IT-Security

Chapter 7: Security of Selected Classical Applications

E-Mail, DNS, Remote Login

Prof. Dr.-Ing. Ulrike Meyer

Overall Lecture Context

- **Many applications can be protected using TLS**
  - Most prominent example HTTP over TLS = HTTPs
  - Also FTPs, SIPS, SRTP ,…
- **Some distributed applications, however, cannot easily use TLS end-to-end**
  - **Email**: asynchronous, no handshake between sender and receiver possible
  - **DNS:** connectionless, runs on UDP, caching necessary for performance reasons
  - …
- **Secure versions of some applications have been developed in parallel to the first TLS version**
  - SSH: secures one of the oldest internet applications, namely **remote login**

Overview

Email Security

- Email Architecture
- Threats
- End-to-end protection
  - PGP and S/MIME
- Backbone protection
  - SMTPs
  - …

DNS Security

- DNS System
- Threats
- DNSSec
- DoT / DoH

Remote Login with SSH

- Primary use case
- Security services offered
- TCP payload protection

Overview

Email Security

- Email Architecture
- Threats
- End-to-end protection
  - PGP and S/MIME
- Backbone protection
  - SMTPs
  - …

DNS Security

- DNS System
- Threats
- DNSSec
- DoT / DoH

Remote Login with SSH

- Primary use case
- Security services offered
- TCP payload protection

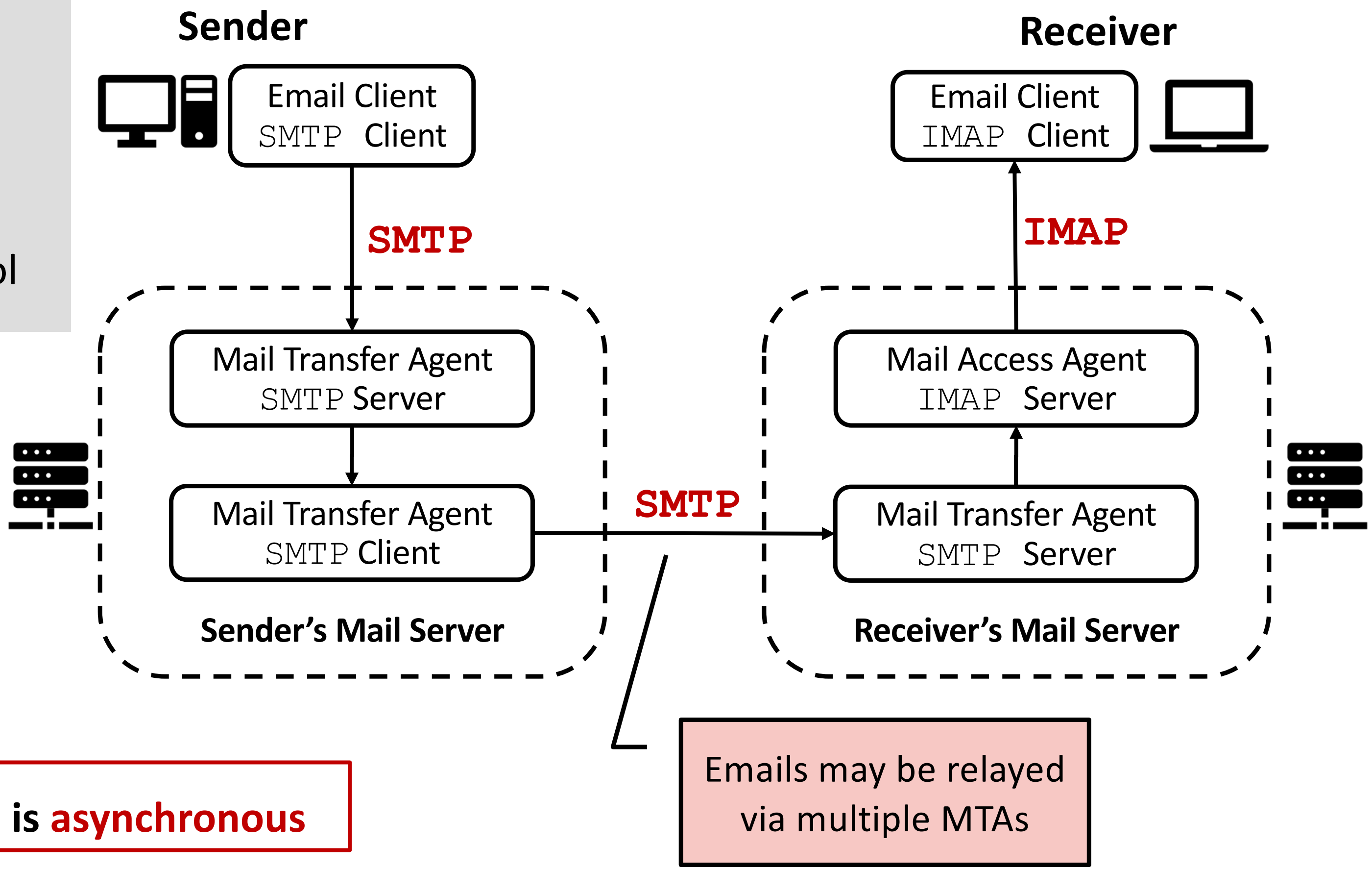
Classical Email Architecture (Simplified)

SMTP:

Simple Mail Transfer Protocol

IMAP:

Internet Message Access Protocol



Note: sending /receiving email is asynchronous

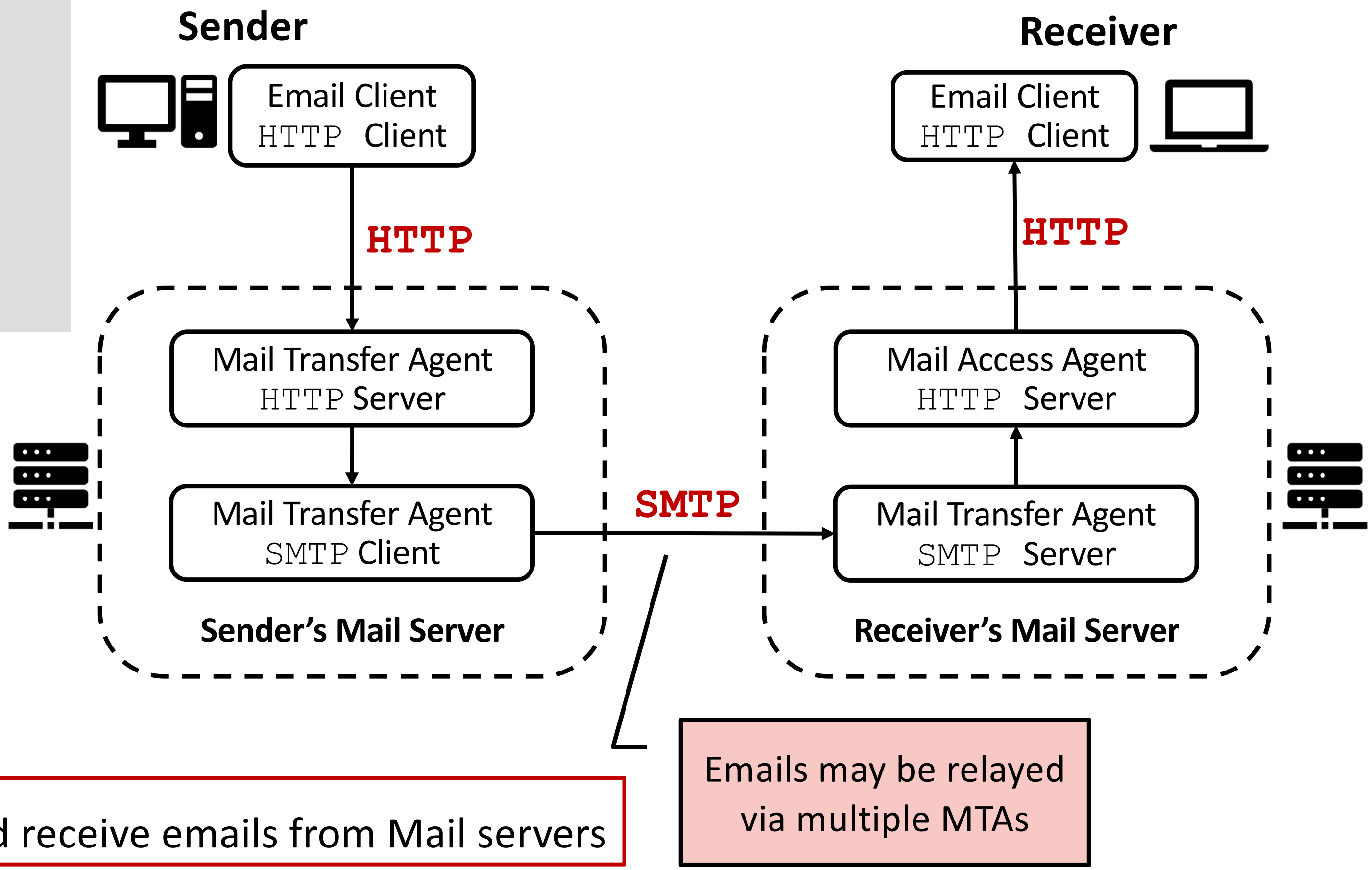
Alternative: Web Email Architecture (Simplified)

SMTP:

Simple Mail Transfer Protocol

HTTP:

Hypertext Transfer Protocol

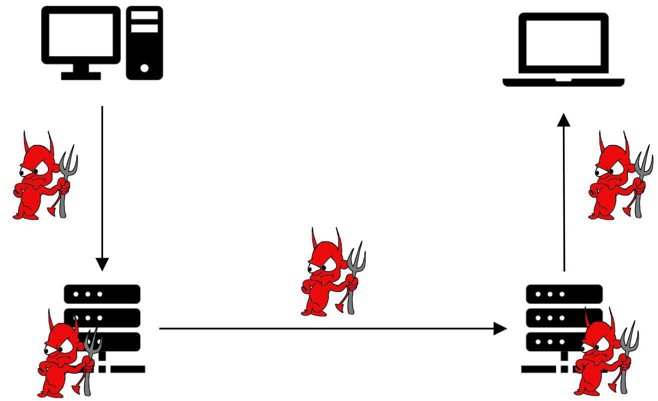


HTTP used to send emails to and receive emails from Mail servers

Email Threats

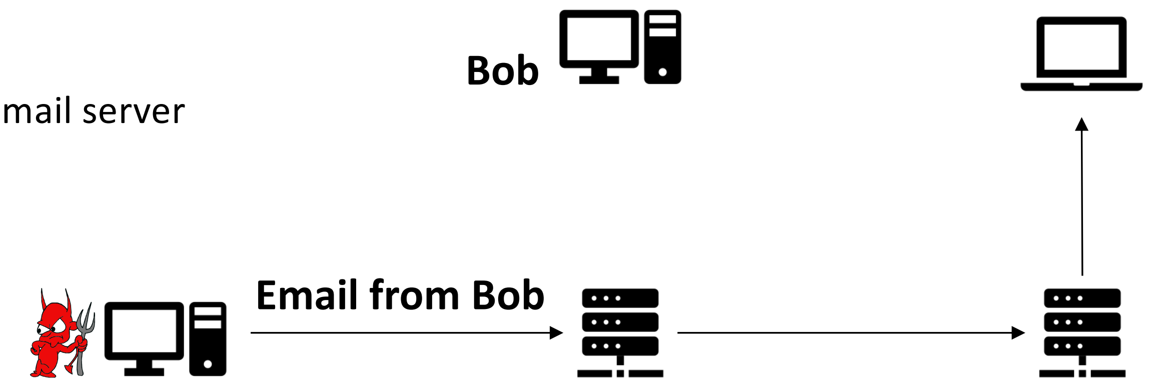
Eavesdropping and Manipulation

- During transfer
  - Between email clients and mail servers, between mail servers
- On storage at mail server
  - Emails stored in cleartext



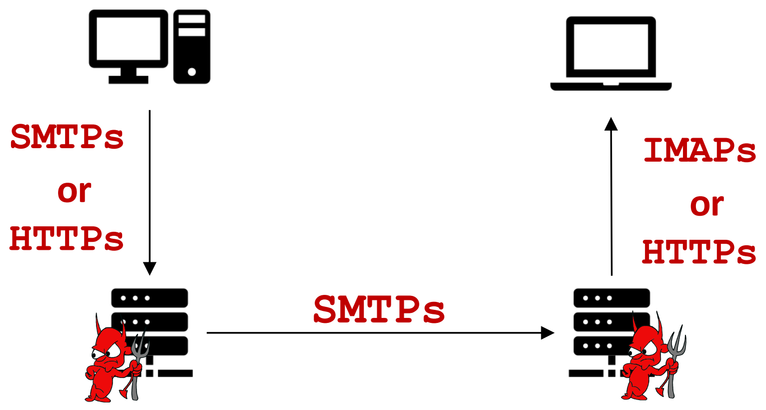
Email Spoofing

- Attacker submits an email to some mail server
- Claims the email is from Bob



Protecting Emails with TLS

- **TLS allows us to protect TCP connections**
  - `SMTPs: SMTP` over `TLS`
    - protects email transfer from sender to email server
    - protects email transfer between email servers
  - `IMAPs: IMAP` over `TLS`
    - protects email transfer from email server to receiver
  - Alternatively: `HTTPs HTTP` over `TLS`
    - protects email transfer from sender to email server
    - protects email transfer from email server to receiver



- **Hop-by-hop protection of confidentiality and integrity**
  - No non-repudiation

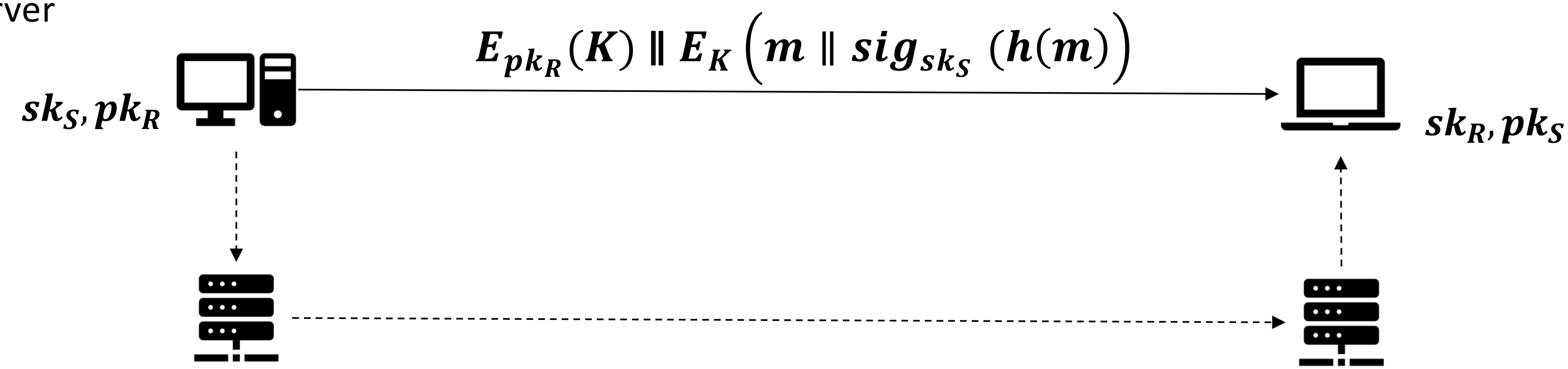
End-to-end protection with TLS not possible

- **Emails still stored in the clear on mail server**

End-to-End Protection

Approach used by S/MIME and PGP

- Signs hash of message m with own private key $\boldsymbol{sk_S}$
- Sender generates symmetric key K
- Encrypts mail and signature with K
- Encrypts K with receiver's public key $\boldsymbol{pk_R}$
- Sends encrypted message and encrypted key as mail to its mail server



Threats covered

- Eavesdropping – symmetric encryption
- Manipulation – digital signature
- Repudiation – digital signature

