Protection of Communication Infrastructures

Chapter 5
Secure Name Resolution

- DNS Security Issues
- DNSSEC
- Alternative Approaches

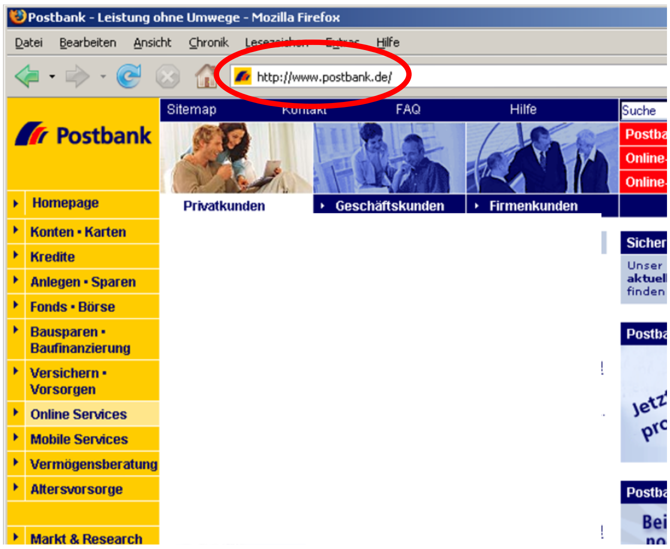
Security of the Domain Name System

- Vital service for the Internet
  - “Do you know the IP-Address of your mail server?”
  - “You know you should not follow the link
    `http://very.malicio.us`
    but what about
    `http://www.yourbank.de` ??”
- But: DNS does not support
  - Data integrity
  - Data origin authentication
  - (Confidentiality)
- Threats:
  - Data Authenticity/Integrity
  - Denial of Service

[Acknowledgement: Thanks to Thorsten Strufe for help in preparing this material.]

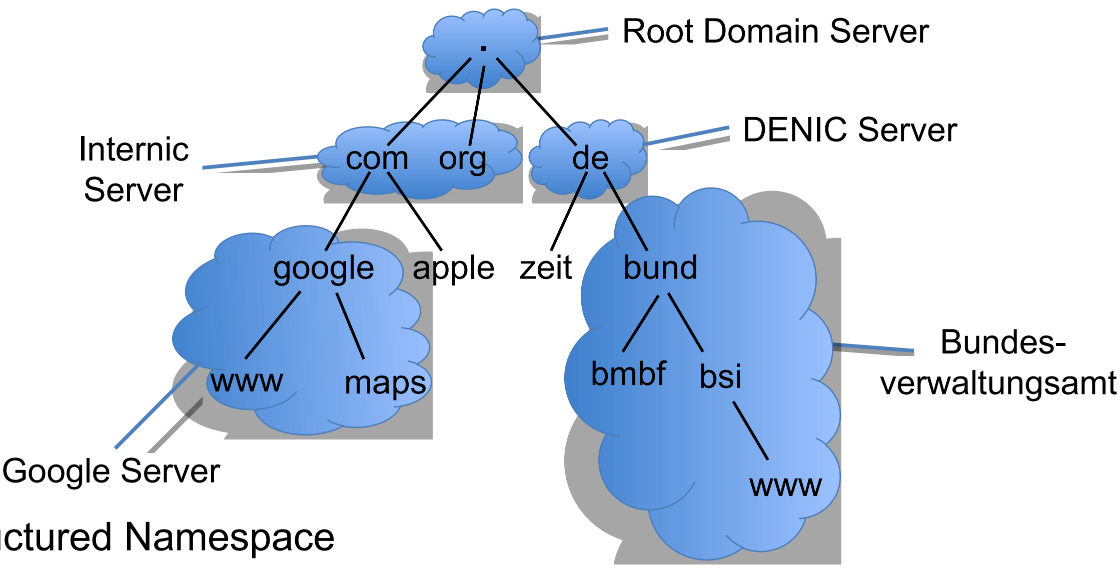
DNS – The Domain Name System

- Naming service for (almost all) Internet traffic
- Lookup of (resolve)
  - IP addresses
  - Mail servers
  - Alias names
  - Alternative name servers
  - …



- Decentralized database consisting of multitude of servers

DNS – Data Organization: Domains / Zones



- Structured Namespace
- Hierarchical organization in sub domains/zones
- Sourced at "root zone" (". ")
- Parent zones maintain pointers to child zones ("*zone cuts*")
- Zone data is stored as "Resource Records" (RR)

DNS – Components

- Authoritative Server
  - Server maintaining authoritative content of a complete DNS zone
  - Top-Level-Domain (TLD) servers & auth servers of organization's domains
  - Pointed to in parent zone as authoritative
  - Possible load balancing: master/slaves
- Recursive (Caching) Server
  - Local proxy for DNS requests
  - Caches content for specified period of time (soft-state with TTL)
  - If data not available in the cache, request is processed recursively
- Resolver
  - Software on client's machines (part of the OS)
  - Windows, MacOS, and *nix: Stub resolvers
    - Delegate request to local server
    - Recursive requests only, usually no iterative requests

DNS – Resource Records

- Atomic entries in DNS are called "Resource Records" (RR)
- Format:

  <name> [<ttl>] [<class>] <type> <rdata>

  - `name` (domain name of resource)
  - `ttl` (Time-to-live)
  - `class` (used protocol): IN (Internet), CH (Chaosnet)…
  - `type` (record type): A (Host-Address), NS (Name Server), MX (Mail Exchange), CNAME (Canonical Name), AAAA (IPv6-Host-Address), DNAME (CNAME, IPv6)
  - `rdata` (resource data): Content! (What did we want to look up?)

DNS – Message Format

0	1	2	3
Identification		Flags and Codes	
Question Count		Answer Record Count	
Name Server (Auth Record) Count		Additional Record Count	
Questions			
Answers			
Authority			
Additional Information			

