S66cy5/aO1DBkOdX5OdHD4S6iOcVDn2dG1+SgJ8sYzTDLhTg02+0fp1pZGZ/ mz3uB7RzcsdrV6I2JlY35vzk3REPO3J5GyKJt10B26MFycOeQw7yQyTZENwonjZo+ix3/c3SWP3EwnwJkOr7QuZ9lJ/8g3HgSs ToIKI1/IVxoajxSeIAgd9oF0kCZq4LiS52Xw2TNMmvB8J6n8mDVDW/tw== 8B6458CBGSDSSOA6DQ3VF6L4LLQP9LRC.zone. 0 IN NSEC3 1 1 1 5853dde5 8d8esci4g6 d3a0tv0plkb8o70jpqe4i8 NS DS RRSIG C2TIK4MJGGLGCSEASCU9RQCM4PEGJN76.zone. 0 IN RRSIG NSEC3 8 2 86400 20161205 194707 20161105194047 25890 zone. rUiCJLu76D1vnnk4dQioPaP4aLMWjOAscHVuADuk9qgUNfd+UTH/bfE8ohvz9c/l3xEKy VR3e20stjGLBf+Ad5cpM4ngLwaJJB6Q0GxLPhnPaisnw9UuN40FLc4X3uocmvdgPz1E+8QgPsM0fQ8h1Q2g0XrUSS6+orJaeYu t06w3OAqpRiewnWvlEQH/4y7PlF8Yor2RwJVe3huFAN3Ugg== C2TIK4MJGGLGCSEASCU9RQCM4PEGJN76.zone. 0 IN NSEC3 1 1 1 5853dde5 c311nn83l2 0dsquntthiuf2sggenj78t NS SOA RRSIG DNSKEY NSEC3PARAM TYPE65534

Key Rollovers

Schlüssel müssen getauscht werden

- Schlüssel veralten bald
  - Neue exploits werden jeden Tag offen gelegt
  - “brute force” wird zunehmend machbar
- Schlüssel können gestohlen oder kompromittiert werden
- Man braucht einen Plan

Key rollover Methoden

- Vor-Veröffentlichung (“pre-publish”)
- Doppelte Singnaturen (“double signature”)
- Für ZSK und KSK
  - Einen KSK zu tauschen bedeutet DS records zu verändern
- “Rollover”-Zeiten hängen von TTL und der Methode ab
- Schlüssel zu (alten) RRSigs müssen vorhanden sein

Pre-publish Methode

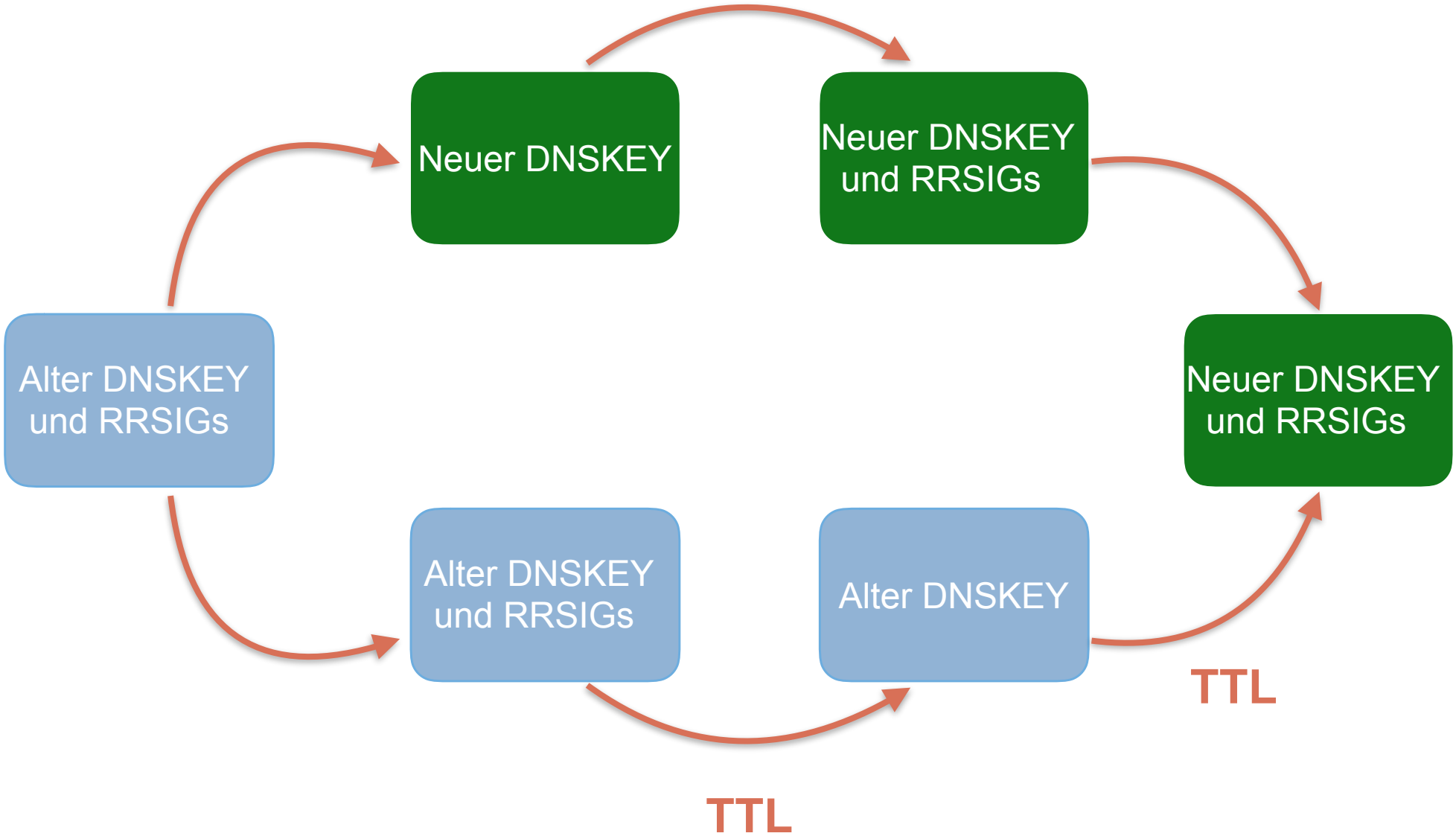


Bild: RIPE NCC

Pre-publish Methode

- Ein neuer DNSKEY record with mit dem neuen Schlüssel eingeführt
  - allerdings noch nicht zum Signieren verwendet

- Nachdem die TTL abgelaufen ist, werden RRSIGs mit den neuen DNSKEY erzeugt
  - alter DNSKEY wird weiterhin veröffentlicht

- Nachdem die TTL erneut abgelaufen ist, wird der alte DNSKEY entfernt

- DNSKEY / RRSIGs müssen immer in der Zone auffindbar sein (Key_A zu RRSIG_Key_A darf nicht fehlen), TTL beachten

Double Signature Methode

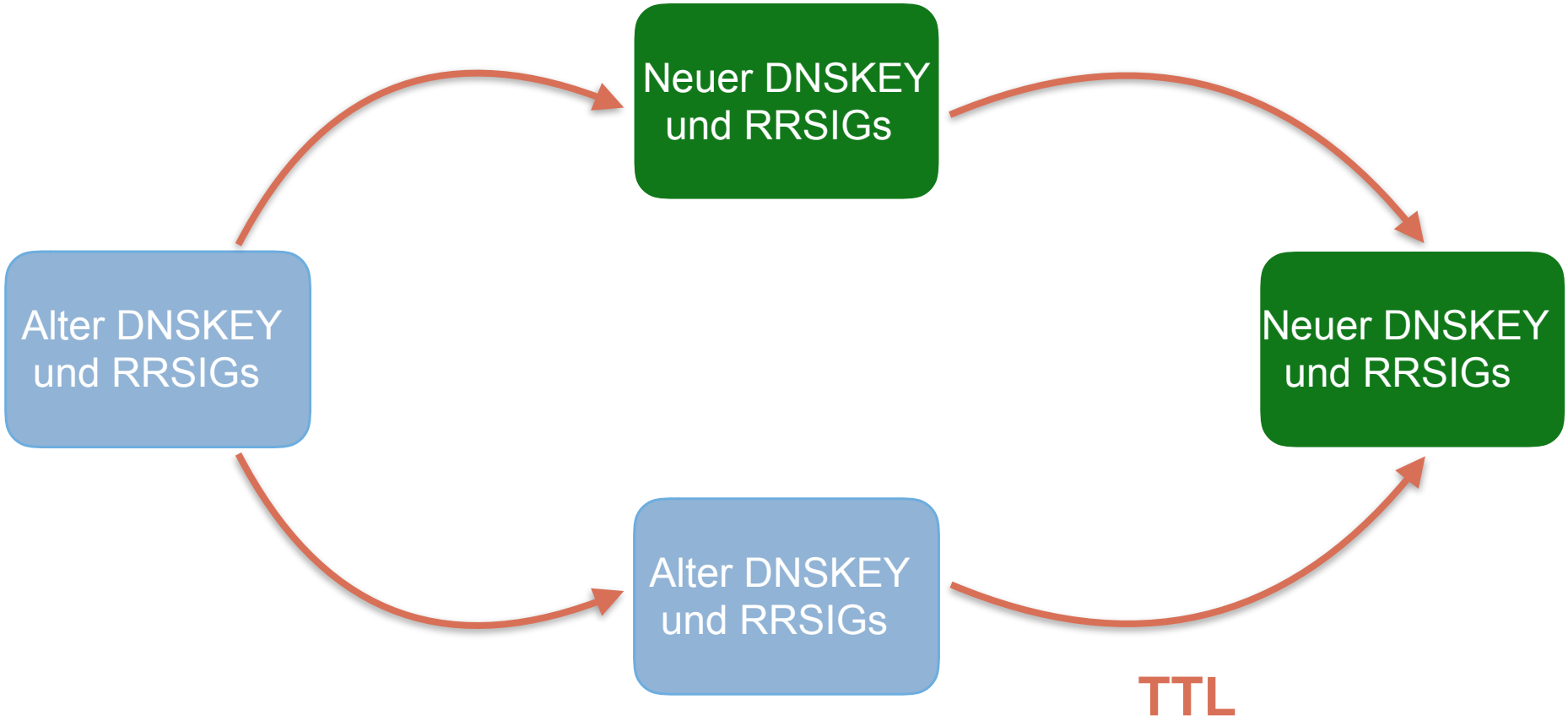
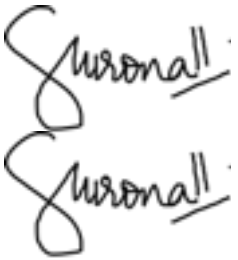


Bild: RIPE NCC

Double Signature Methode

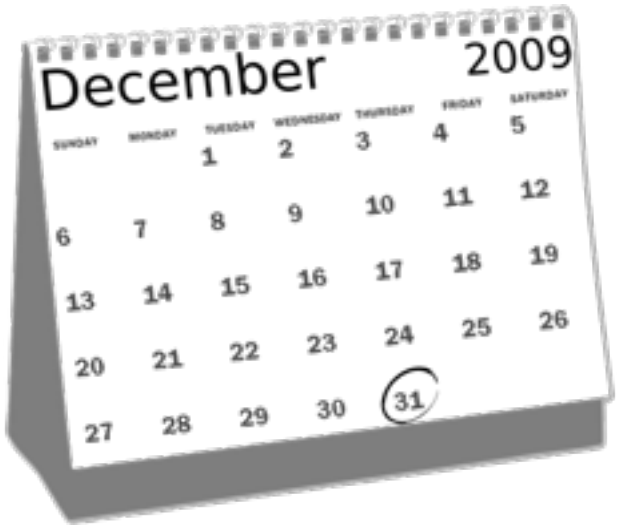


- Ein neuer DNSKEY wird eingeführt und sofort zum Signieren der Records verwendet
- Es gibt zwei RRSIGs für jeden Record, mit Signaturen von beiden DNSKEYs → Zone-Dateien **doppelt** so groß
- Nachdem die TTL abgelaufen ist, wird der alte DNSKEY entfernt, und Records werden wieder nur einmal signiert

Muss ich an den Rollover denken?