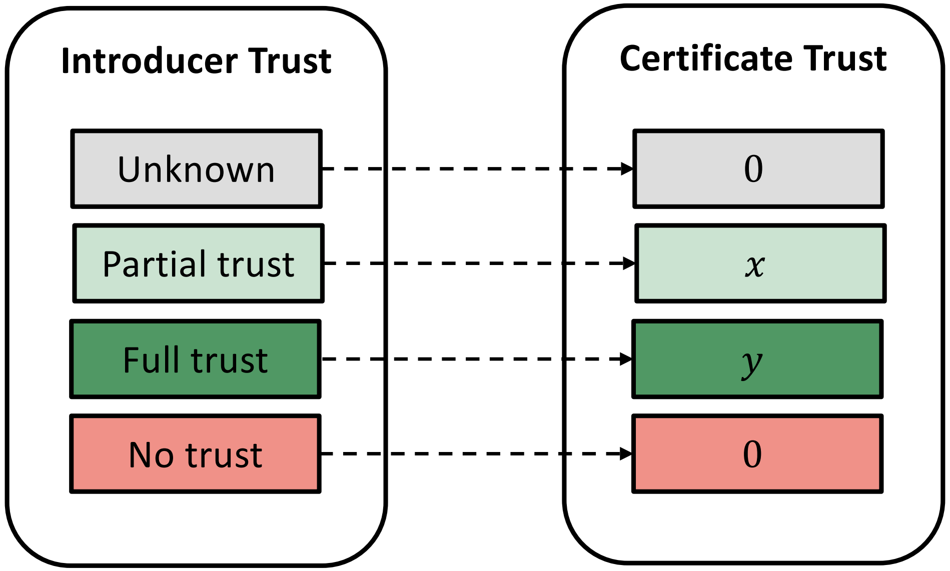
Main Conceptional Difference between S/MIME and PGP

Distribution of public keys

- S/MIME: certificates signed by CAs, Trusted CAs configured in email client
- PGP: Web of trust
  - PGP users sign certificates for other PGP users
  - Each user decides which keys to trust

PGP Web of Trust

- **Each PGP user**
  - assigns **introducer trust level** to other users
  - assigns **certificate trust level** $0 \leq w \leq 1$ to each introducer trust level



User	Introducer Trust
Clare	Partial trust
Dave	No trust
Tom	Partial trust
Fred	Full trust

Certificate	Certificate Trust level
$Cert(pk_{Bob})_{Clare}$	x
$Cert(pk_{Bob})_{Dave}$	0
$Cert(pk_{Bob})_{Tom}$	x

Key Legitimacy

Key legitimacy is computed from certificate trust levels

- Let N_x (N_y) be the number of certificates of certificate trust value x (y)
- Then the key legitimacy is computed by

$$\text{key legitimacy} = \begin{cases} 1 & if \quad x \cdot N_x + y \cdot N_y \geq 1 \\ 0 & if \quad x \cdot N_x + y \cdot N_y < 1 \end{cases}$$

Example with $x = ½$ and $y = 1$

User	Introducer Trust
Clare	Partial trust
Dave	No trust
Tom	Partial trust
Fred	Full trust

Certificate	Certificate Trust level
$Cert(pk_{Bob})_{Clare}$	$1/2$
$Cert(pk_{Bob})_{Dave}$	0
$Cert(pk_{Bob})_{Tom}$	$1/2$
$Cert(pk_{Ted})_{Dave}$	0
$Cert(pk_{Alf})_{Fred}$	1

public key	Key legitimacy
pk_{Bob}	1
pk_{Ted}	0
pk_{Alf}	1

Threats covered

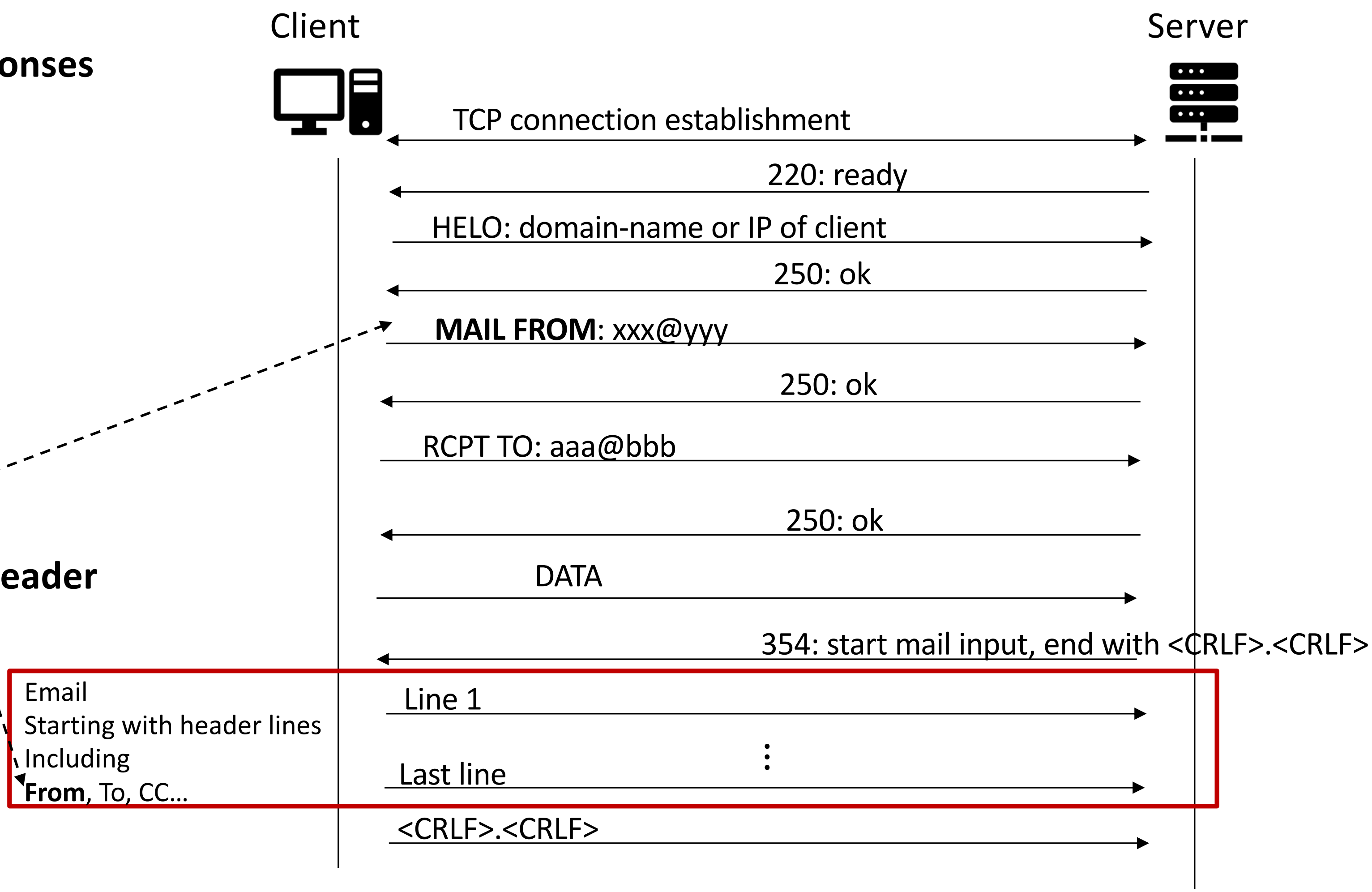
Hop-by-hop protection with TLS	End-to-end with S/MIME or PGP
Eavesdropping on transfer – Symmetric encryption	Eavesdropping on transfer and in storage – Symmetric encryption
Manipulation on transfer – MACs	Integrity protection on transfer and in storage – Digital Signature
	Non-repudiation – Digital Signature