Q/R	OPCode	AA	TC	RD	RA	Zero	RespCode
16	17	21	22	23	24	25	28–31

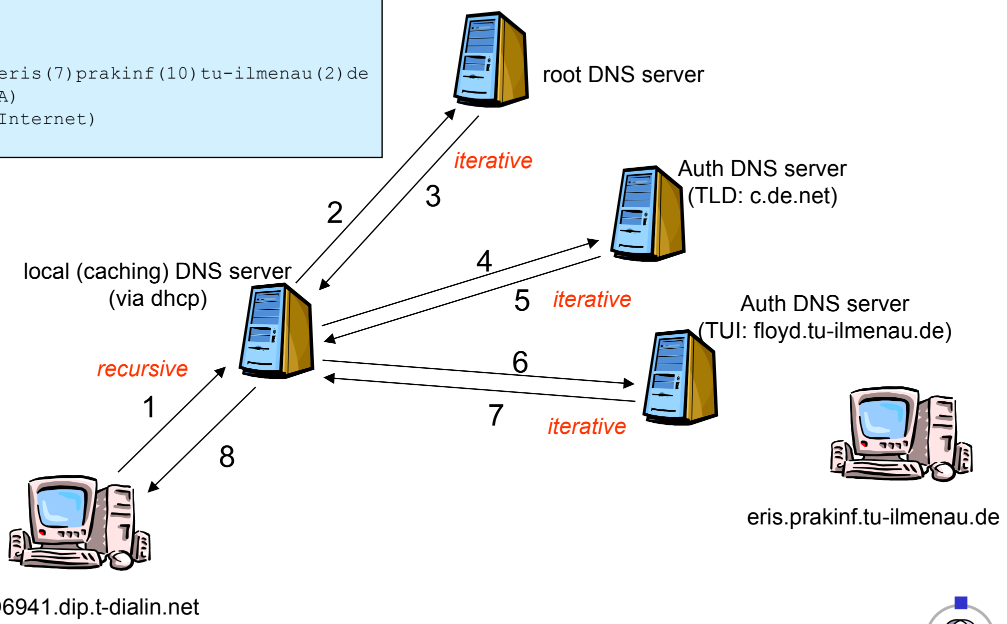
- Q/R *Query/Response Flag*
- *Operation Code*
- AA *Auth. Answer Flag*
- TC *Truncation Flag*
- RD *Recursion Desired Flag*
- RA *Recursion Available Flag*
- Zero (three resv. bits)
- *Response Code*

DNS – Header Fields

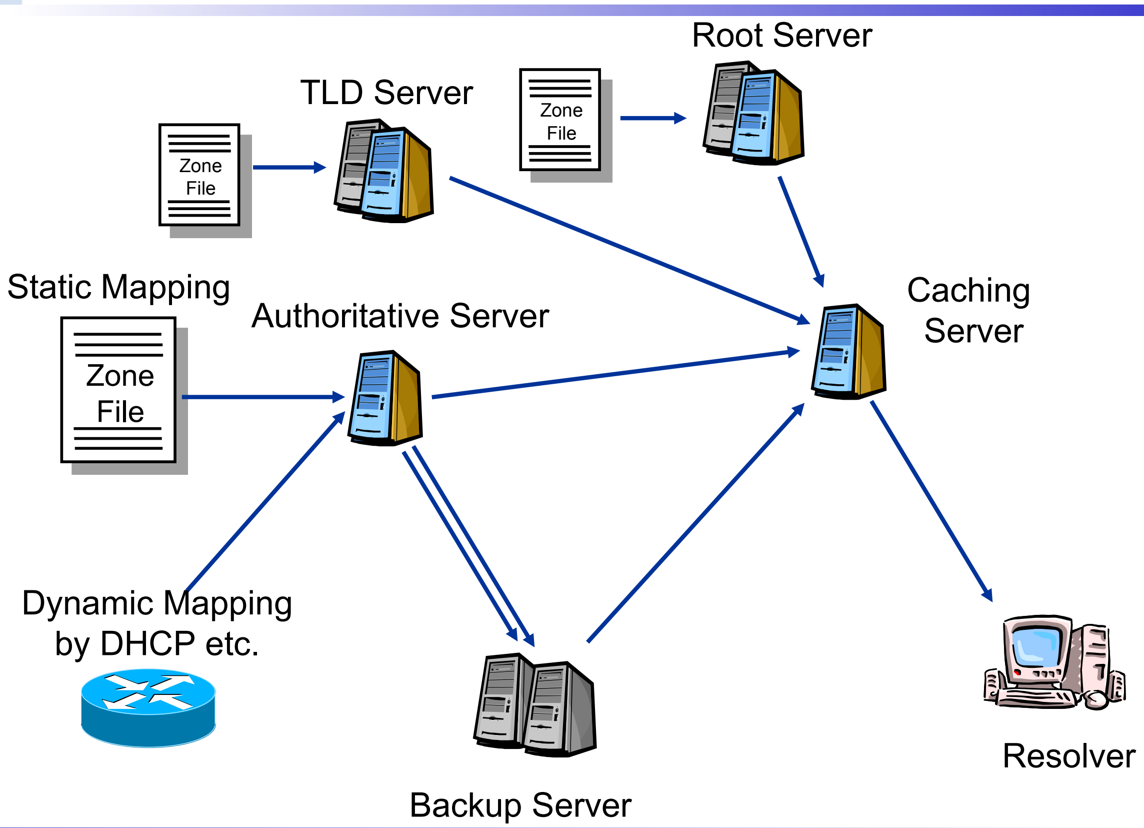
- *Identifier:* a 16-bit identification field generated by the device that creates the DNS query. It is copied by the server into the response, so it can be used by that device to match that query to the corresponding reply
- *Query/Response Flag:* differentiates between queries and responses (0 ~ Query, 1 ~ Response)
- *Operation Code:* specifies the type of message (Query, Status, Notify, Update)
- *Authoritative Answer Flag (AA):* set to1 if the answer is authoritative
- *Truncation Flag:* When set to 1, indicates that the message was truncated due to its length (might happen with UDP, requestor can then decide to ask again with TCP as transport service)
- *Recursion Desired:* set to 1 if requester desired recursive processing
- *Recursion Available:* set to 1 if server supports recursive queries

DNS – Recursive and Iterative Queries

```
DNS HEADER (send)
- Identifier:           0x6012
- Flags:                0x00 (Q )
- Opcode:               0 (Standard query)
- Return code:          0 (No error)
- Number questions:     1
- Number answer RR:     0
- Number authority RR:  0
- Number additional RR: 0
QUESTIONS (send)
- Queryname:            (4)eris(7)prakinf(10)tu-ilmenau(2)de
- Type:                 1 (A)
- Class:                1 (Internet)
```



DNS – Data Flow



DNS Security Objectives

- Authentication and integrity of name mappings
  - Most prominent thread: Cache Poisoning
  - Simple Countermeasures:
    - Random Port Numbers, Split-Horizon DNS, no PMTUD…
  - Cryptographic countermeasures
    - Data Integrity with TSIG records
    - DNSSEC
- Availability
  - DNS being a target of DoS attacks
  - DNS being a mean of amplification attacks
- (Confidentiality & Access Control)