- Nein, er kann vorbereitet werden (z.B. BIND >9.8)
  - in der Konfiguration
  - inklusive des Zeitplans
- Durchführung hängt aber vom DNS-Admin ab
- DNSSEC keys für den nächsten rollover müssen rechtzeitig bereit liegen

Schlüssel in der Praxis



- Ein Schlüssel besitzt 5 wichtige Daten:
  - Veröffentlichung
  - Aktivierung
  - Deaktivierung
  - Zurückziehung
  - Löschung
- korrespondierende **dnssec-keygen** Optionen:
  -P publication date
  -A activation date
  -R revocation date
  -I retIrement date
  -D deletion date

- BIND mit **auto-dnssec** verwendet die Schlüssel im Rahmen dieser konfigurierten Zeiten, erzeugt aber **keine neuen**

Ratschläge zur Schlüsselverwaltung

- Verwende pre-publishing für ZSK

  - insbesondere bei großen Zonen

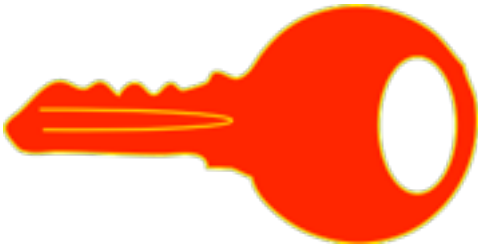
- Verwende double signature für KSK
  - KSK signiert DNSKEY doppelt, nicht die Zone

- Für KSK rollovers, DS records updaten

Ratschläge zur Schlüsselverwaltung

- Schlüssel regelmäßig wechseln!
- Rotation ~ alle 2 Jahre durchführen
- Schlüssel können 2 bis 4 Jahre im Voraus auf Vorrat erzeugt werden
- Das verschiebt aber nur das Problem, an neue Schlüssel muss (irgendwann) gedacht werden

Wechsel des Root-Keys kritisch



- Am **11.Oktober 2017** wird ICANN den DNSSEC .root-Key wechseln
- .root-Key wird bei der Einrichtung des DNSSEC-fähigen Nameserver herunter geladen (BIND >9.7 automatisch)
- **Bis** zum Zeitpunkt des .root-Key-Wechsels muss der neue auf dem Nameserver vorhanden sein
- Sonst schlagen alle DNSSEC-Validierungen fehl und alle DNSSEC-authentifizierten Zonen sind nicht erreichbar

Zeitplan des root KSK Rollovers

- 27. Oktober 2016 : neuer KSK wurde erzeugt
- Februar 2017: Veröffentlichung auf http://data.iana.org/root-anchors/
- 11. Juli 2017: neuer KSK wird im DNS veröffentlicht
- 11. Oktober 2017: neuer KSK wird zum Signieren verwendet
- Januar 2018: Rücknahme des alten KSK
- März 2018: Sichere Vernichtung des alten KSK und Abschluss des Key-rollover Prozesses

DANE - Grundlagen

DANE - Was ist das?

- “Domain name based authenticated named entity”
- Einem Objekt kann ein Zertifikat authentisch zugeordnet werden
- DNSSEC garantiert Authentizität für einen Eintrag auf DNS
- TLSA = TLS certificate association (RFC6698), neuer RR Eintrag
- Public Key erlaubt Verifizierung dieses Eintrags über den DNS

Warum DANE? - Zertifikat-Probleme

- Basiert auf chain-of-trust
- root-Certificate Authority als root-Anchor
- Kompromittierte CAs z.B. Comodo (OCR-Fehler, 19.Okt. 2016) oder DigiNotar BV, Wosign, Startcom

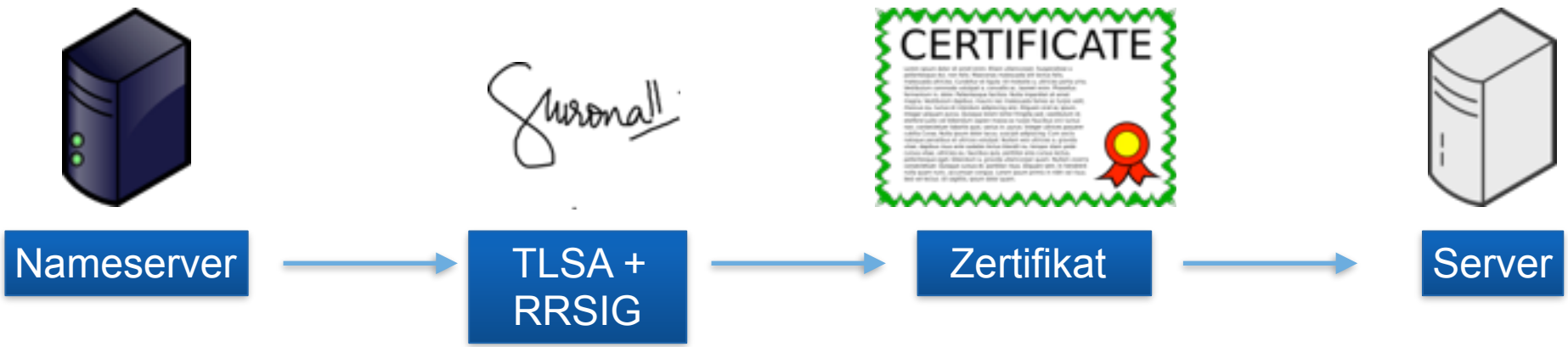
Root CA

CA (DFN)

LRZ

DNSSEC - garantiert Authentizität ohne CA

- DNS Name und Dienst wird mit Zertifikat assoziiert (“pinning”)
- Unabhängig von CAs, kann selbst-signiertes Zertifikat sein
- Vertrauenskette



DANE - DNS-based authenticated named Entity

Resolving
Nameserver

Autoritativer
Nameserver

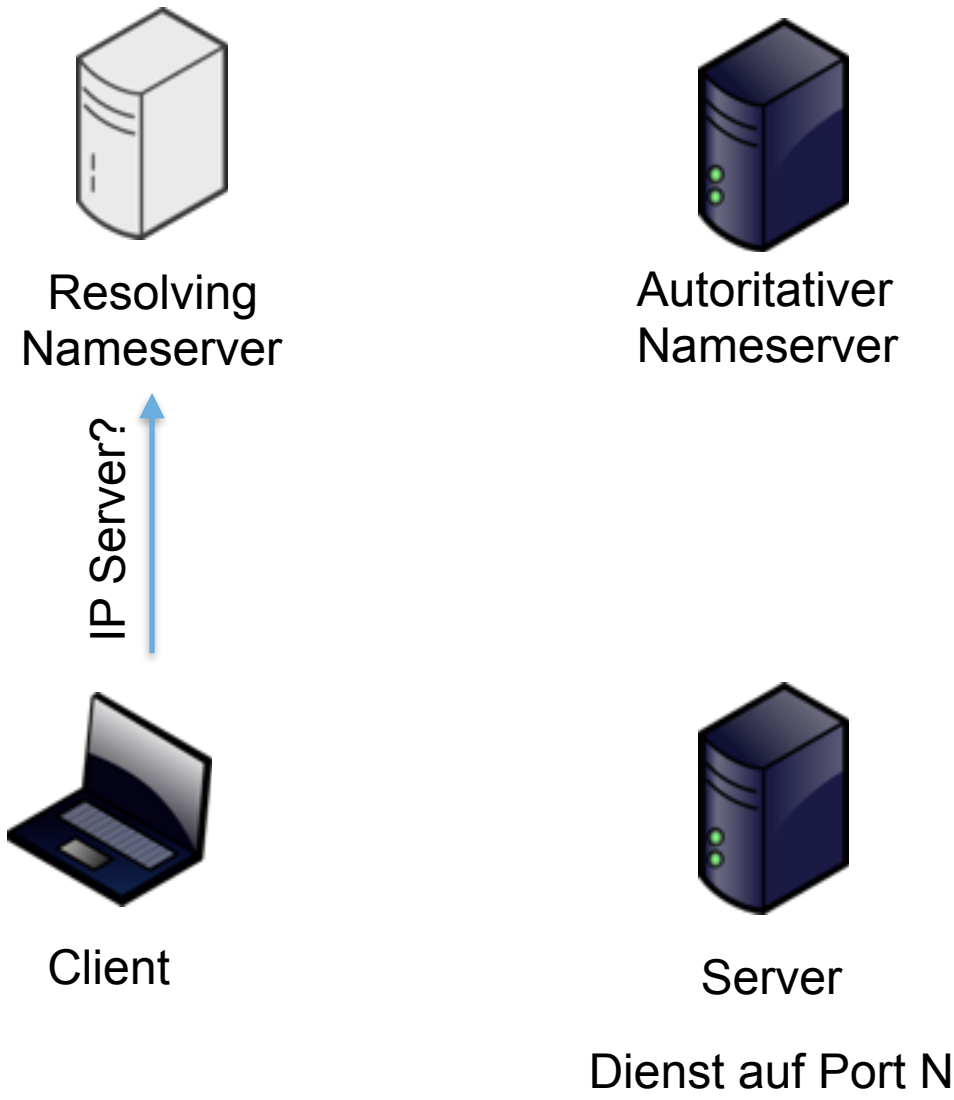
Client

Server

Dienst auf Port N

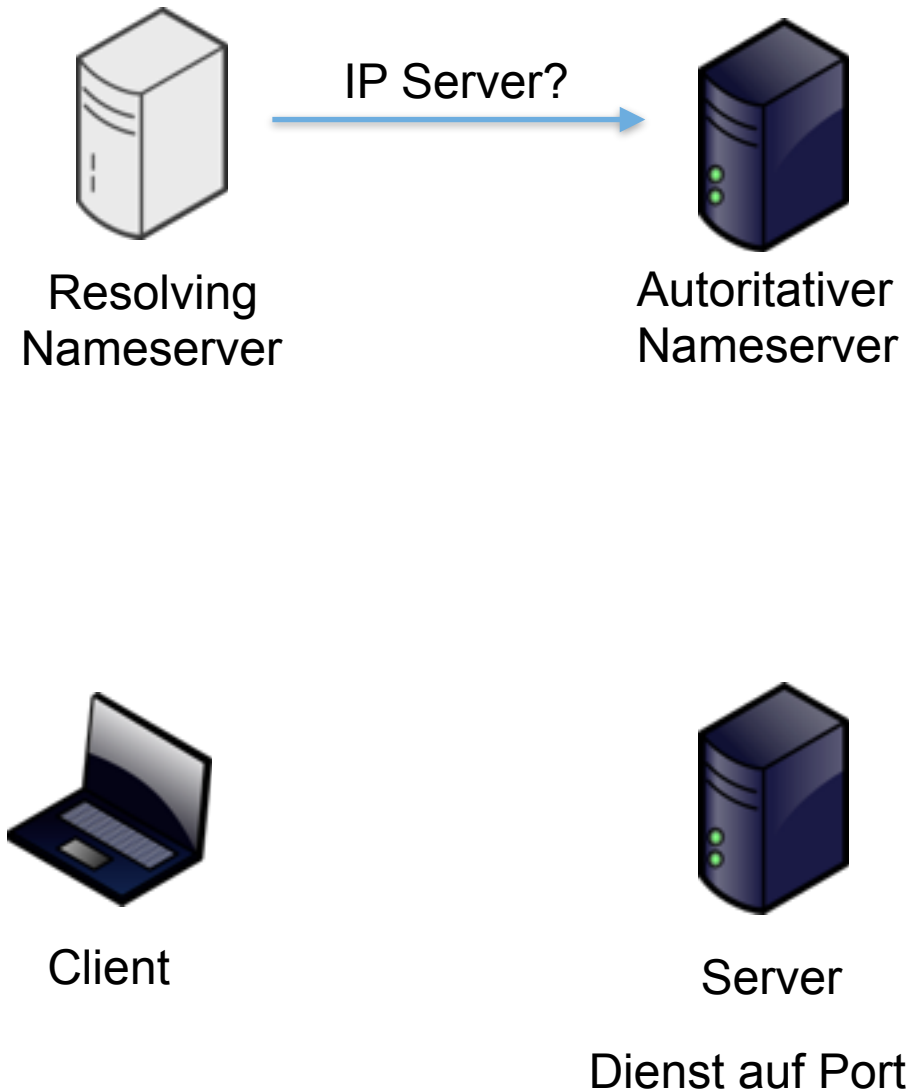
DANE - DNS-based authenticated named Entity