But: mail servers and mail clients still accept unprotected messages

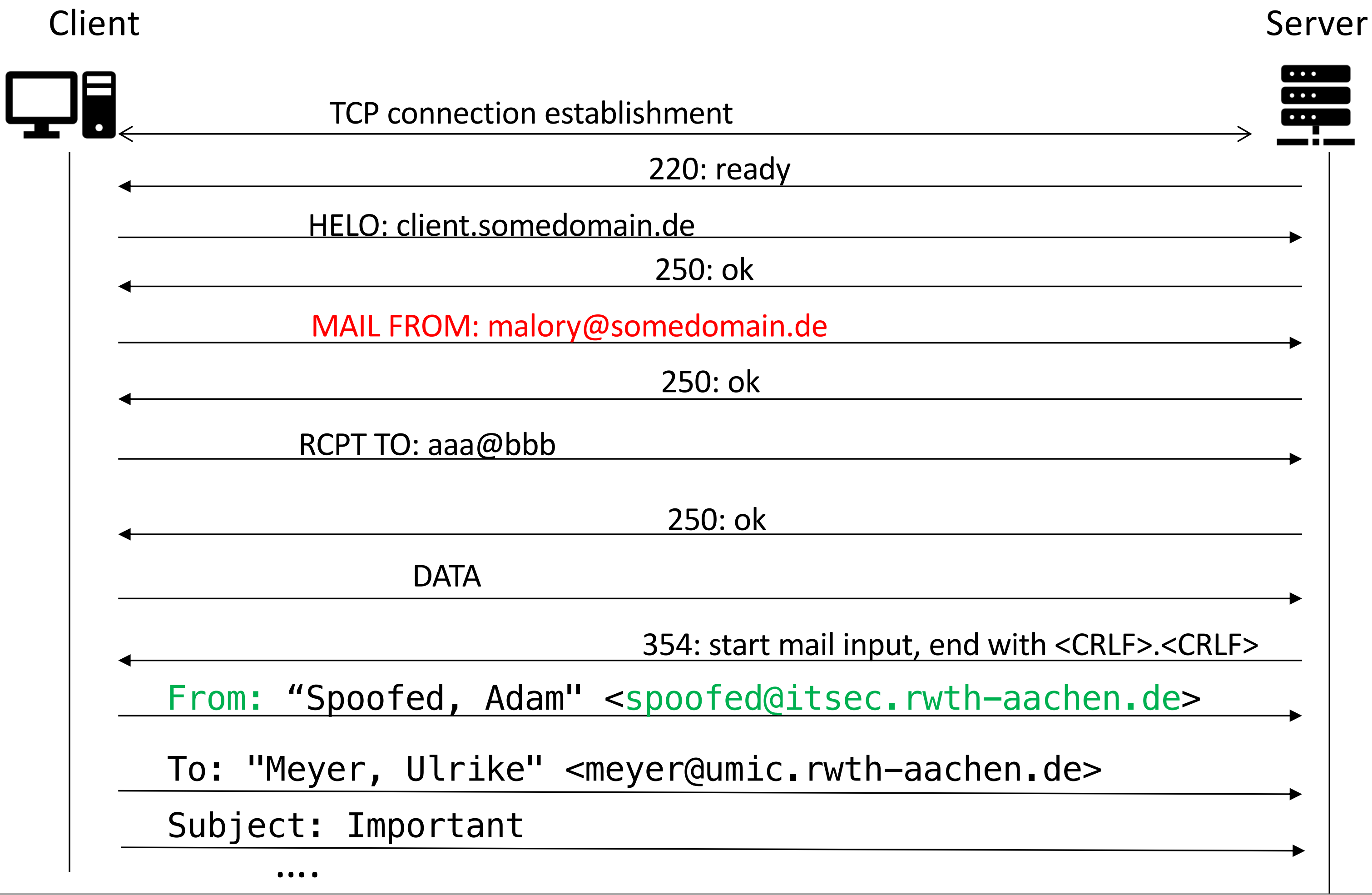
- Email spoofing still possible und extensively used e.g. in phishing attacks