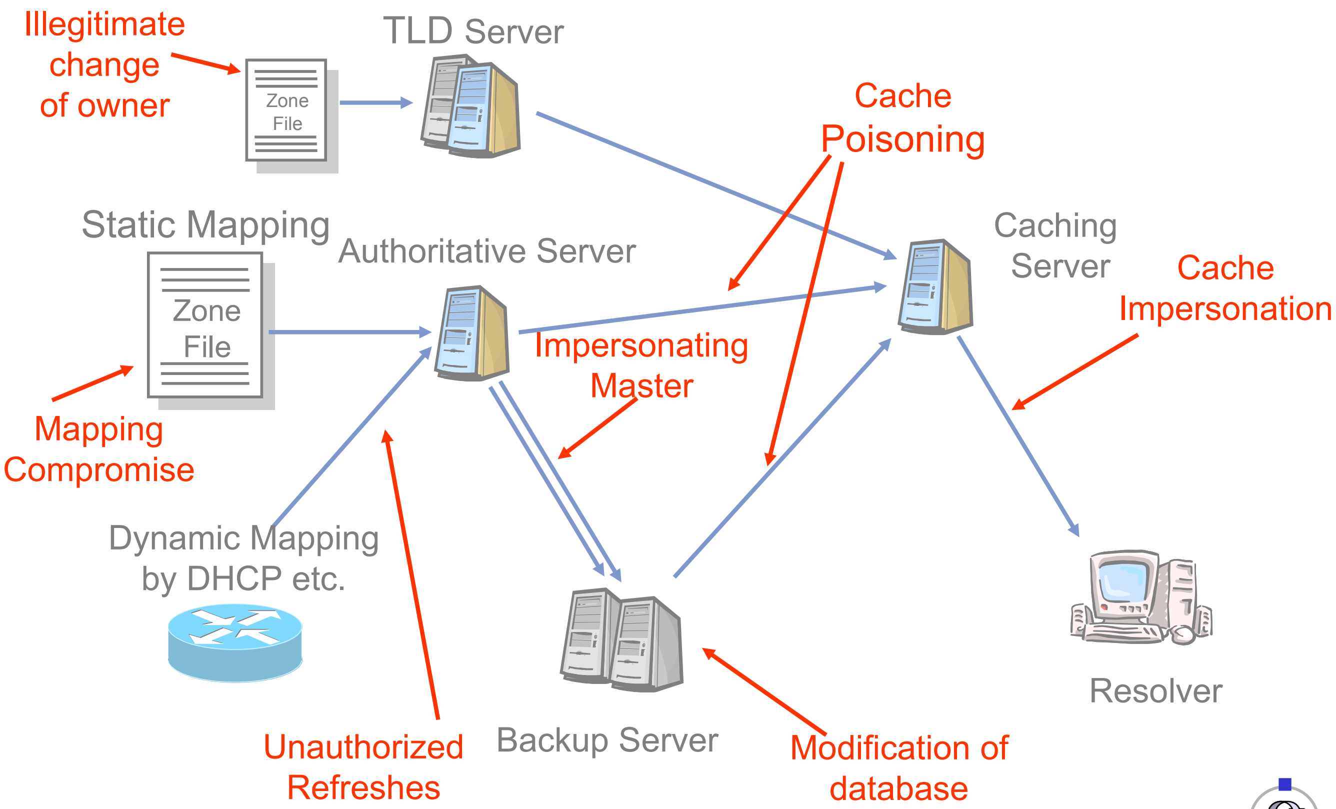
Threats to DNS: Denial of Service

- DNS itself as vital service a "worthy" target
  - Without DNS most Internet services will not work
    (usage of names rather than IP-Addresses for numerous reasons!)
  - DDoS Attacks on root servers: via notorious "typos" in TLDs
  - Extremely difficult to trace back!
- DNS Amplification Attacks
  - Spoofed queries (60 Bytes) may generate potentially large responses (4KBytes)
  - Exploit open recursive servers to generate load on other DNS servers
  - Exploit open servers as reflectors when flooding a victim with traffic
    (via source IP Address spoofing in request)

Robustness towards DDoS

- General issues
  - Secure DNS server
    - OS selection and updates
    - Firewalls
    - Server software selection and updates
  - Redundancy and over provisioning
    - Root ".": 13 name server "names" (`{a..m}.root-servers.net`)
    - "com", "net" 13 name servers each, "de" 16 name servers
  - Anycast
    - Announcement of an IP prefix from multiple locations
    - Requests from different parts of the internet are routed to different machines with the same IP address
    - Done by 11 of the 13 root-servers

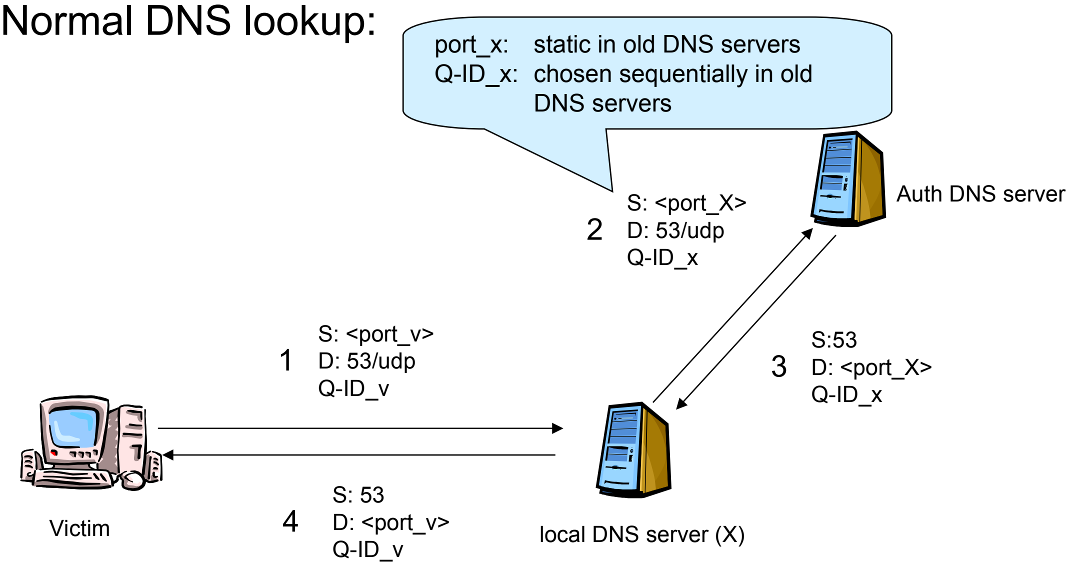
DNS – Threats to Data Integrity & Authentication



Threats to DNS: Data Corruption / Cache Poisoning

- All resolved RRs are cached at local DNS servers
- DNS slave servers replicate zone data from master

- Normal DNS lookup:



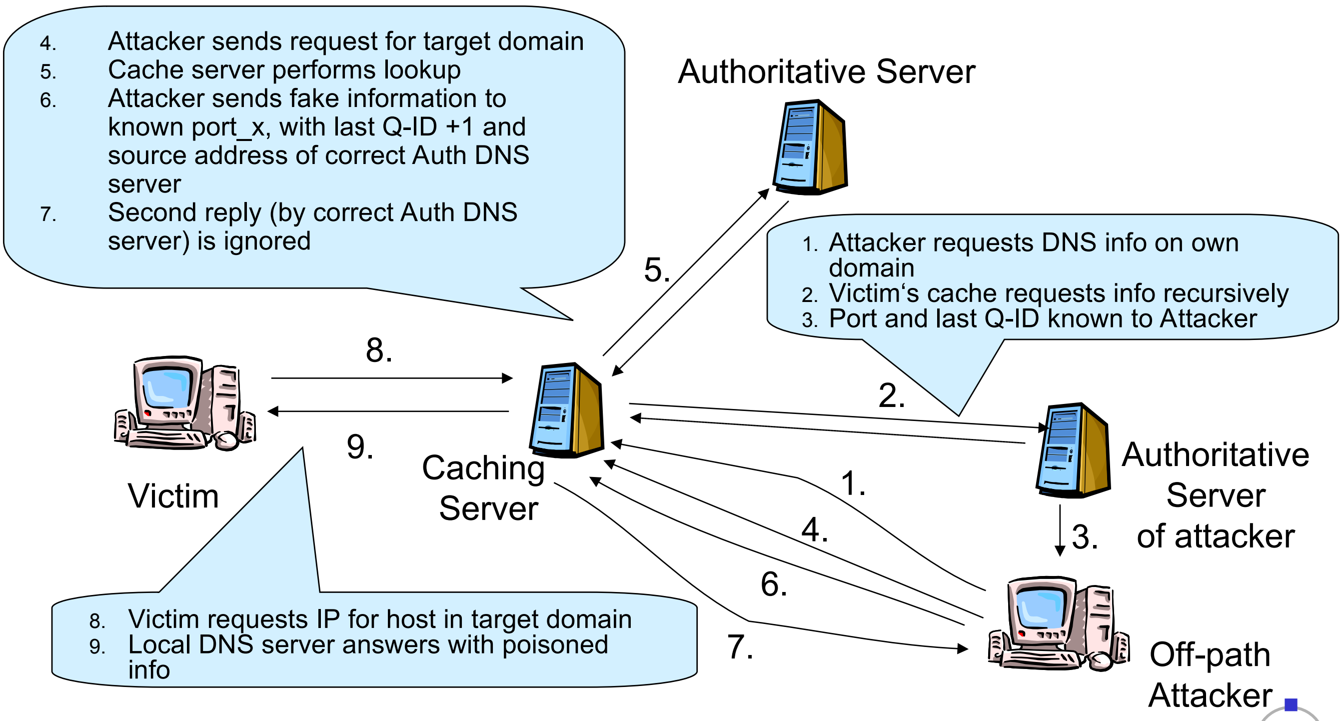
DNS Threats – Cache Poisoning: Simple Poisoning (1)

- Attack: put fake data in slaves / caching servers
  (*and nobody will realize the redirection from* **`www.yourbank.de`** *to* **`very.malicio.us`** *…)*

- DNS via UDP, no handshakes for connection establishment
- Transactions in DNS only through tuple of
  `<auth server(ip address), auth server(port), transaction id>`

- With knowledge about transactions distribute malicious data
- **IP Address** of authoritative name servers are well known
- In many implementations same **port** for all transactions
- **Q-ID** unknown, *but*: servers used to choose them sequentially…

DNS Threats – Cache Poisoning: Simple Poisoning (2)

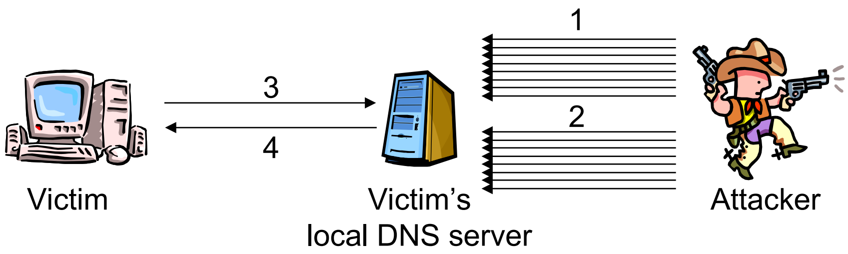


Mitigation of Cache Poisoning

- Random ports for each transaction (BIND8)
  - Since Version 8 BIND uses PRNG for port number and query id selection
  - However PRNG == ***Pseudo*** Random Number Generator, with knowledge about previous port numbers future port numbers can be guessed if PRNG not cryptographically secure (see network security course chapter 6)
- More random ports for each transaction (BIND9)
  - New and better PRNG since BIND9, random numbers are harder to guess
- Cache Poisoning usually only after aging of entry in local DNS server
  - Only if attacker attacks at the right moment, he can poison the cache
  - Typical TTL:
    - 172800 (2d) for most name servers
    - Seconds to hours for A-Entries of organizations (tu-ilmenau.de 4h, deutschebank.de 30min, commerzbank.de 60mins, postbank.de 60s)
- *Nevertheless: cache poisoning is still not solved completely!*

Cache Poisoning with "Brute Force"

1. Attacker sends multitude of requests for targeted domain to local DNS server of victim and
2. Attacker sends multitude of fake replies with IP and port from auth server of targeted domain, guessing transaction id for one of the recursive requests from local caching server to auth server ($2^{16} \times 2^{16} = 2^{32} \approx$ 4 billion possible combinations in theory)
3. Victim requests data about targeted domain
4. Local caching server responds with fake data



More Sophisticated Cache Poisoning

- Usually not a high number of chances when TTL high, e.g., 1 day
- But there is a trick [Kam08]
- Imagine an attacker M:
  - M → Cache: Give me `kslkskdf.bank.com` (`kslkskdf` being a random value)
  - The Cache Server must now ask the Authoritative Server at `bank.com`
  - M → Cache: Not responsible for `kslkskdf.bank.com`, but `www.bank.com` is. You may reach `www.bank.com` at `141.24.212.114` (with `141.24.212.114` being the address of the attacker)
  - The cache will now ask `141.24.212.114` for `kslkskdf.bank.com` and will also remember the "name server" `www.bank.com`
- Even the ideal entropy of 2^{32} is insufficient!