1. Client fragt Dienst auf Port N auf Server an



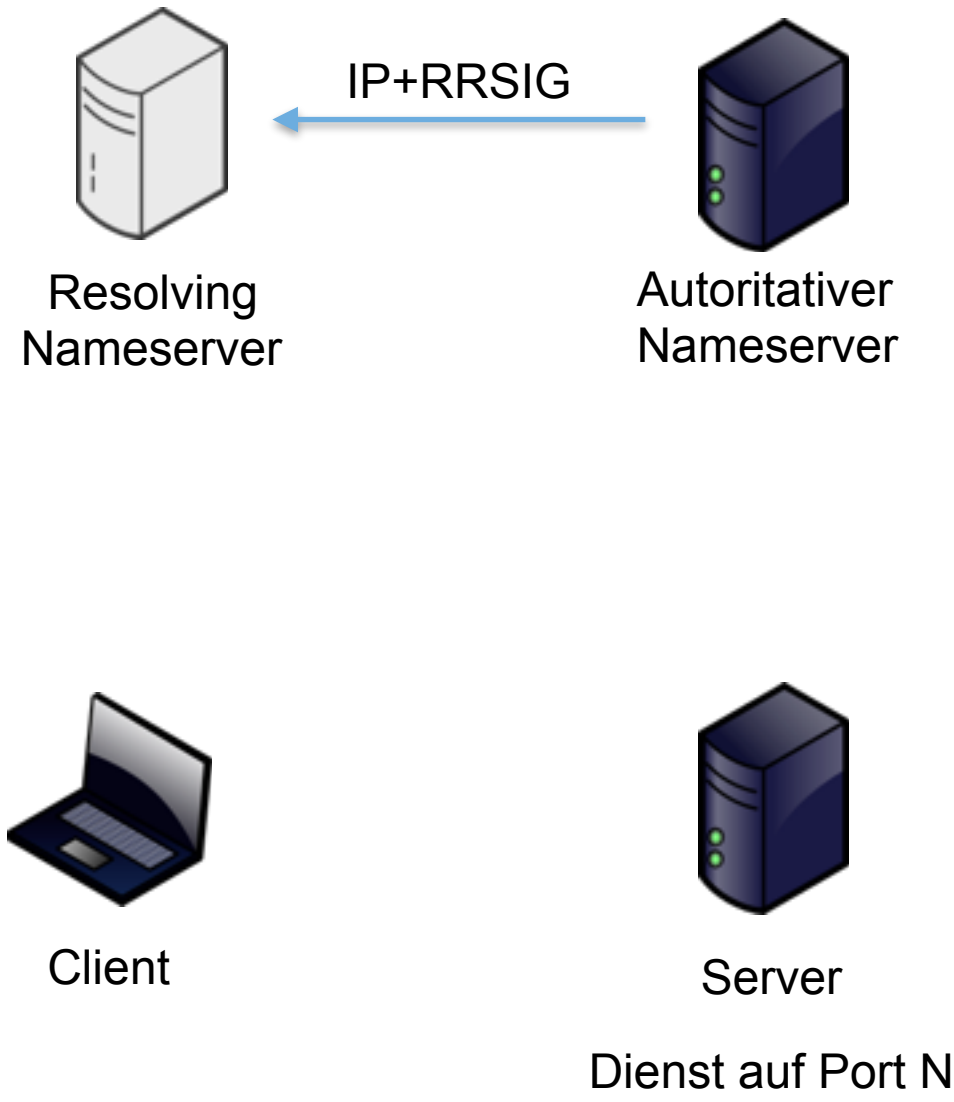
DANE - DNS-based authenticated named Entity

1. Client fragt Dienst auf Port N
   auf Server an
2. Resolver fragt Autoritativen Nameserver
   nach IP Server



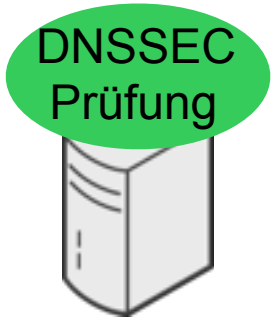
DANE - DNS-based authenticated named Entity

1. Client fragt Dienst auf Port N
   auf Server an
2. Resolver fragt Autoritativen Nameserver
   nach IP Server
3. Aut. NS sendet IP+RRSIG



DANE - DNS-based authenticated named Entity

1. Client fragt Dienst auf Port N auf Server an
2. Resolver fragt Autoritativen Nameserver nach IP Server
3. Aut. NS sendet IP+RRSIG
4. Resolving Nameserver prüft IP mit DNSSEC Hash und RRSIG für IP



Resolving Nameserver

Autoritativer Nameserver

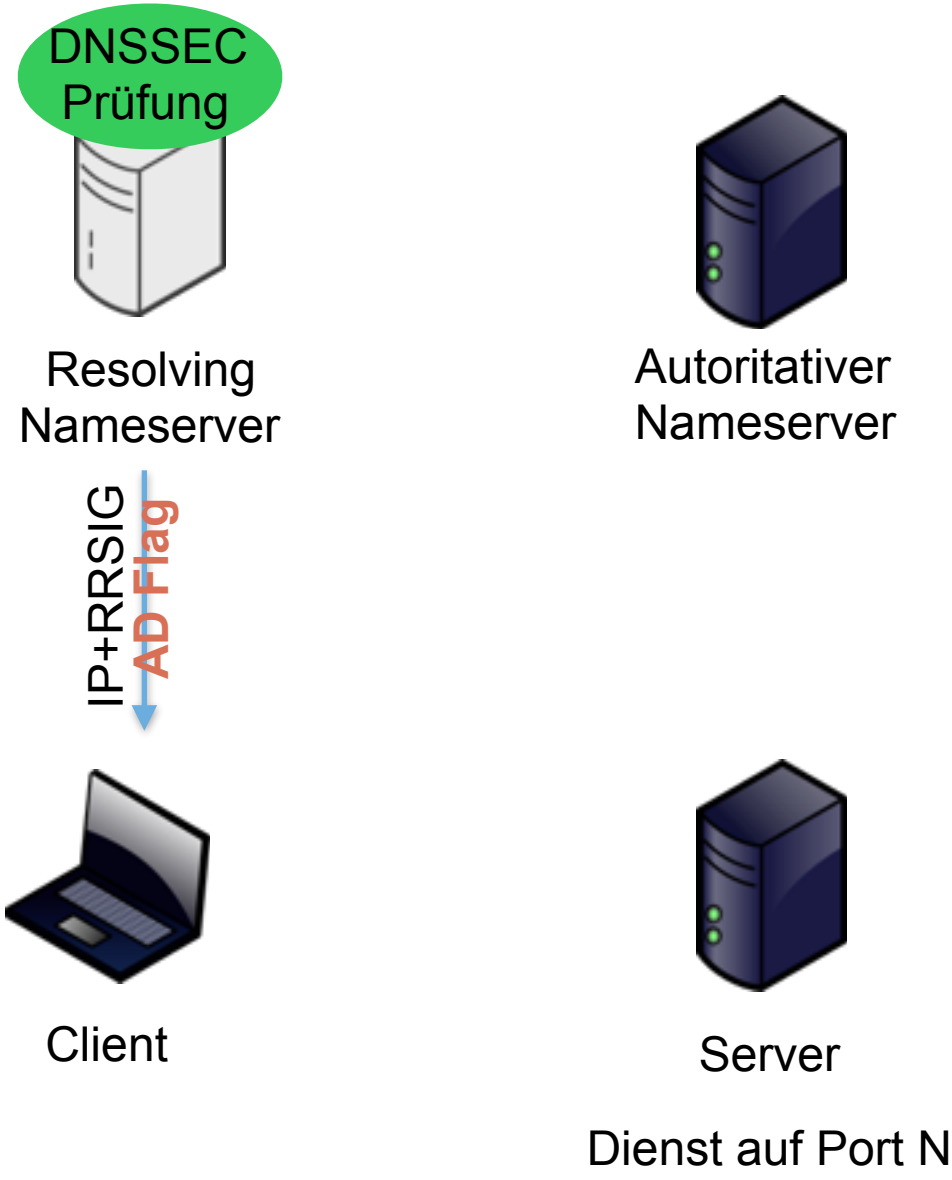
Client

Server

Dienst auf Port N

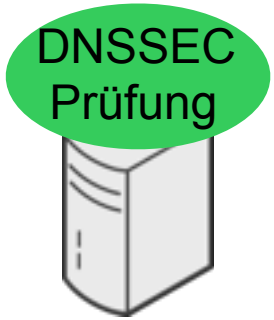
DANE - DNS-based authenticated named Entity

1. Client fragt Dienst auf Port N auf Server an
2. Resolver fragt Autoritativen Nameserver nach IP Server
3. Aut. NS sendet IP+RRSIG
4. Resolving Nameserver prüft IP mit DNSSEC Hash und RRSIG für IP
5. Resolver sendet DNSSEC-AD Antwort an Client



DANE - DNS-based authenticated named Entity

1. Client fragt Dienst auf Port N auf Server an
2. Resolver fragt Autoritativen Nameserver nach IP Server
3. Aut. NS sendet IP+RRSIG
4. Resolving Nameserver prüft IP mit DNSSEC Hash und RRSIG für IP
5. Resolver sendet DNSSEC-AD Antwort an Client
6. Client vertraut DNSSEC-AD