Typical SMTP Exchange between Client and Server

- **SMTP commands / responses**
- **Email message**
  - Email header lines
    - From header line
    - To header line
    - …
  - Email Body
- **MAIL FROM and From header line may differ**



Example: Spoofed From Header Line



Further Protocols between Mail Servers

Domain Key Identified Mail

- Mail server signs mail header lines and body
- Thereby "authenticates" from header line
- Signature checked by receiving mail server
- Allows receiving mail server to discard unsigned messages with from email address of domain that is know to be signing

Sender Policy Framework

- Allows to specify hosts that are allowed to send mail on behalf of the domain
  - In HELO and MAIL FROM SMTP commands
- Emails from other SMTP clients trying to sent email on behalf of the domain can than be blocked

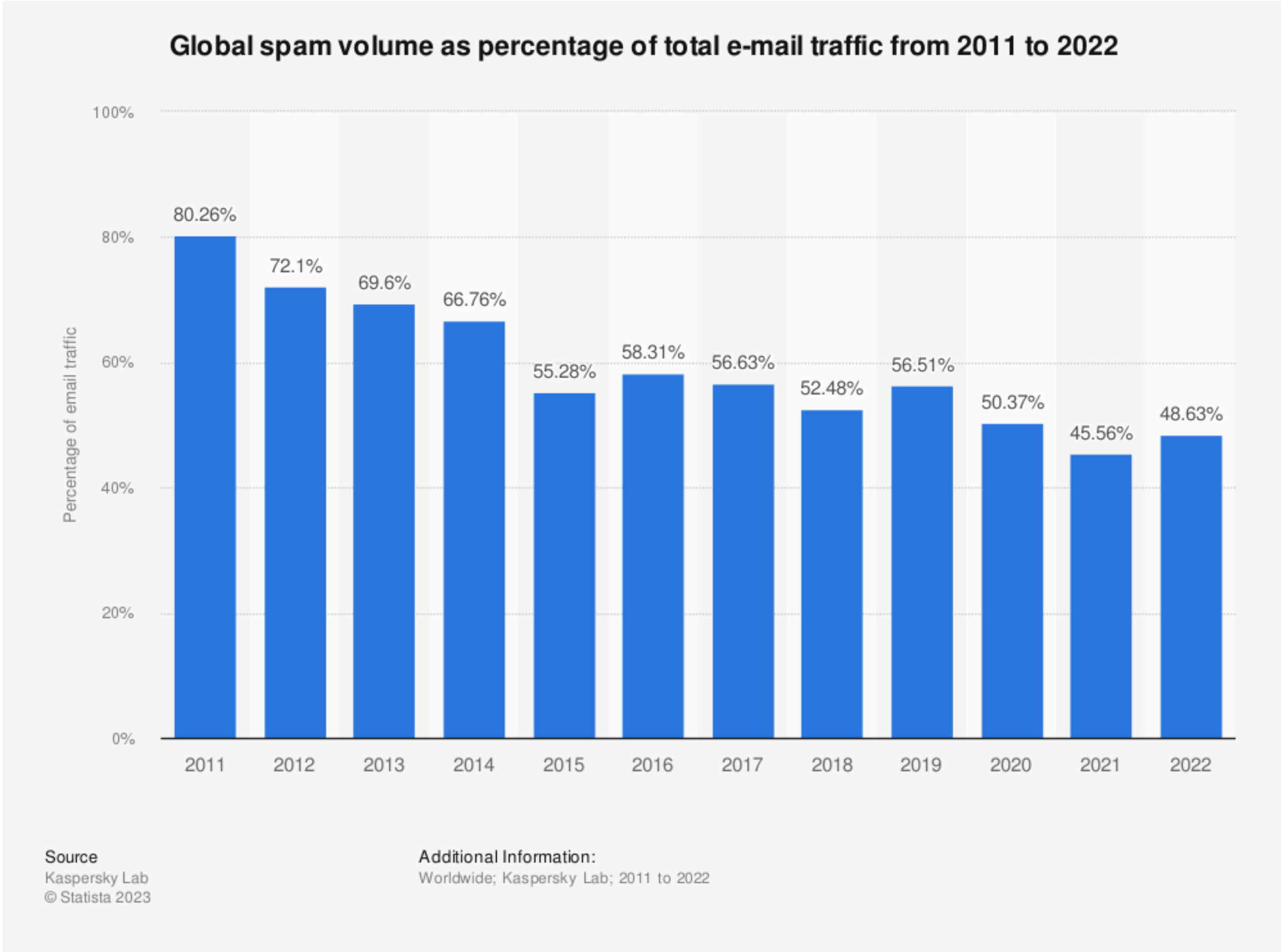
DMARC

- Complements SPF and DKIM by a DNS-based mechanism to distribute policies on
  - How emails claiming to come from a domain are to be handled
  - How to receive reports on domain abuse