More Sophisticated Cache Poisoning - Defense

- How can we increase the entropy of DNS queries?
- Idea: DNS does not distinguish between upper and lower case, encode more bits in the name [DAV+08]
- Now the same attack:
  - M → Cache: Give me `kslkskdf.bank.com`
  - The Cache Server must now ask the Authoritative Server at `bank.com` Cache → Auth Server: Give me `kSLkSkdF.bAnK.COM`
  - M → Cache: Not responsible for `kslkskdf.bank.com`, but `www.bank.com` is. You may reach `www.bank.com` at `141.24.212.114`. (Ignored as `kslkskdf.bank.com` does not match the case of the query)
  - Auth Server → Cache: `kSLkSkdF.bAnK.COM` is unknown

- Entropy is increased to 2^{32+n} for n being the letters in a domain name
- Helps for `www.deutsche-bank.de` but not much for `db.com`

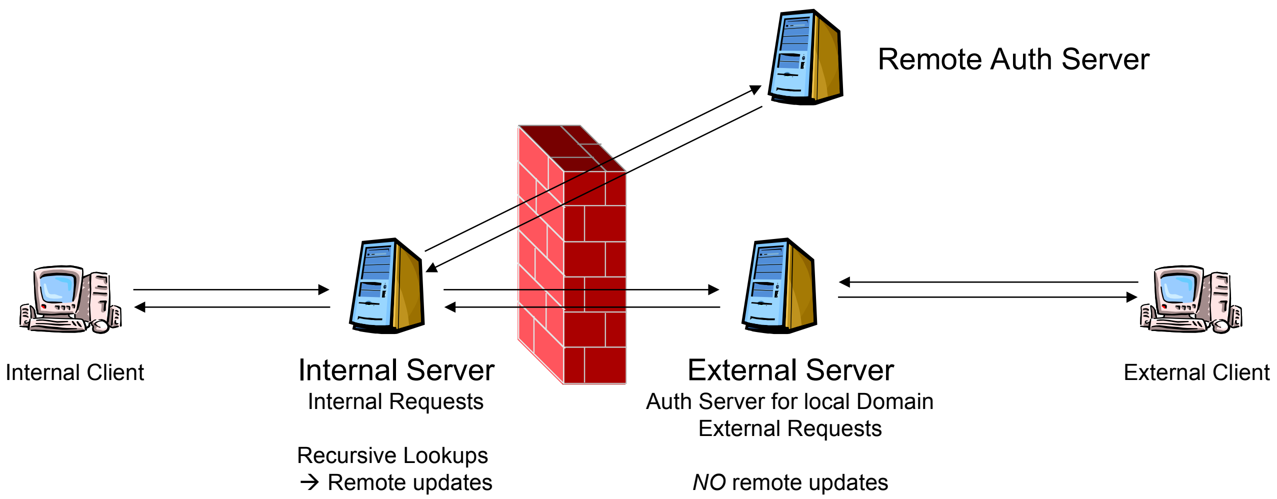
Most Sophisticated Cache Poisoning

- DNS is usually transported over UDP, which may fragment
- What happens when a DNS reply gets fragmented?
  - Random port numbers, Query ID and perhaps the original question (e.g. `kSLkSkdF.bAnK.COM`) are in the first fragment
  - Depending on the query and the MTU the actual answer may be in the second fragment
  - Fragments are matched by a 16 bit identification field…
- Attackers thus can try to [HS13b]
  - Find queries leading to large answers
  - Spoof PTMU related ICMP messages to set the fragmentation boundary
  - Send a “second” fragment with different identifications to the cache
  - Send the query to the cache
  - Wait for the cache to reassemble the reply and the crafted second fragment…
- DNS server should avoid large answers and PMTU discovery...

Robustness towards Data Corruption: Split-Horizon DNS (1)

- Goal: Avoid cache poisoning from external machines
- Idea: Split the name service functions
  - Name resolution (look up of DNS info)
  - Domain information (Auth service of local DNS info)
- Internal server
  - Implements name resolution
  - Performs recursive look-ups at remote DNS servers
  - Located behind firewall and only accepts requests from local LAN
- External server
  - Authoritative server of domain
  - Accepts remote requests but ***never accepts*** remote updates
- Zone transfer from external to internal server allowed

Split-Horizon DNS (2)



Robustness towards Data Corruption: Data Integrity

- Secret Key Transaction Authentication for DNS (TSIG) [VGE00]
- Usage of signatures to secure data at zone transfer master → slave
- Symmetric keys between entities
  - MD5 Hash used as signature or
  - HMAC with SHA-1 and SHA-2 [Eas06]

- TSIG Resource Record:
  (Name, Type (“TSIG”), Class (“ANY”), TTL(“0”), Length, Data(<signature>))

- Possibility to authenticate, but very complex to administrate in large domains (manual pre-sharing of keys)
  amount of keys required: n x (n-1) / 2
- Kerberos may help [Eas00], but not on global scope…
- Main application area: Secure communication between authoritative and backup servers

DNSSEC – Objectives

- DNSSEC aims at:
  - End-to-end zone data origin authentication and integrity
    - in a global scope
    - without breaking the established DNS mechanisms

- DNSSEC does not provide:
  - Availability (in fact, it facilitates DoS Attacks on DNS servers)
  - Confidentiality
  - Controlled Access (makes it even worse)
  - Guarantee for correctness of data (only that it has been signed by some authoritative entity)