Resolving Nameserver

Autoritativer Nameserver

Client

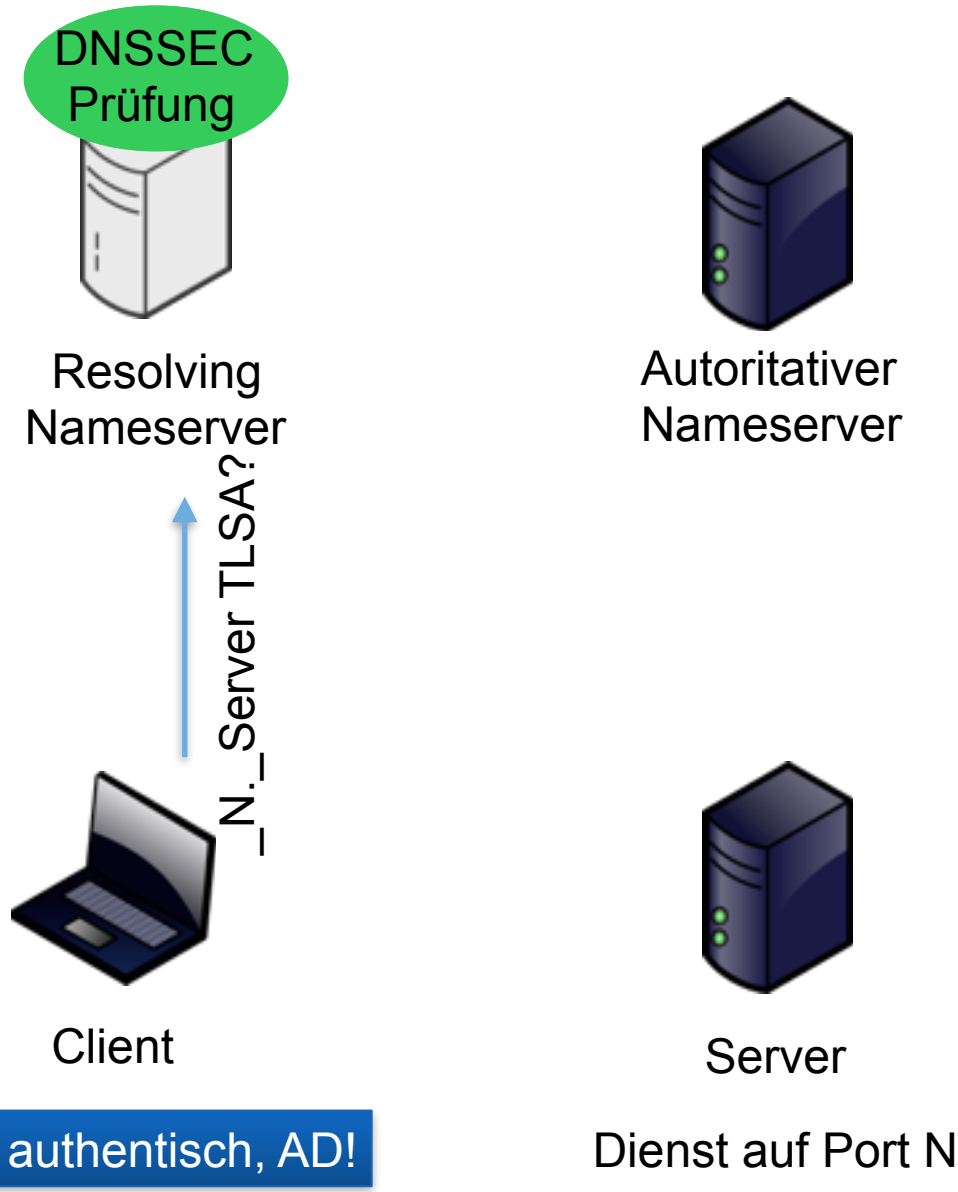
Server

IP authentisch, AD!

Dienst auf Port N

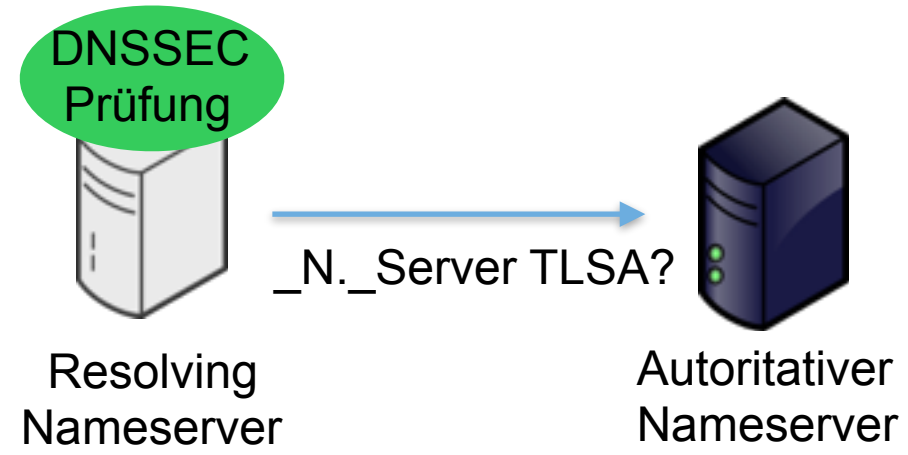
DANE - DNS-based authenticated named Entity

1. Client fragt Dienst auf Port N auf Server an
2. Resolver fragt Autoritativen Nameserver nach IP Server
3. Aut. NS sendet IP+RRSIG
4. Resolving Nameserver prüft IP mit DNSSEC Hash und RRSIG für IP
5. Resolver sendet DNSSEC-AD Antwort an Client
6. Client vertraut DNSSEC-AD
7. _N._Server TLSA (für Dienst auf Server)?



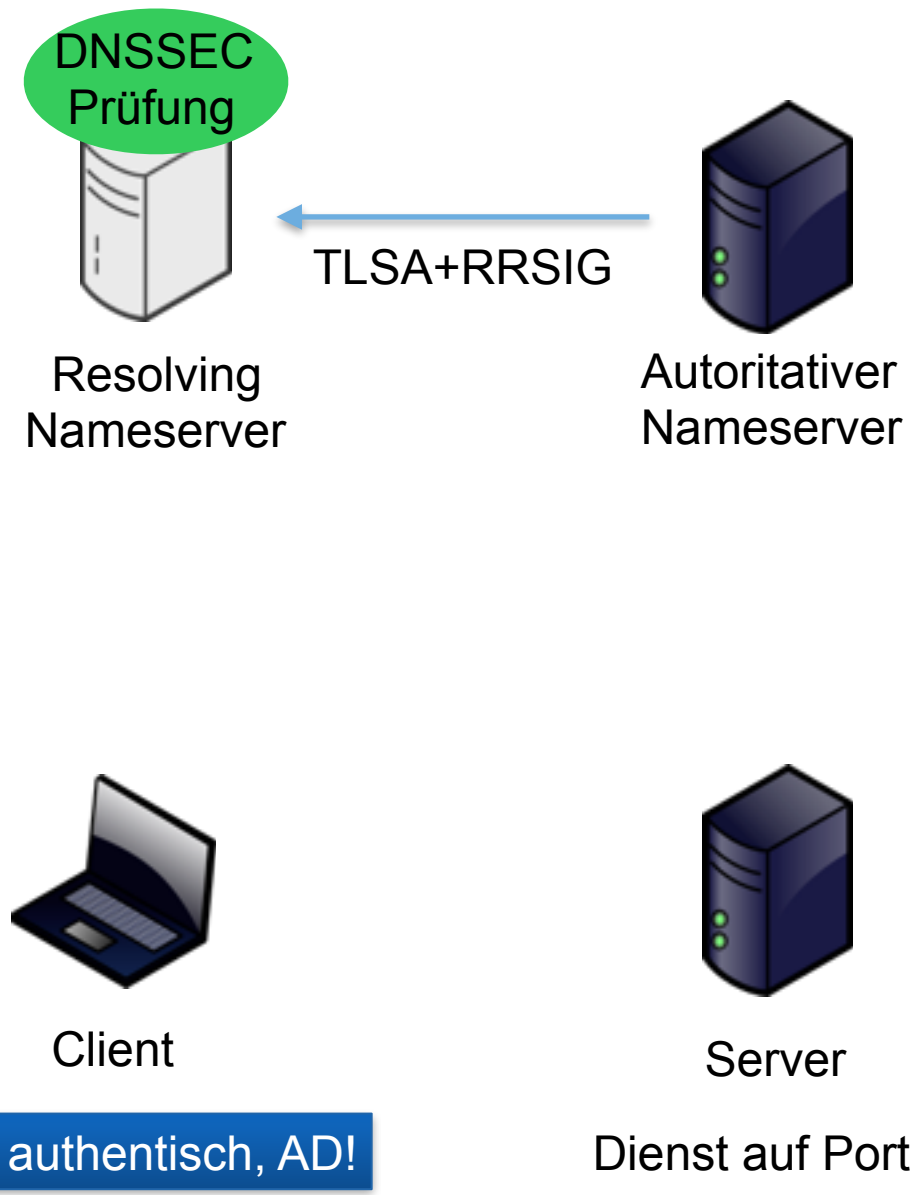
DANE - DNS-based authenticated named Entity

1. Client fragt Dienst auf Port N auf Server an
2. Resolver fragt Autoritativen Nameserver nach IP Server
3. Aut. NS sendet IP+RRSIG
4. Resolving Nameserver prüft IP mit DNSSEC Hash und RRSIG für IP
5. Resolver sendet DNSSEC-AD Antwort an Client
6. Client vertraut DNSSEC-AD
7. _N._Server TLSA (für Dienst auf Server)?
8. Resolver fragt Autoritativen NS nach _N._Server TLSA Record



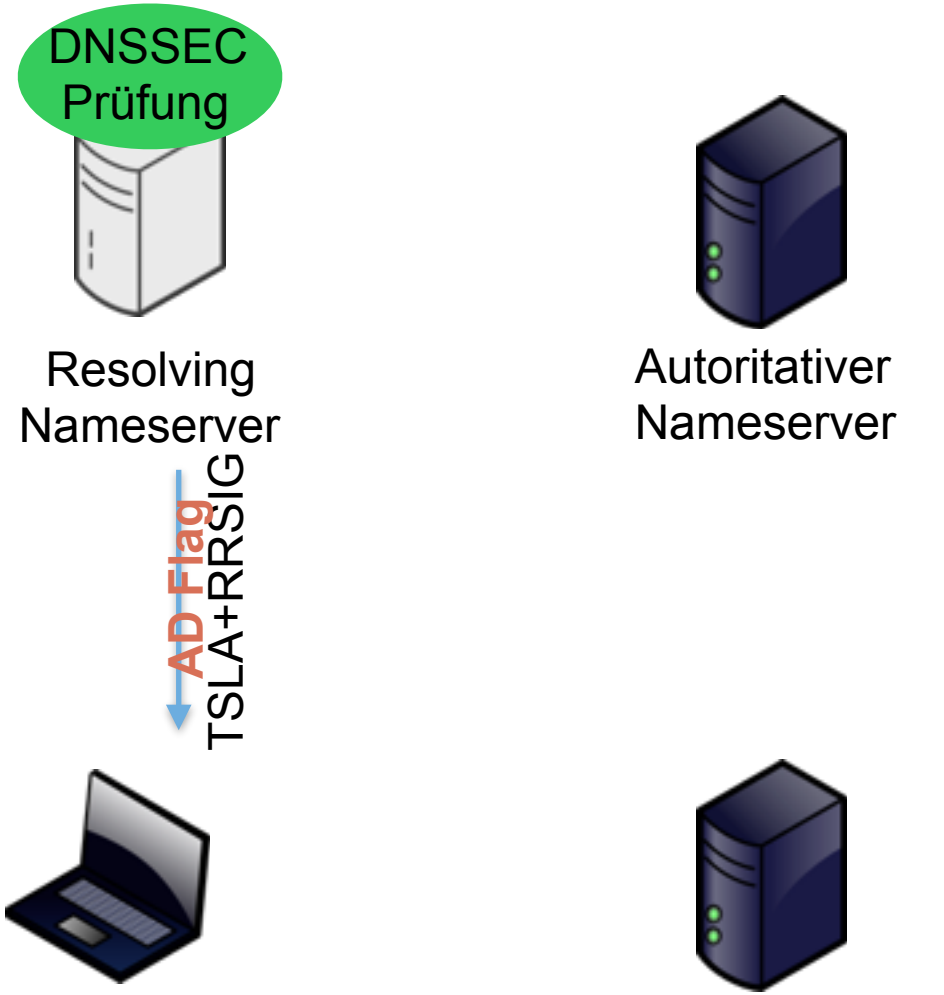
DANE - DNS-based authenticated named Entity

1. Client fragt Dienst auf Port N auf Server an
2. Resolver fragt Autoritativen Nameserver nach IP Server
3. Aut. NS sendet IP+RRSIG
4. Resolving Nameserver prüft IP mit DNSSEC Hash und RRSIG für IP
5. Resolver sendet DNSSEC-AD Antwort an Client
6. Client vertraut DNSSEC-AD
7. _N._Server TLSA (für Dienst auf Server)?
8. Resolver fragt Autoritativen NS nach _N._Server TLSA Record
9. Aut. NS sendet TSLA+RRSIG-Eintrag für Port N



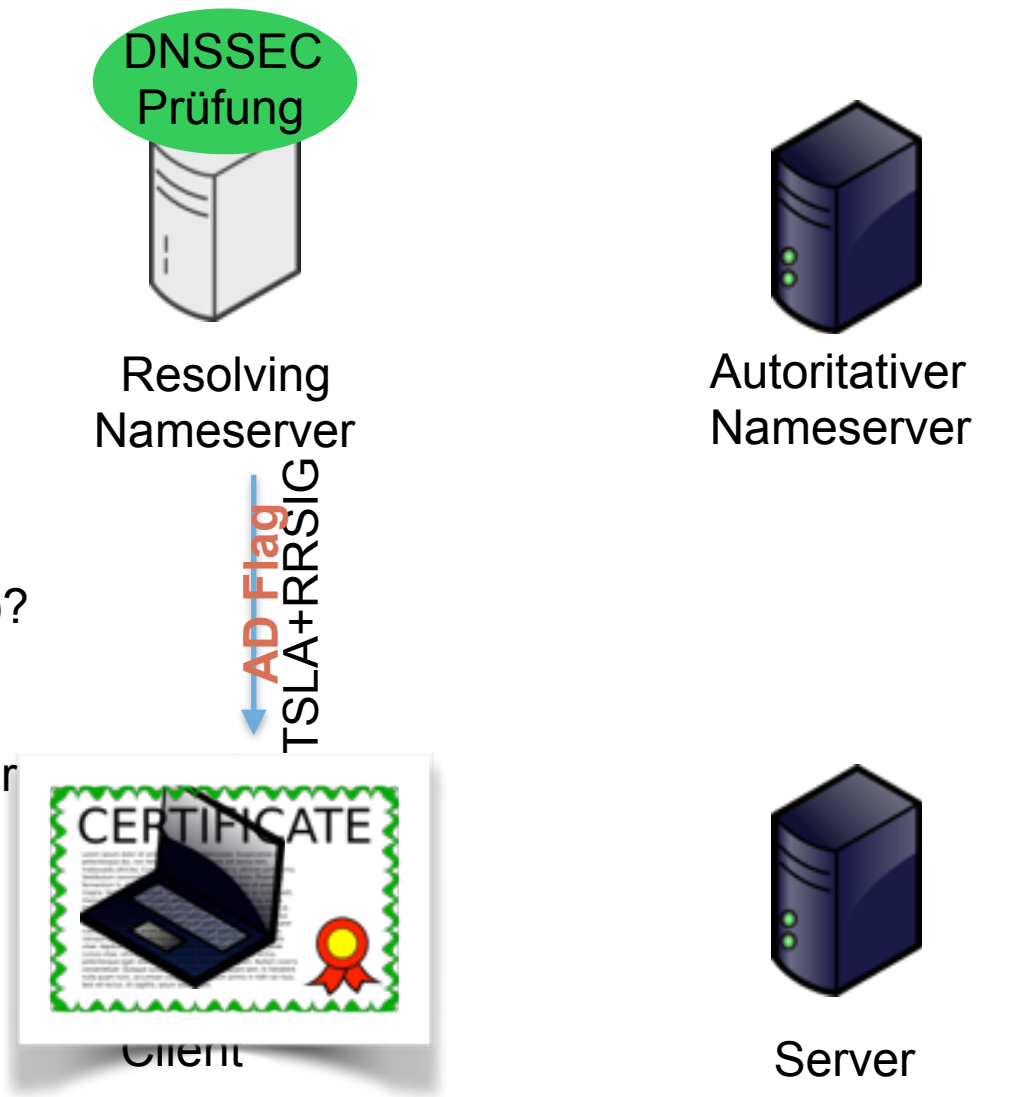
DANE - DNS-based authenticated named Entity

1. Client fragt Dienst auf Port N auf Server an
2. Resolver fragt Autoritativen Nameserver nach IP Server
3. Aut. NS sendet IP+RRSIG
4. Resolving Nameserver prüft IP mit DNSSEC Hash und RRSIG für IP
5. Resolver sendet DNSSEC-AD Antwort an Client
6. Client vertraut DNSSEC-AD
7. _N._Server TLSA (für Dienst auf Server)?
8. Resolver fragt Autoritativen NS nach _N._Server TLSA Record
9. Aut. NS sendet TSLA+RRSIG-Eintrag für Port N
10. Resolver sendet TLSA-Signatur an Client



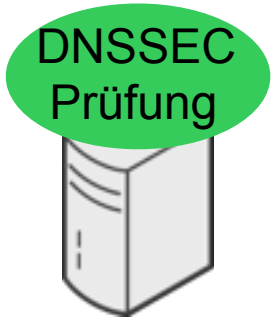
DANE - DNS-based authenticated named Entity

1. Client fragt Dienst auf Port N auf Server an
2. Resolver fragt Autoritativen Nameserver nach IP Server
3. Aut. NS sendet IP+RRSIG
4. Resolving Nameserver prüft IP mit DNSSEC Hash und RRSIG für IP
5. Resolver sendet DNSSEC-AD Antwort an Client
6. Client vertraut DNSSEC-AD
7. _N._Server TLSA (für Dienst auf Server)?
8. Resolver fragt Autoritativen NS nach _N._Server TLSA Record
9. Aut. NS sendet TSLA+RRSIG-Eintrag für Port N
10. Resolver sendet TLSA-Signatur an Client
11. Zertifikat empfangen



DANE - DNS-based authenticated named Entity

1. Client fragt Dienst auf Port N auf Server an
2. Resolver fragt Autoritativen Nameserver nach IP Server
3. Aut. NS sendet IP+RRSIG
4. Resolving Nameserver prüft IP mit DNSSEC Hash und RRSIG für IP
5. Resolver sendet DNSSEC-AD Antwort an Client
6. Client vertraut DNSSEC-AD
7. _N._Server TLSA (für Dienst auf Server)?
8. Resolver fragt Autoritativen NS nach _N._Server TLSA Record
9. Aut. NS sendet TSLA+RRSIG-Eintrag für Port N
10. Resolver sendet TLSA-Signatur an Client
11. Zertifikat empfangen
12. Client überprüft TSLA-Signatur anhand Public Key-Hash



Resolving Nameserver

Autoritativer Nameserver

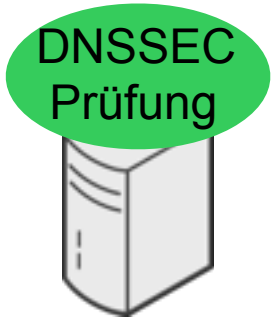
Client

Server

Dienst auf Port N

DANE - DNS-based authenticated named Entity

1. Client fragt Dienst auf Port N auf Server an
2. Resolver fragt Autoritativen Nameserver nach IP Server
3. Aut. NS sendet IP+RRSIG
4. Resolving Nameserver prüft IP mit DNSSEC Hash und RRSIG für IP
5. Resolver sendet DNSSEC-AD Antwort an Client
6. Client vertraut DNSSEC-AD
7. _N._Server TLSA (für Dienst auf Server)?
8. Resolver fragt Autoritativen NS nach _N._Server TLSA Record
9. Aut. NS sendet TSLA+RRSIG-Eintrag für Port N
10. Resolver sendet TLSA-Signatur an Client
11. Zertifikat empfangen
12. Client überprüft TSLA-Signatur anhand Public Key-Hash
13. Zertifikat gepinnt



Resolving Nameserver

Autoritativer Nameserver

Client

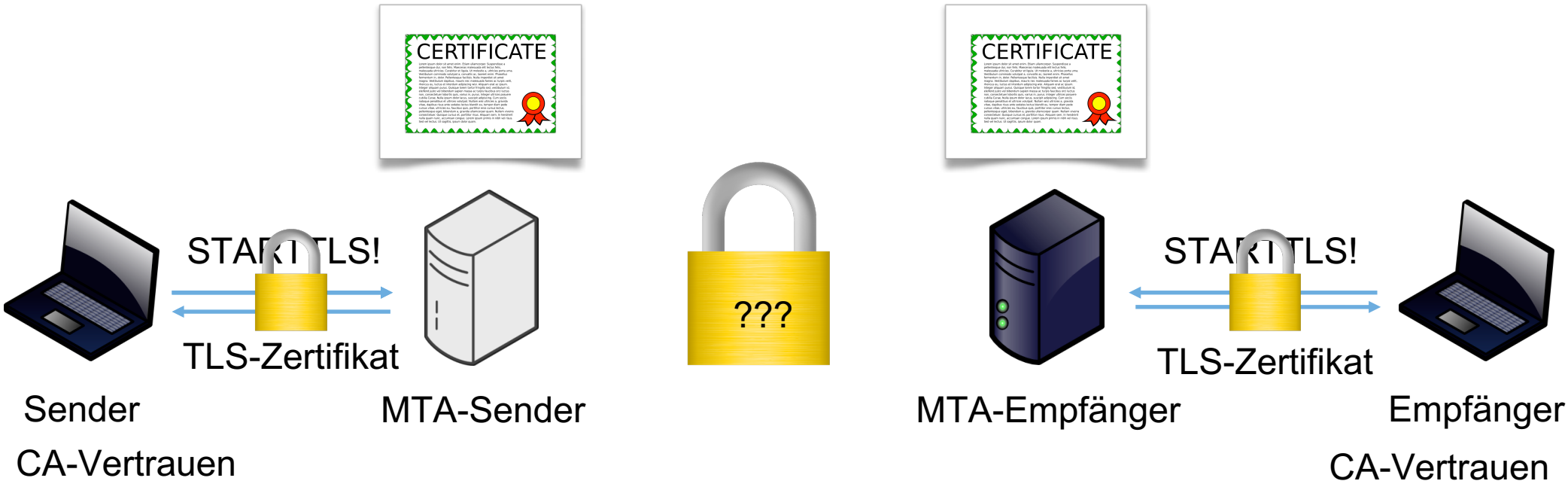
Server

Dienst auf Port N

DANE - Beispiel Absicherung eines Emailservers

TLS-Verschlüsselte SMTP Verbindung

- Mailtransport via SMTP, TLS-verschlüsselt, Zertifikat-Vertrauen



TLS-Verschlüsselte SMTP Verbindung

- STARTTLS-**Erweiterung** für SMTP wurde erst 2002 spezifiziert,
- 20 Jahre nach RFC 821
  - signalisiert durch STARTTLS-Keyword in EHLO (unauthentifizierte Plaintext-Session!)
  - Triviale Downgrade-Attacke auf unverschlüsselte Verbindung
  - Opportunistische Verschlüsselung ohne Zertifikatsprüfung
    - 70% gültiges Zertifikat, 20% selbstsigniert, 10% kein SSL
- Schützt lediglich gegen passive Lauscher, aber nicht gegen aktive Angriffe (MitM)