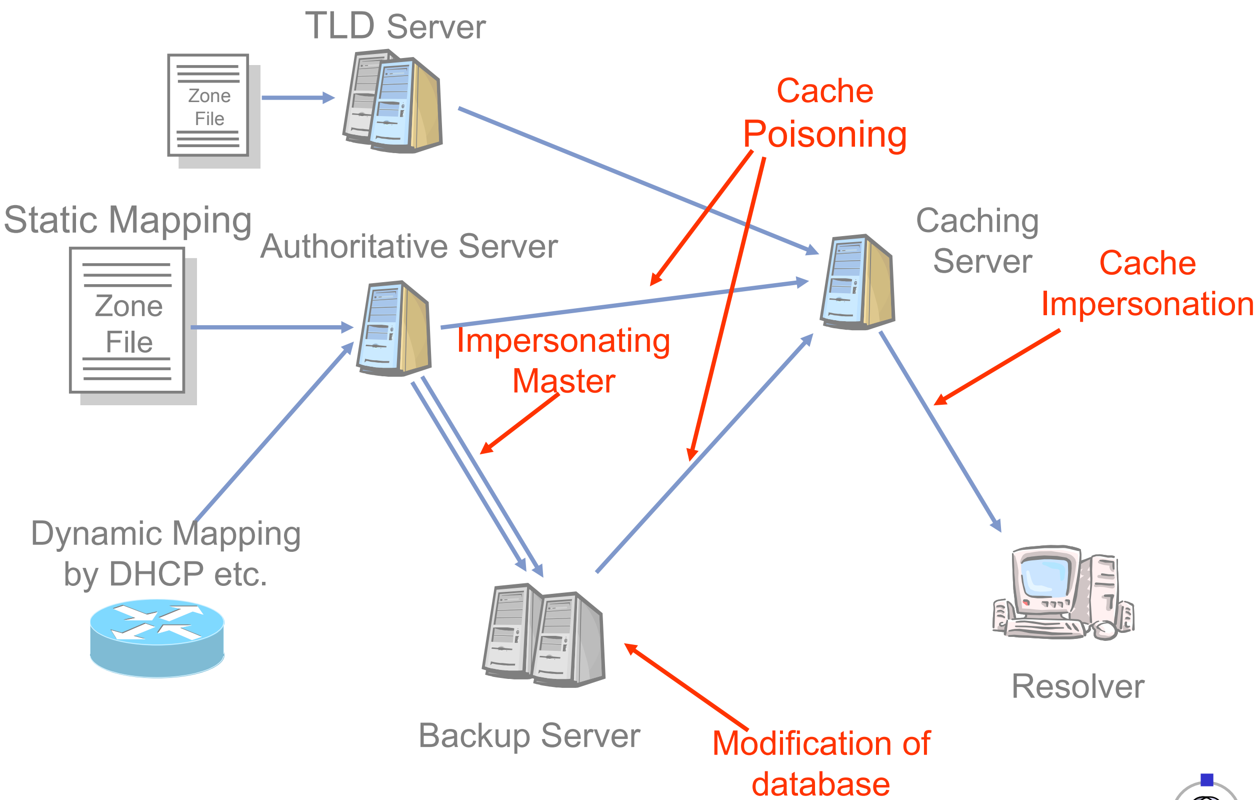
DNSSEC – Overview

- Usage of public key cryptography for data origin authentication on a world wide scale
- Use of DNS itself to distribute signatures and keys!

- RRSets (groups of RRs) are signed with the private key of authoritative entities
- Public keys (DNSKEYs) published using DNS
- Child zone keys are authenticated by parents (according to the zone hierarchy) and hence anchored *trust chains* established
- Only root zone key signing key (KSK) needed (manual distribution) to create complete trust hierarchy (for supported zones)

- No key revocation → DNSSEC keys should have short expiration date (quick rollover)

DNSSEC – Targeted Threats



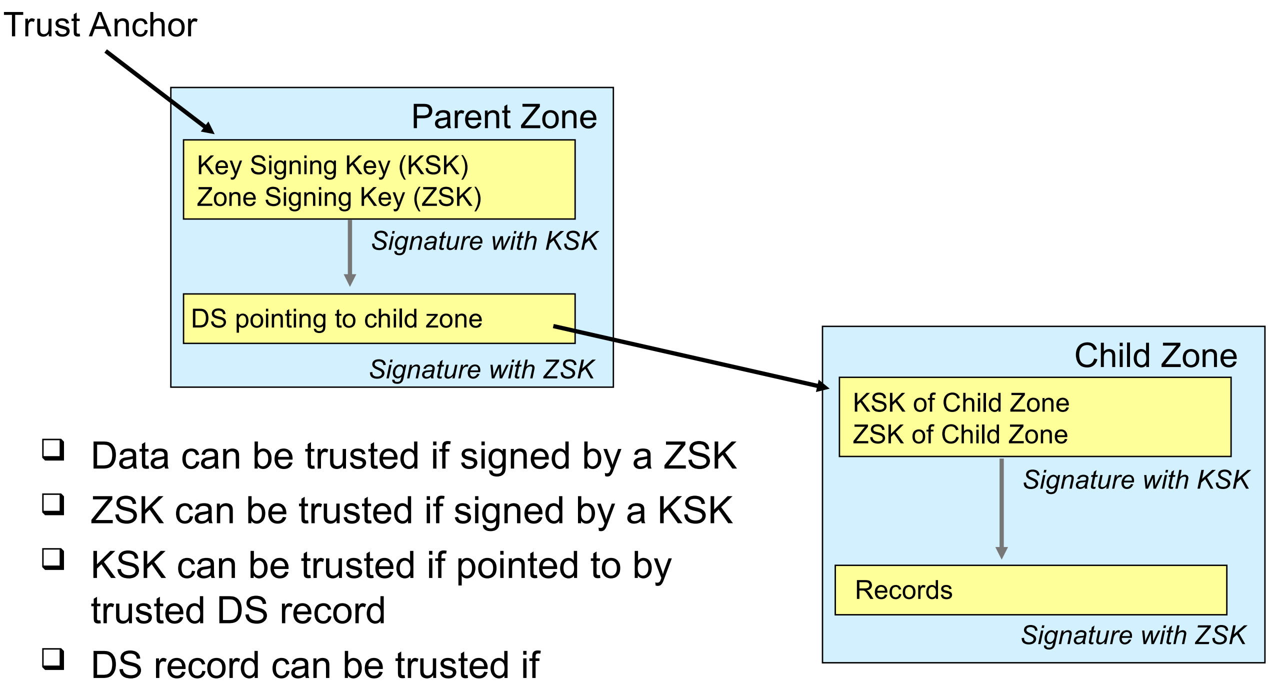
DNSSEC – Means of Securing RRSets

- Goal: authenticity and integrity of Resource Record Sets

- Means:
  - Public Key Cryptography (with Trust Chains)
  - Security integrated in DNS (new RRs)

- New Resource Record Types:
  - RRSig: RR for signatures to transmitted RRs
  - DNSKEY: RR for transmission of public keys
  - DS: RR for trust chaining (Trust Anchor signs Key of Child Zone)
  - NSEC: RR for next secure zone in canonical order (*authenticated denial* for requested zone; introduces new confidentiality/privacy problems)
  - NSEC3: RR for next secure zone in canonical order (hashed)

DNS – Authority Delegation and Trust Chaining



- Data can be trusted if signed by a ZSK
- ZSK can be trusted if signed by a KSK
- KSK can be trusted if pointed to by trusted DS record
- DS record can be trusted if
  - Signed by parents ZSK
  - Signed by locally configured trusted key

DNS – Authority Delegation and Trust Chaining (Example)

Trusted Key (locally configured)

Parent Zone

```
child    NS       ns.child
         DS       (…) <KSK-id>
         RRSIG    DS (…) parent.
```

Child Zone

```
@NS
    ns
    DNSKEY  (…) <KSK-id>
    DNSKEY  (…) <ZSK-id>
    RRSIG   dnskey (…)<KSK-id> parent.
    RRSIG   dnskey (…)<ZSK-id> child.parent.

ns  A       10.5.1.2
    RRSIG   A (…) <ZSK-id> child.parent.

www A       10.5.1.3
    RRSIG   A (…) <ZSK-id> child.parent.
```


DNSSEC Deployment

- KSK for Root Zone
  - Stored in two redundant Hardware Security Modules (HSM)
  - Each activated by 3 out 7 international experts (*Crypto Officers)*
  - Recovery by 5 out of 7 different international experts (*Recovery Key Share Holder)*
  - Sign ZSK for root zone (under control of Verisign, a US company…)
- In Subzones: Splitting KSK and ZSK allows for different security levels
  - KSK generates long-living signatures (stored offline)
  - ZSK generates short-living signatures (“accessible” system)
  - But what if ZSK becomes compromised?
    - Cannot change ZSK quickly (as KSK signatures are long living)
    - Asking provider to change KSK is quicker…
    - Sense of splitting?!