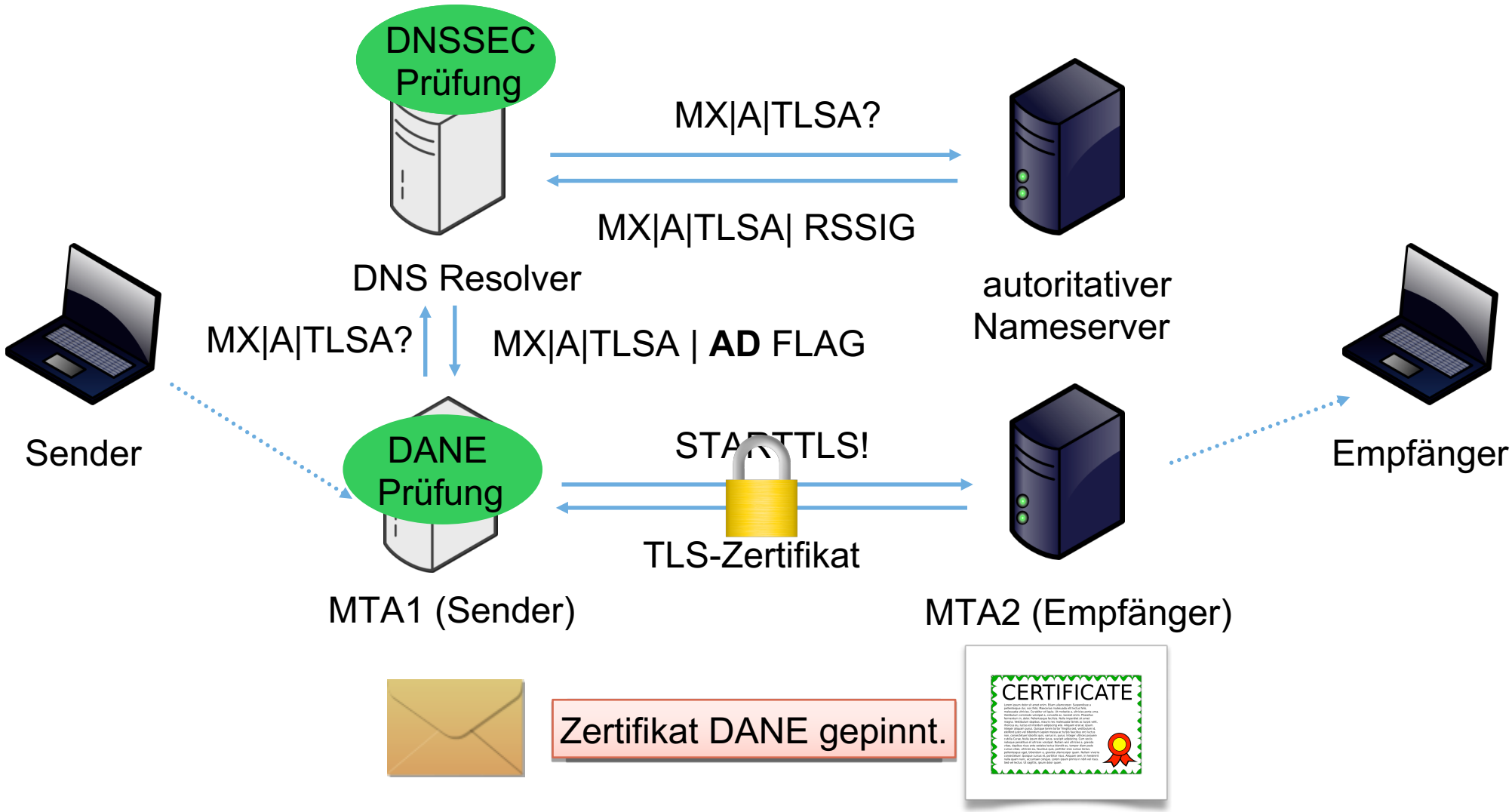
TLS-Verschlüsselte SMTP Verbindung mit DANE

- DANE für SMTP (RFC 7672) erlaubt dem Empfänger über DNSSEC die sichere Signalisierung von
  - „ich spreche STARTTLS“
  - „ich habe ein Zertifikat mit bestimmten Eigenschaften“
- DANE-fähiger Sender kann diese Hinweise beachten und SMTP-Session bei einem Fehler terminieren

TLS-Verschlüsselte SMTP Verbindung mit DANE - Ablauf

- Sendender MTA frägt nach MX-Record der Empfängerdomain
  - erhält Liste der Mailserver und (bei DNSSEC) AD-Flag
- Sendender MTA wählt einen Mailserver aus, frägt A/AAAA an
- Sendender MTA stellt Frage nach _25._tcp.mailserver
  - erhält TLSA-Records und AD-Flag
- Sendender MTA pinnt diese Zertifikate für SSL-Session und baut eine verifizierte Verbindung auf

TLS-Verschlüsselte SMTP Verbindung mit DA
Ablauf



TLSA Ressource Record im Detail (RFC6698)

TLSA pinning kann verschieden interpretiert werden

_25._tcp.<servername>. IN TLSA 3 0 1 8cb0fc6c527506a053f4f1…

TLSA Ressource Record im Detail (RFC6698)

TLSA pinning kann verschieden interpretiert werden

_25._tcp.<servername>. IN TLSA 3 0 1 8cb0fc6c527506a053f4f1…

Port

TLSA Ressource Record im Detail (RFC6698)

TLSA pinning kann verschieden interpretiert werden

_25._tcp.<servername>. IN TLSA 3 0 1 8cb0fc6c527506a053f4f1…

Protokoll

TLSA Ressource Record im Detail (RFC6698)

TLSA pinning kann verschieden interpretiert werden

_25._tcp.<servername>. IN TLSA 3 0 1 8cb0fc6c527506a053f4f1…

Name des Mailservers

TLSA Ressource Record im Detail (RFC6698)

TLSA pinning kann verschieden interpretiert werden

_25._tcp.<servername>. IN TLSA 3 0 1 8cb0fc6c527506a053f4f1…

Typ TLSA

TLSA Ressource Record im Detail (RFC6698)

TLSA pinning kann verschieden interpretiert werden

_25._tcp.<servername>. IN TLSA 3 0 1 8cb0fc6c527506a053f4f1…

Zertifikat Nutzung

- → 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!)

TLSA Ressource Record im Detail (RFC6698)

TLSA pinning kann verschieden interpretiert werden

_25._tcp.<servername>. IN TLSA 3 0 1 8cb0fc6c527506a053f4f1…

Zertifikat Nutzung

→ 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!)

0 und 1 sollten nicht benutzt werden, dieselben Probleme wie Zertifikate!

TLSA Ressource Record im Detail (RFC6698)

TLSA pinning kann verschieden interpretiert werden

_25._tcp.<servername>. IN TLSA 3 0 1 8cb0fc6c527506a053f4f1…

Selector

→ 0 = Full Certificate
→ 1 = SubjectPublicKeyInfo

TLSA Ressource Record im Detail (RFC6698)

TLSA pinning kann verschieden interpretiert werden

_25._tcp.<servername>. IN TLSA 3 0 1 8cb0fc6c527506a053f4f1…

Matching

→ 0 = Keinen Hash verwenden
→ 1 = SHA-256
→ 2 = SHA-512

TLSA Ressource Record im Detail (RFC6698)

TLSA pinning kann verschieden interpretiert werden

_25._tcp.<servername>. IN TLSA 3 0 1 8cb0fc6c527506a053f4f1…

Signatur

TLSA Ressource Record im Detail (RFC6698)

TLSA pinning kann verschieden interpretiert werden

_25._tcp.<servername>. IN TLSA 3 0 1 8cb0fc6c527506a053f4f1…

DANE ausgehend

Outbound DANE benötigt Support in Software

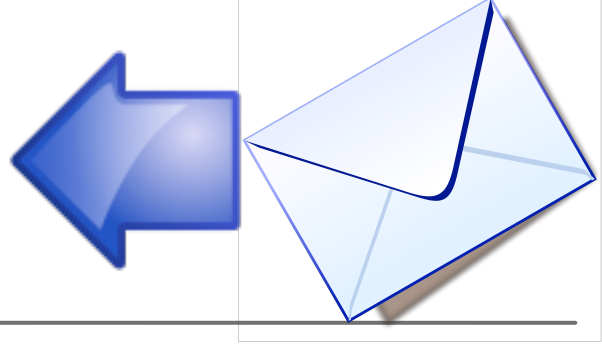
Die folgenden MTAs unterstützen ausgehend DANE

- Postfix Version >= 2.11 mit OpenSSL 1.0.0+
- Exim 4.85 (EXPERIMENTAL_DANE)
- Halon > 3.4-rocky-r2 (kommerziell)
- sendmail unterstützt DANE nicht ausgehend
- Microsoft Exchange Server nicht ausgehend

Statistik am LRZ (31.10. - 4.11.):

- 17% der ausgehenden TLS-Verbindungen mit DANE gesichert
- hauptsächlich GMX/Web.de, bund.de, bayern.de, uni-kl.de, fau.de, posteo.de, mailbox.org, ...

DANE eingehend



- kein Support der lokalen MTA nötig, STARTTLS genügt
- Zonen DNSSEC-signieren und Zertifikat im TLSA-Record hinterlegen
- keine Signalisierung der TLSA-Nutzung durch den Sender

  → keine Statistik

Erstellen eines TLSA-Records

x509 Zertifikat wie gewohnt erstellen, dann kann man den TLSA erstellen:

$ openssl x509 -in server.crt -outform DER | openssl sha256

(stdin)=8cb0fc6c527506a053f4f14c8464bebbd6dede2738d11468dd953d7d6a3021f1

Eintrag in der DNSSEC-Zone:

_port._protokoll.<servername>. IN TLSA 3 0 1 \

8cb0fc6c527506a053f4f14c8464bebbd6dede2738d11468dd953d7d6a3021f1

DANE-Voraussetzungen SMTP MTAs

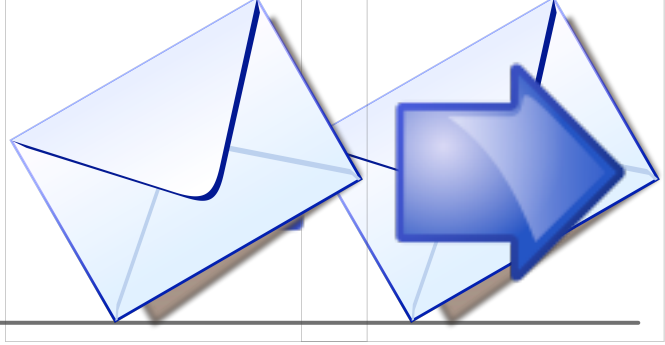
- Postfix Version > 2.11

  /etc/postfix.main.cf
  smtp_dns_support_level = dnssec
  smtp_tls_security_level = dane
  smtp_tls_loglevel = 1

- Exim 4.85 (EXPERIMENTAL_DANE)
- Halon > 3.4-rocky-r2[(kommerziell)
- sendmail unterstützt DANE nicht
- Microsoft Exchange Server (noch?) nicht

Konfiguration in Postfix 3.1

Outbound Mailserver



- DNSSEC-validierender Resolver, dem vertraut wird
-
- Postfix >= 2.11, gebaut gegen OpenSSL 1.0.0+
  - smtp_tls_security_level = dane
  - smtp_dns_support_level = dnssec
  - smtp_tls_loglevel = 1

Postfix log - Überprüfen der DANE-Verbindung

Postfix loggt verschiedene Sicherheitslevel für SMTP-Verbindungen

- Anonymous
  - Anonymer Cipher, kein Zertifikat übertragen (i.A. Postfix vs. Postfix)
- Untrusted
  - Zertifikat nicht verifizierbar, z.B. self-signed
- Trusted
  - Zertifikat von einer vertrauenswürdigen CA
- Verified
  - Zertifikat matcht DANE oder manuell konfigurierten
  - Trust-Anchor

Verified TLSA-Eintrag gefunden

Postfix log:

postfix/smtp: Verified TLS connection established to dane.lhns.org[2001:db8:b51d:e5

:510::2]:25: TLSv1.2 with cipher ECDHE-RSA-AES256-GCM-SHA384 (256/256 bits)

postfix/smtp: DD8748072B: to=<danetest@lhns.org>, relay=dane.lhns.org[2001:db8:b51d:e5

:510::2]:25, delay=0.55, delays=0.09/0.02/0.25/0.1, dsn=2.0.0, status=sent (250

2.0.0 Ok: queued as 84A6B3FD38)

Verifizierte SMTP-Verbindung, Zertifikat entspricht entweder TLSA-Record oder manuell konfiguriertem Pinning

Message wird versendet

Invalid TLSA-Eintrag - Opportunistic TLS

Postfix log:

postfix/smtp: Trusted TLS connection established to dane.lhns.org[2001:db8:b51d:e5

:510::2]:25: TLSv1.2 with cipher ECDHE-RSA-AES256-GCM-SHA384 (256/256 bits)

postfix/smtp: 730D78072B: to=<danetest@lhns.org>, relay=dane.lhns.org[2001:db8:b51d:e5

:510::2]:25, delay=0.21, delays=0.09/0/0.12/0, dsn=4.7.5, status=deferred (Server certificate not verified)

Dem Zertifikat würde nach normalen „chain-of-trust“ vertraut

Aber das Zertifikat entspricht nicht dem (TLSA)-Pinning.

E-Mail deferred, möglicherweise MitM-Attacke oder Fehlkonfiguration

Opportunistic TLS - Kein TLSA-Eintrag

Postfix log:

postfix/smtp: Trusted TLS connection established to dane.lhns.org[2001:db8:b51d:e5:510::2]:25: TLSv1.2 with cipher ECDHE-RSA-AES256-GCM-SHA384 (256/256 bits)

postfix/smtp: DD8748072B: to=<danetest@lhns.org>, relay=dane.lhns.org[2001:db8:b51d:e5:510::2]:25, delay=0.55, delays=0.09/0.02/0.25/0.1, dsn=2.0.0, status=sent (250 2.0.0 Ok: queued as 84A6B3FD38)

Es wird kein TLSA-Eintrag gefunden, und SMTP fällt auf “opportunistic TLS” zurück, Message sent

Kein TLSA-Eintrag - TLS verpflichtend

Postfix log:

postfix/smtp: warning: TLS policy lookup for lhns.org/dane.lhns.org: no TLSA records found

postfix/smtp: E9BCC8072B: to=<danetest@lhns.org>, relay=none, delay=0.15, delays=0.13/0.02/0/0, dsn=4.7.5, status=deferred (no TLSA records found)

Es wird kein TLSA-Eintrag gefunden, Postfix verschiebt das Versenden (“deferred”)

Postfix TLS Security Levels

- In /etc/postfix/main.cf wird der globale Security-Level gesetzt (smtp_tls_security_level)
  - none — kein TLS
  - may — opportunistisches TLS
  - dane — opportunistisches TLS + DANE
  - dane-only — DANE verpflichtend
  - verify/secure — verifiziertes TLS verpflichtend
- Kann über smtp_tls_policy_maps pro Domain überschrieben werden, bspw. um fehlerhafte TLSA-Einträge zu übergehen oder bekannte Ziele auch ohne DANE zu pinnen

Übung - TLSA mit Postfix

SMTP Email-Server - TLS Schritt für Schritt

1. Zertifikat erstellen, selbst oder durch CA signieren
   hier bereits erledigt

2. TLSA RR erstellen

3. TLSA Signatur für MX in der Zone eintragen

4. posttls-finger

5. Postfix konfigurieren

TLSA-Eintrag erstellen

Workshop-VMs haben bereits ein selbstsigniertes Zertifikat

/etc/ssl/certs/ssl-cert-snakeoil.pem (Zertifikat)
/etc/ssl/private/ssl-cert-snakeoil.key (privater Schlüssel)

```
ws01.ws.dnssec.bayern.  300     IN      MX      100 dnssec-ws01.ws01.ws.dnssec.bayern.
```

In die Zone-Datei eintragen und überprüfen:

```
_25._tcp.dnssec-ws01 IN TLSA 3 0 1 \
        a52e5c88044a0a65c92efd133ed90919dd9a11812eeb2d2da20ee7612686aa6f
```

```
dig -t tlsa _25._tcp.dnssec-ws01.ws01.ws.dnssec.bayern
```

TLSA-Eintrag erstellen

- Erstellung mit OpenSSL nur manchmal intuitiv (für 3 0 1):

```
openssl x509 -in /etc/ssl/certs/ssl-cert-snakeoil.pem -outform DER | openssl sha256
(stdin) = a52e5c88044a0a65c92efd133ed90919dd9a11812eeb2d2da20ee7612686aa6f
```

- Online-Dienste für die Erstellung von beliebigen TLSA-Records
  - https://www.huque.com/bin/gen_tlsa
  - https://de.ssl-tools.net/tlsa-generator
- CLI-Tools, z.B. /usr/bin/tlsa aus hash-slinger oder ldns-dane
  - tlsa --create --port 25 --certificate /etc/ssl/certs/ssl-cert-snakeoil.pem --selector 1 ws01.ws.dnssec.bayern
  - _25._tcp.ws01.ws.dnssec.bayern. IN TLSA 3 1 1 850e901d6c0989d4421f13953653c601e70fe69afdaa513274965f0f6e61471d
  - ldns-dane create -c /etc/ssl/certs/ssl-cert-snakeoil.pem dnssec-ws01.ws01.ws.dnssec.bayern 25 -n

TLSA-Eintrag prüfen

posttls-finger ws01.ws.dnssec.bayern

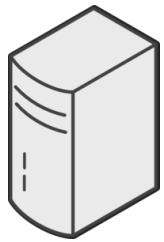
Ohne TLSA:

posttls-finger: Connected to dnssec-ws01.ws01.ws.dnssec.bayern[2001:4ca0:800:2:250:56ff:fe8f:5584]:25
posttls-finger: dnssec-ws01.ws01.ws.dnssec.bayern[2001:4ca0:800:2:250:56ff:fe8f:5584]:25 CommonName debian.srv.lrz.de
posttls-finger: certificate verification failed for dnssec-ws01.ws01.ws.dnssec.bayern[2001:4ca0:800:2:250:56ff:fe8f:5584]:25:
self-signed certificate
posttls-finger: dnssec-ws01.ws01.ws.dnssec.bayern[2001:4ca0:800:2:250:56ff:fe8f:5584]:25: subject_CN=debian.srv.lrz.de,
issuer_CN=debian.srv.lrz.de,
fingerprint=82:DA:1B:BB:48:07:D1:CD:6C:22:63:32:77:27:8C:24:2F:11:53:F2,
pkey_fingerprint=39:CD:0F:39:3C:11:90:60:7C:1F:99:29:7F:24:42:2F:D3:CE:5C:54
posttls-finger: Untrusted TLS connection established to dnssec-ws01.ws01.ws.dnssec.bayern[2001:4ca0:800:2:250:56ff:fe8f:5584]:25:
TLSv1.2 with cipher ECDHE-RSA-AES256-GCM-SHA384 (256/256 bits)

Mit TLSA:

posttls-finger: using DANE RR: _25._tcp.dnssec-ws01.ws01.ws.dnssec.bayern IN TLSA 3 0 1
A5:2E:5C:88:04:4A:0A:65:C9:2E:FD:13:3E:D9:09:19:DD:9A:11:81:2E:EB:2D:2D:A2:0E:E7:61:26:86:AA:6F
posttls-finger: Connected to dnssec-ws01.ws01.ws.dnssec.bayern[2001:4ca0:800:2:250:56ff:fe8f:5584]:25
posttls-finger: dnssec-ws01.ws01.ws.dnssec.bayern[2001:4ca0:800:2:250:56ff:fe8f:5584]:25: depth=0 matched
end entity certificate sha256 digest A5:2E:5C:88:04:4A:0A:65:C9:2E:FD:13:3E:D9:09:19:DD:9A:11:81:2E:EB:2D:2D:A2:0E:E7:61:26:86
posttls-finger: dnssec-ws01.ws01.ws.dnssec.bayern[2001:4ca0:800:2:250:56ff:fe8f:5584]:25 CommonName debian.srv.lrz.de
posttls-finger: dnssec-ws01.ws01.ws.dnssec.bayern[2001:4ca0:800:2:250:56ff:fe8f:5584]:25: subject_CN=debian.srv.lrz.de,
issuer_CN=debian.srv.lrz.de, fingerprint=82:DA:1B:BB:48:07:D1:CD:6C:22:63:32:77:27:8C:24:2F:11:53:F2,
pkey_fingerprint=39:CD:0F:39:3C:11:90:60:7C:1F:99:29:7F:24:42:2F:D3:CE:5C:54
posttls-finger: Verified TLS connection established to dnssec-ws01.ws01.ws.dnssec.bayern[2001:4ca0:800:2:250:56ff:fe8f:5584]:25:
TLSv1.2 with cipher ECDHE-RSA-AES256-GCM-SHA384 (256/256 bits)

Outbound Mailserver



- DNSSEC-validierender Resolver, dem vertraut wird

- Postfix >= 2.11
  - smtp_tls_security_level = dane
  - smtp_dns_support_level = dnssec

- In /etc/postfix/main.cf ändern, postfix reload eingeben
- Testmail an DANE-fähigen Server verschicken
  - echo Test | sendmail test@ws01.ws.dnssec.bayern
- Logdatei /var/log/mail.log ansehen

DNSSEC/DANE - Unterstützung in Clients

Webbrowsers

- Kein Browser unterstützt „out-of-the-box“ DNSSEC/DANE
- Internet Explorer
  https://labs.nic.cz/en/dnssec-ie.html (pre-release)
- Javascript-Plugin kann DNSSEC-Verifikation nachrüsten
  - Google Chrome
    http://www.internetsociety.org/deploy360/resources/how-to-add-dnssec-support-to-google-chrome/
  - Mozilla Firefox
    http://www.internetsociety.org/deploy360/resources/how-to-add-dnssec-support-to-mozilla-firefox/