DNSSEC – New Resource Records: RRSIG

- Resource Record for transmission of signatures
- RRSIG:
  - `Name` – name of the signed RR
  - `Type` – RRSIG (46)
  - `Algorithm` – DSA (3), **RSASHA1** (5), RSASHA1-NSEC3-SHA1(7) RSASHA256 (8), ECDSAP256SHA256 (13)…
  - `Labels` – number of labels in original RR (to differentiate wildcards)
  - `TTL` – TTL at time of signature (for later reconstruction)
  - `Signature Expiration` – End of validity period of signature
  - `Signature Inception` – Beginning of validity period of signature
  - `Key Tag` – ID of used key if signer owns multiple keys
  - `Signer's Name` – DNS Name of the signer
  - `Signature` – Actual Signature

DNSSEC – New Resource Records: DNSKEY

- Resource Record containing public keys for distribution
- DNSKEY:
  - `Label` – Name of key owner
  - `Class` – Always IN (3)
  - `Type` – DNSKEY (48)
  - `Flags` – key types: KSK (257) or ZSK (256)
  - `Protocol` – Always DNSSEC (3)
  - `Algorithm` – DSA (3), **RSASHA1** (5), RSASHA1-NSEC3-SHA1(7), RSASHA256 (8), ECDSAP256SHA256 (13)…
  - `Key` – Actual key

DNSSEC – New Resource Records: DS

- DS contains hash-value of DNSKEY of the name server of a sub zone
- Together with NS Resource Record, DS is used for trust chaining

- DS (Delegation Signer)
  - `Name` – Name of the chained sub zone
  - `Type` – DS (43)
  - `Key Tag` – Identification of the hashed key
  - `Algorithm` – DSA (3), **RSASHA1** (5), RSASHA1-NSEC3-SHA1(7), RSASHA256 (8), ECDSAP256SHA256 (13)…
  - `Digest Type` – SHA-1(1), SHA-256(2)
  - `Digest` – Actual value of hashed DNSKEY

DNSSEC – New Resource Records: NSEC

- Next Secure (NSEC) gives information about the next zone / sub domain in canonical order (last entry points to first entry for the construction of a closed ring)
- Gives the ability to prove the non-existence of a DNS entry: *Authenticated Denial*

- NSEC
  - `Name` – Name of the signed RR
  - `Type` – NSEC (47)
  - `Next Domain` – Name of the next domain in alphabetical order

- Authenticated denial with NSEC gives the possibility to “walk” the chain of NSEC and to gain knowledge on the full zone content (all zones/ sub domains) in O(N)
  - This may be a privacy violation

DNSSEC – New Resource Records: NSEC3 (1)

- Successor to NSEC: NSEC3 and NSEC3PARAM [LSA08]
- Uses hashed domain names to make zone walking more difficult
- Hashing based on salt and multiple iterations to make dictionary attacks more difficult
- NSEC3
  - `Name` – Name of the signed RR
  - `Type` – NSEC3 (50)
  - `Hash Algorithm` – SHA-1 (1)
  - `Flags` – To Opt-Out unsigned names
  - `Iterations` – Number of iterations of Hash Algorithm
  - `Salt Length` – Length of the salt value
  - `Salt` – Actual salt value
  - `Hash Length` – Output length of hash function
  - `Next Hashed Owner Name` – Next Hash of domain name in alphabetical order

TELEMATIK Rechnernetze

DNSSEC – New Resource Records: NSEC3 (2)

- Potential advantage: Salting and hashing does not allow for easily deducting hostnames from zone walks
- Problem:
  - Hostnames usually have very low entropy (to remember them)
  - Easy dictionary attacks - despite the usage of salts & iterations
  - But not used heavily anyways:
    - .: Uses NSEC
    - .com: No salt, No iterations
    - .de: Static salt BA5EBA11, 15 Iterations
- Only solution generate NSEC3 records "on-demand"
  - For a question q generate NSEC3 answers for h(q) - 1 and h(q) + 1
  - Slow (DoS) & Protection of keys?
- NSEC3 = Complicated mechanism for low gain…

© Dr.-Ing G. Schäfer

DNSSEC Issues

- Pro's:
  - DNSSEC allows to counter unauthorized/spoofed DNS records
- Con's:
  - Added complexity (signing, checking, key distribution)
    - Significant chance of misconfiguration
    - DoS attacks on DNS servers
  - Zones completely need to be signed otherwise no gain!
  - No confidentiality of requests/responses
  - Authenticated denial with NSEC and NSEC3 gives the possibility to "walk" and to gain knowledge on the full zone content (all zones/ sub domains) in O(N)
    - This may be a privacy violation due to some regulations…
  - Trust hierarchy without redundancy & anchor keys in the hand of single US company

Alternatives to DNSSEC*

- Some more or less exotic approaches
- „Transport Layer" Encryption
  - DNSCurve
- Distributed Name Resolution
  - Peer Name Resolution Protocol
  - GNU Name System
- Try different strategies to deal with Zookos triangle, informally stating that global name resolution can only offer any two of the following properties:
  - Security
  - Distributed system
  - Human-memorable Names

DNSCurve (I)

- Developed by Daniel Bernstein as an alternative to DNSSEC [Ber09]
- Uses online cryptographic functions, i.e., more like transport layer encryption
- Exchanges (either binary or tunneled in DNS TXT records):

$$C \rightarrow S: \left(+K_C, r_C, B[r_C, -K_C, +K_S, Request]\right)$$
$$S \rightarrow C: \left(r_C, r_S, B[r_C || r_S, -K_S, +K_C, Reply]\right)$$

- With B[] being the "cryptographic box function" developed by Bernstein

$$K_s := H(\{+K_2\}_{-K_1}, 0) = H(\{+K_1\}_{-K_2}, 0)$$
$$B[n, -K_1, +K_2, m] := H_2(\{m\}_{g(K_s,n)}) || \{m\}_{f(K_s,n)}$$