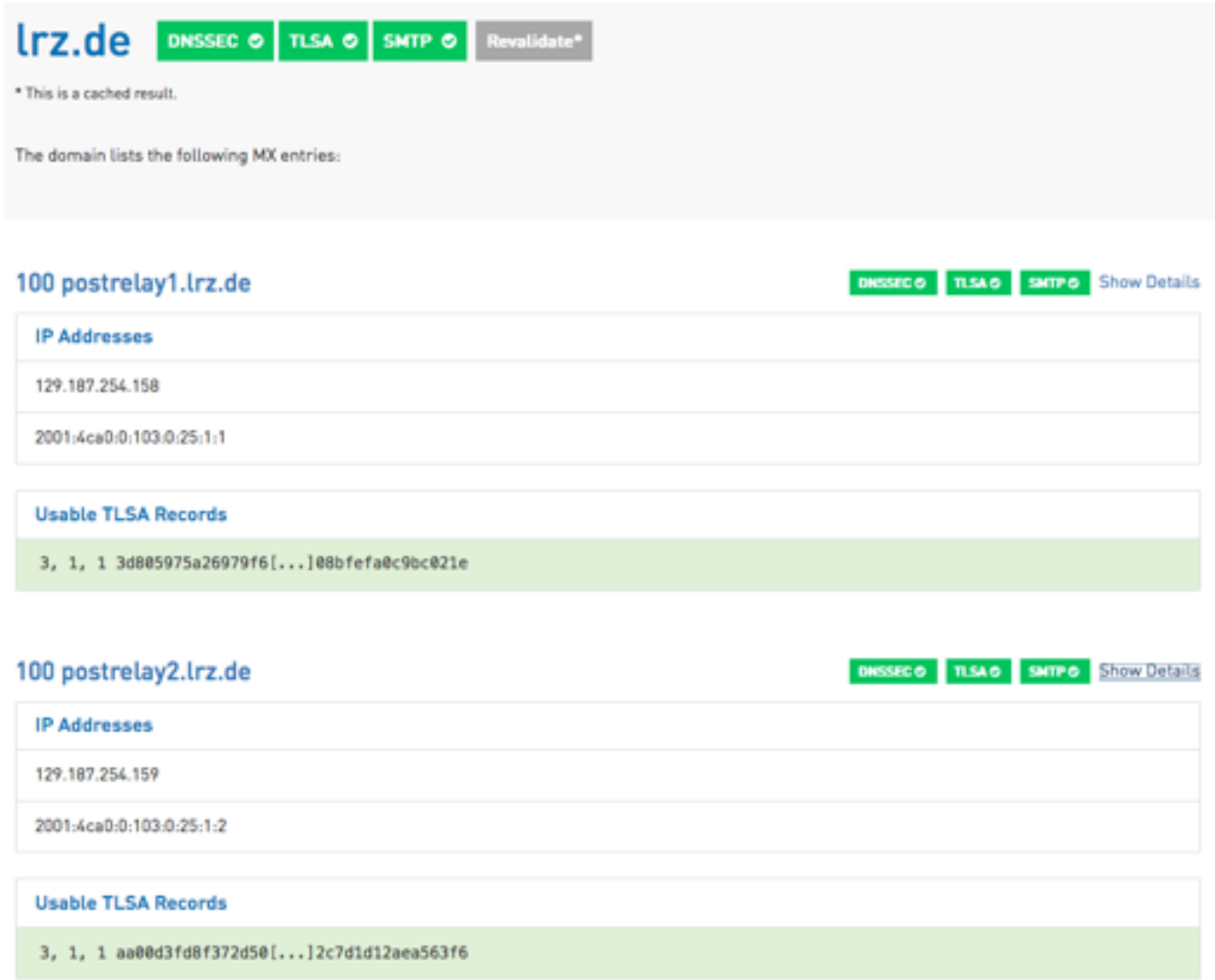
Email-Clients

- Bisher nur sehr geringe Unterstützung
- Keine „out-of-the-box“-Lösung
- Zusatztools und Plugins erlauben TLSA Verification
- Thunderbird-Startskript mit DANE-Test

  (https://www.privacy-handbuch.de/download/thunderbird_mit_danetest.sh)

DANE SMTP Validator - https://dane.sys4.de/

Online Tool, um TLSA der Mailserver eine Domain zu überprüfen:



SSH Host Key Verification

DNSSEC kann auch dazu verwendet werden, SSH Host Keys zu authentifizieren.

- SSH Host key wird in der DNS Zone gespeichert
- SSHFP record
- /etc/ssh/ssh_config oder ~/.ssh/config

  VerifyHostKeyDNS yes

- ssh -o "VerifyHostKeyDNS=yes" www.kernel-error.de (fallweise)
- DNSSEC SSHFP Überprüfung als zwingende Voraussetzung
  /etc/ssh/ssh_config

  *Host **
  StrictHostKeyChecking yes

SSHFP RR in der Zone


```
$ sshfp -s dnssec-ws01.dnssec.bayern
dnssec-ws01.dnssec.bayern     IN  SSHFP  1 1  57D79129FA85BCC0D9E15CE25C96E639E2DB3316
dnssec-ws01.dnssec.bayern     IN  SSHFP  2 1   CF239AE438FF149185378D9735BE42B519416D0F
```

SSHFP RR in der Zone


```
$ sshfp -s dnssec-ws01.dnssec.bayern
dnssec-ws01.dnssec.bayern     IN  SSHFP  1 1  57D79129FA85BCC0D9E15CE25C96E639E2DB3316
dnssec-ws01.dnssec.bayern     IN  SSHFP  2 1   CF239AE438FF149185378D9735BE42B519416D0F
```

Zielhost

SSHFP RR in der Zone


```
$ sshfp -s dnssec-ws01.dnssec.bayern
dnssec-ws01.dnssec.bayern     IN  SSHFP  1 1  57D79129FA85BCC0D9E15CE25C96E639E2DB3316
dnssec-ws01.dnssec.bayern     IN  SSHFP  2 1   CF239AE438FF149185378D9735BE42B519416D0F
```

Protrokollart

SSHFP RR in der Zone


```
$ sshfp -s dnssec-ws01.dnssec.bayern
dnssec-ws01.dnssec.bayern      IN  SSHFP  1 1  57D79129FA85BCC0D9E15CE25C96E639E2DB3316
dnssec-ws01.dnssec.bayern      IN  SSHFP  2 1  CF239AE438FF149185378D9735BE42B519416D0F
```

RR-Typ

SSHFP RR in der Zone


```
$ sshfp -s dnssec-ws01.dnssec.bayern
dnssec-ws01.dnssec.bayern     IN  SSHFP  1 1  57D79129FA85BCC0D9E15CE25C96E639E2DB3316
dnssec-ws01.dnssec.bayern     IN  SSHFP  2 1  CF239AE438FF149185378D9735BE42B519416D0F
```

Host-Key Algorithmus

SSHFP RR in der Zone


```
$ sshfp -s dnssec-ws01.dnssec.bayern
dnssec-ws01.dnssec.bayern     IN  SSHFP  1 1  57D79129FA85BCC0D9E15CE25C96E639E2DB3316
dnssec-ws01.dnssec.bayern     IN  SSHFP  2 1  CF239AE438FF149185378D9735BE42B519416D0F
```

Hash-Art des FP

SSHFP RR in der Zone


```
$ sshfp -s dnssec-ws01.dnssec.bayern
dnssec-ws01.dnssec.bayern     IN  SSHFP  1 1  57D79129FA85BCC0D9E15CE25C96E639F2DB3316
dnssec-ws01.dnssec.bayern     IN  SSHFP  2 1  CF239AE438FF149185378D9735BE42B519416D0F
```

Fingerprint

SSHFP RR in der Zone


```
$ sshfp -s dnssec-ws01.dnssec.bayern
dnssec-ws01.dnssec.bayern     IN  SSHFP  1 1  57D79129FA85BCC0D9E15CE25C96E639E2DB3316
dnssec-ws01.dnssec.bayern     IN  SSHFP  2 1   CF239AE438FF149185378D9735BE42B519416D0F
```

SSHFP RR in der Zone


```
$ sshfp -s dnssec-ws01.dnssec.bayern
dnssec-ws01.dnssec.bayern     IN  SSHFP  1 1  57D79129FA85BCC0D9E15CE25C96E639E2DB3316
dnssec-ws01.dnssec.bayern     IN  SSHFP  2 1   CF239AE438FF149185378D9735BE42B519416D0F
```

- sshfp erstellt RR Eintrag für SSH Hostkey Fingerprint (-s scan for public key)

SSHFP RR in der Zone


```
$ sshfp -s dnssec-ws01.dnssec.bayern
dnssec-ws01.dnssec.bayern     IN  SSHFP  1 1  57D79129FA85BCC0D9E15CE25C96E639E2DB3316
dnssec-ws01.dnssec.bayern     IN  SSHFP  2 1   CF239AE438FF149185378D9735BE42B519416D0F
```

- sshfp erstellt RR Eintrag für SSH Hostkey Fingerprint (-s scan for public key)
- Unterstützte Algorithmen:
  1 ssh-rsa
  2 ssh-dsa
  3 ecdsa
  4 ed25519

SSHFP RR in der Zone


```
$ sshfp -s dnssec-ws01.dnssec.bayern
dnssec-ws01.dnssec.bayern     IN  SSHFP  1 1  57D79129FA85BCC0D9E15CE25C96E639E2DB3316
dnssec-ws01.dnssec.bayern     IN  SSHFP  2 1   CF239AE438FF149185378D9735BE42B519416D0F
```

- sshfp erstellt RR Eintrag für SSH Hostkey Fingerprint (-s scan for public key)
- Unterstützte Algorithmen:
  1 ssh-rsa
  2 ssh-dsa
  3 ecdsa
  4 ed25519
- SSHFP RR record dann in der Zone eintragen

DNS SSHFP-Einträge überprüfen

SSHFP RR records mit dig abrufen

```
$ dig +short www.kernel-error.de IN SSHFP
3 1 3DD9DE0DCF1523341B45A53F1D57043609E26C62
3 2 E1C76BD66B5A0641789B0B37BE5B80AE3F6395C1CD2B73CCA532A811 1C9515B4
1 1 47890EECC9A2893061734B07B8F60CAA1A856148
1 2 B2518AD49CC2ADF517D3F6A9FAAF4017ABC2C3E33DAE0A29C46226E9 FF691CD2
SSHFP 7 3 86400 20160307114536 20150313114536 13952 kernel-error.de.
hib4uoiPCBlAXytsVHsqvGdGDRPdec42nA2J7IW8mQVV/o7fb5kPfV6J Zlp4D6O0RsSWW+
SSHFP 10 3 86400 20160307114536 20150313114536 12698 kernel-error.de. dbv2KMbxb4Vc
+kLz86hGjdc12b
```

DNSSEC mit dig auf Vollständigkeit und Sicherheit überprüfen

```
$ dig +dnssec www.kernel-error.de IN SSHFP |grep flags
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 6, AUTHORITY: 0, ADDITIONAL: 1
; EDNS: version: 0, flags:; MBZ: 7e06 , udp: 512
```

In der Antwort muss das **ad** flag gesetzt sein!

DNSSEC für andere Anwendungen

DNSSEC läßt sich zur Absicherung von kryptographischen Informationen in anderen Anwendungen verwenden

- XMPP/Jabber
- SIP
- SSHFP — *SSH Host key fingerprint*
- IPSEC Key — *IPSEC keys in DNS*
- OPENPGPKEY — *PGP keys in DNS*
- …

Nützliche Ressourcen - Anleitungen

- DNSSEC HowTo - A tutorial in disguise
  https://www.nlnetlabs.nl/publications/dnssec_howto/dnssec_howto.pdf
- BIND DNSSEC Guide
  https://users.isc.org/~jreed/dnssec-guide/dnssec-guide.html
- BIND Automatic Signing
  http://www.average.org/dnssec/dnssec-configuring-auto-signed-dynamic-zones.txt
- White paper Deploying DNSSEC
  https://www.surf.nl/binaries/content/assets/surf/en/knowledgebase/2012/rapport_Deploying_DNSSEC_v20.pdf
- Heise Artikel http://www.heise.de/netze/artikel/Transitschutz-DNSSEC-und-DANE-auf-Linux-Servern-konfigurieren-2636175.html

Nützliche Ressourcen - Youtube Videos

- Auf Youtube finden sich viele kurze oder auch ausführliche Präsentationen zu DNSSEC und DANE

Vielen Dank für Ihre Aufmerksamkeit! Fragen?