- Online operation possible using very efficient ciphers: Curve25519 (ECC based pub key system), Poly1305 (hash function) and Salsa20 (stream cipher) all developed by Bernstein

DNSCurve (II) – Distribution of public keys

- Public keys are not validated using novel records, but DNS names
- Name servers are no longer called `ns.example.org` but like `uz5xgm1kx1zj8xsh51zp315k0rw23csgyabh2sl7g8tjg25ltcvhyw.example.org`
  - `uz5` means name server speaks DNSCurve
  - Rest of the string encodes 256 bit public key
- Parent zones pointing to DNSCurve names allow to verify sub zones
- DNSCurves in hyperlinks etc. allow to verify DNSCurve servers independent of DNS hierarchy! → No more trust in root zone…

DNSCurve (III) – Discussion

- Slimmer design compared to DNSSEC
- Independent trust paths possible
- Support for confidentiality
- No amplification attacks possible

- But use of hard defined cryptographic operations is not globally acceptable
- Cipher operations are not as well examined as ciphers in DNSSEC

Peer Name Resolution Protocol (PNRP) [Mic13]

- Distributed protocol invented by Microsoft based on Pastry
- Deployed with modern Windows systems, but disabled by default and requiring IPv6
- All computers are equal nodes in structured overlay networks (called clouds here)
  - Link cloud (for all local computers)
  - Site cloud (e.g. for all computers of a company)
  - Global cloud (for all PNRP speakers in the Internet)
- Names in the form `*name*.pnrp.net` can be resolved insecurely (`pnrp.net` being a special domain to map PNRP names in the DNS name space)
- Names in the form `pnamep-*authority*.pnrp.net`, with `authority` being a SHA-1 hash of a node's public key can be securely resolved

PNRP – Properties

- Abandonment of servers might increase robustness and availability of overall system, but no wide research (yet)
  - Some mechanisms to prevent well known eclipse and sybil attacks
- Different clouds allow for increased performance and robustness within same network locations
- Secure names can be verified independently of a trust hierarchy

- Secure names cannot be remembered
- Secure names cannot be verified by a trust hierarchy
- In the insecure name space: anarchy!
  - No security guarantees at all
  - Anybody may register and reregister addresses…

GNU Name System (GNS) [EG11]

- Also distributed name resolution system based on GNUnet (spanning a Kademlia like structured peer-to-peer overlay)
- Also offers secure names in the form `hash.zkey`
- Again not memorable, but GNS offers aliases (so called *petnames*)
  - Every participant may create aliases like `alias.gnu` pointing to `something.zkey`
  - and aliases may be recursive (if users permit)… thus `bob.alice.gnu` points to a system that Alice (known by local system) calls `bob`

- Allows locally unambiguous names with a clear trust hierarchy
- Global names still possible by unmemorable names

References (1)

[AAL+05a] ARENDS, R. ; AUSTEIN, R. ; LARSON, M. ; MASSEY, D. ; ROSE, S.: DNS Security Introduction and Requirements. 2005. – RFC 4033, IETF, Status: Proposed Standard, https://tools.ietf.org/html/rfc4033

[AAL+05b] ARENDS, R. ; AUSTEIN, R. ; LARSON, M. ; MASSEY, D. ; ROSE, S.: Resource Records for the DNS Security Extensions. 2005. – RFC 4034, IETF, Status: Pro- posed Standard, https://tools.ietf.org/html/ rfc4034

[Ber09] BERNSTEIN, D. J.: DNSCurve: Usable security for DNS. http: //dnscurve.org. Version:2009

[DAV+ 08] DAGON, D. ; ANTONAKAKIS, M. ; VIXIE, P. ; JINMEI, T. ; LEE, W.: Increased DNS forgery resistance through 0x20-bit encoding: security via leet queries. In: Proceedings of the 15th ACM conference on Computer and Communications Security, 2008, S. 211–222

[Eas00] EASTLAKE 3RD, D.: Secret Key Establishment for DNS (TKEY RR). September 2000. – RFC 2930, IETF, Status: Proposed Standard, https://tools.ietf.org/html/rfc2930

[Eas06] EASTLAKE 3RD, D.: HMAC SHA TSIG Algorithm Identifiers. August 2006. – RFC 4635, IETF, Status: Proposed Standard, https: //tools.ietf.org/html/rfc4635

[EG11] EVANS, N. S. ; GROTHOFF, C.: R5N : Randomized Recursive Routing for Restricted-Route Networks. In: Network and System Security (NSS), 2011, S. 316–321

[HM09] HUBERT, A. ; MOOK, R. van: Measures for Making DNS More Resilient against Forged Answers. 2009. – RFC 5452, IETF, Status: Proposed Standard, https: //tools.ietf.org/html/rfc5452

References (2)

[HS13a] HERZBERG, A. ; SHULMAN, H.: DNSSEC: Security and availability challenges. In: IEEE Conference on Communications and Network Security (CNS), 2013, S. 365–366

[HS13b] HERZBERG, A. ; SHULMAN, H.: Fragmentation Considered Poisonous, or: One-domain-to-rule-them-all.org. In: IEEE Conference on Communications and Network Security (CNS), 2013, S. 224–232

[KMG12] KOLKMAN, O. ; MEKKING, W. ; GIEBEN, R.: DNSSEC Operational Practices, Version 2. 2012. – RFC 6781, IETF, Status: Proposed Standard, https://tools. ietf.org/html/rfc6781

[LSA08] LAURIE, B. ; SISSON, G. ; ARENDS, R. ; BLACKA, D.: DNS Security (DNSSEC) Hashed Authenticated Denial of Existence. March 2008. – RFC 5155, IETF, Status: Proposed Standard, https://tools.ietf. org/html/rfc5155

[Mic13] MICROSOFT: Peer Name Resolution Protocol (PNRP) Version 4.0. http://download.microsoft.com/download/9/5/E/95EF66AF- 9026-4BB0-A41D-A4F81802D92C/[MS-PNRP].pdf. Version: 2013. – Technical Document, Version 14

[VGE00] VIXIE, P. ; GUDMUNDSSON, O. ; EASTLAKE 3RD, D. ; WELLINGTON, B.: Secret Key Transaction Authentication for DNS (TSIG). May 2000. – RFC 2845, IETF, Status: Proposed Standard, https://tools.ietf. org/html/rfc2845

[WB13] WEILER, S. ; BLACKA, D.: Clarifications and Implementation Notes for DNS Security (DNSSEC). 2013. – RFC 6840, IETF, Status: Proposed Standard, https: //tools.ietf.org/html/rfc6840