Threat to Availability: SPAM

- **SPAM stands for “SPiced hAM”**
  - Any unsolicited commercial email is SPAM

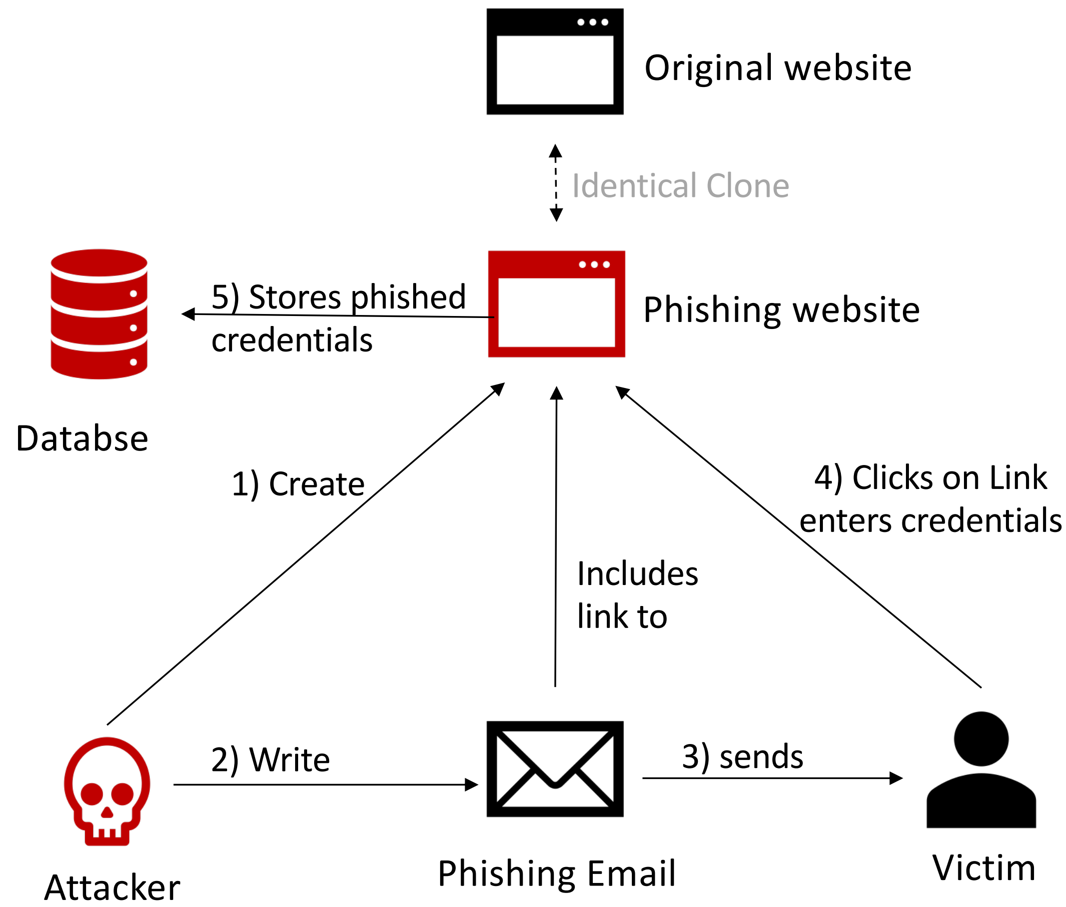
- **Sent by attackers by**
  - Directly connecting to recipient’s email server
  - Using open email relays
  - Using malware-infected client machines



https://www.youtube.com/watch?v=duFierM1yDg

Email Spoofing in Classical Phishing Attacks

- **In a phishing attack and attacker tries to**
  - lure a user into revealing private information
- **Classical attack path**
  - Attacker sends phishing email to victim, **often with spoofed sender address** to lure user into trusting it
  - Email includes link to phishing website
  - Phishing website is a clone of an original website
  - User enters private information into phishing website believing it's the original website
  - Phishing websites sends private information to a database controlled by the attacker
- **What private information?**
  - E.g. username / password, credit card number,…



Overview

Email Security

- Email Architecture
- Threats
- End-to-end protection
  - PGP and S/MIME
- Backbone protection
  - SMTPs
  - …

DNS Security

- DNS System
- Threats
- DNSSec
- DoT / DoH

Remote Login with SSH

- Primary use case
- Security services offered
- TCP payload protection

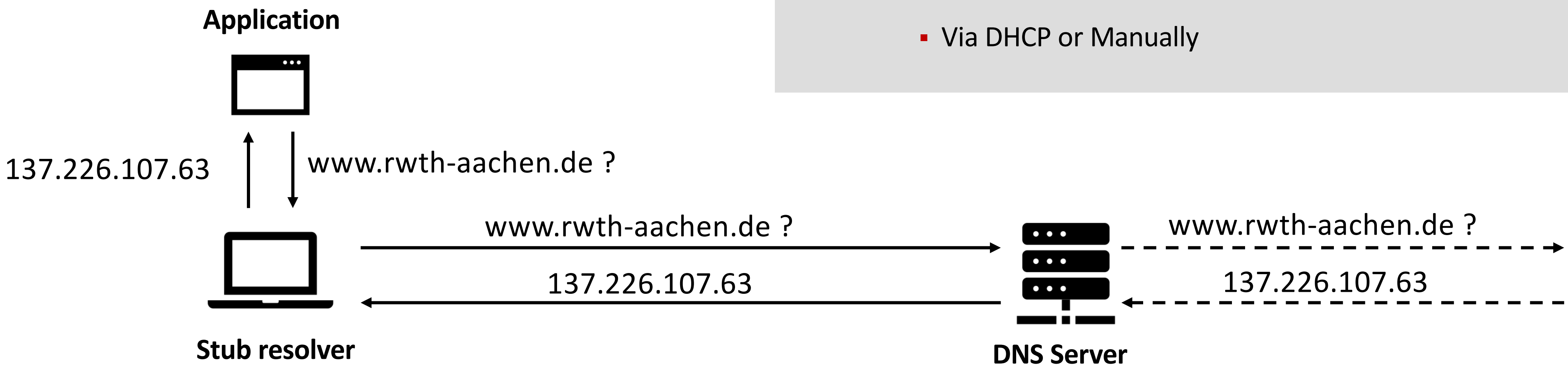
DNS System – Overview

Main purpose:

Map **domain names** to **IP addresses**

One of the oldest Internet applications (1987)

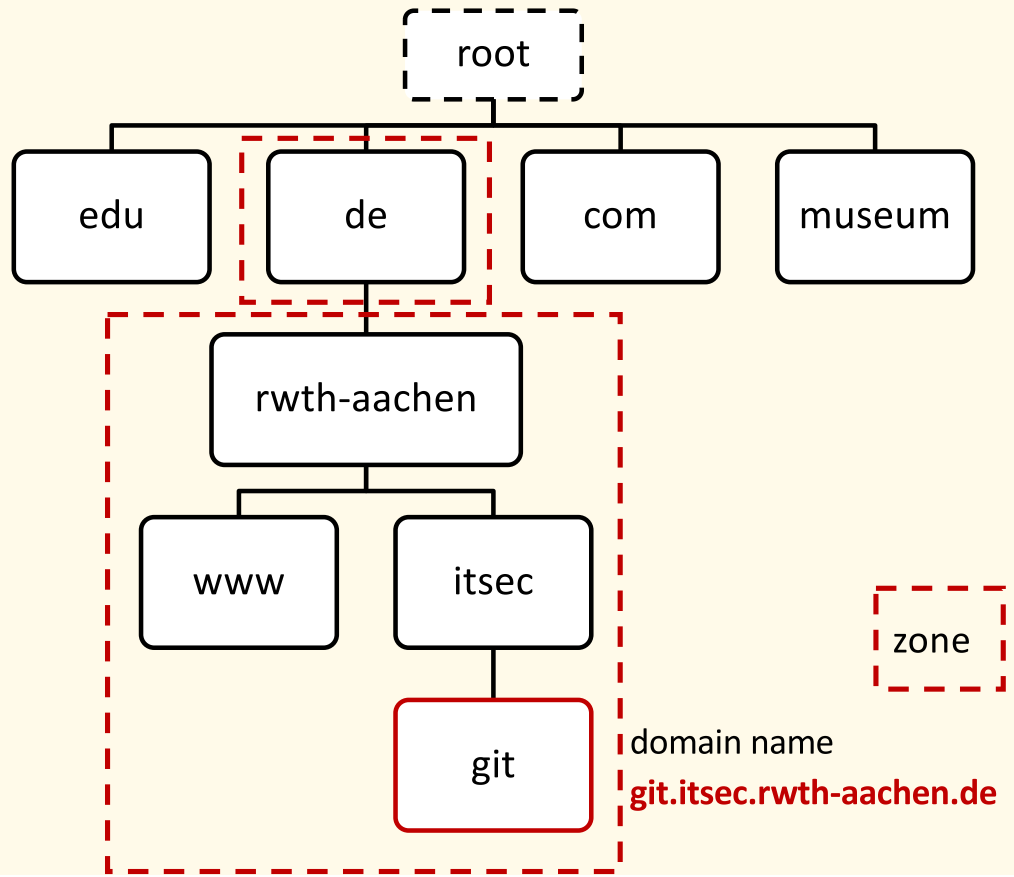
- Consists of thousands of DNS servers
- DNS servers can be queried by applications
  - via DNS client in the operating system: stub resolver
- Each host has a DNS server preconfigured
  - Via DHCP or Manually



DNS System – Domain Name Space

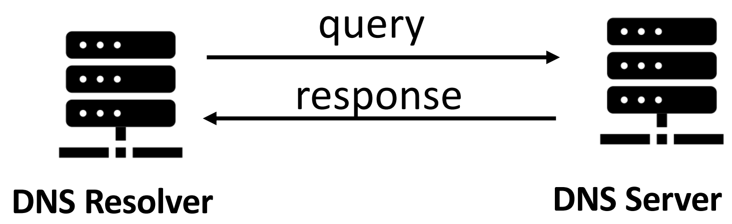
- **Tree-structured name space**
  - starting from unnamed root domain
- **Over 1400 top-level domains: TLDs**
  - generic: e.g., .org, .edu, .mil, .com,…
  - country: e.g., .de, .uk, .es, .cn, …
  - new TLDs: e.g., .tourism, .museum,…
  - Internationalized domain name, e.g. .mockba
- **Leaf domains may refer to single hosts or a collection of hosts**
- **Zone**
  - Connected part of the domain name space
  - child does not need to belong to zone of parent

Some part of the domain name tree



DNS Queries and Responses

- A DNS resolver sends a DNS query to a DNS Server
- A DNS server answers with a DNS response
- Queries and responses use the same format



Query ID: 16 bit, same in query and response	**Flags**
# Questions	**# RRs in Answer Section**
# RRs in Authority Section	**# RRs in Additional Information Section**
Question(s): domain name and type of answer desired	
Answer section: RRs answering the question	
Authority section: RRs of name servers responsible for domain name in answer	
Additional information section: additional RRs, e.g., IP address of name servers	

In practice only 1 question per query is supported

`dig` **www.tu-darmstadt.de**

DNS clients are called resolvers!

RR stands for resource record

Resource records

- **DNS distributes information on domain names as RRs**
- **Structure of RRs**

domain name	time-to-live	class	type	value

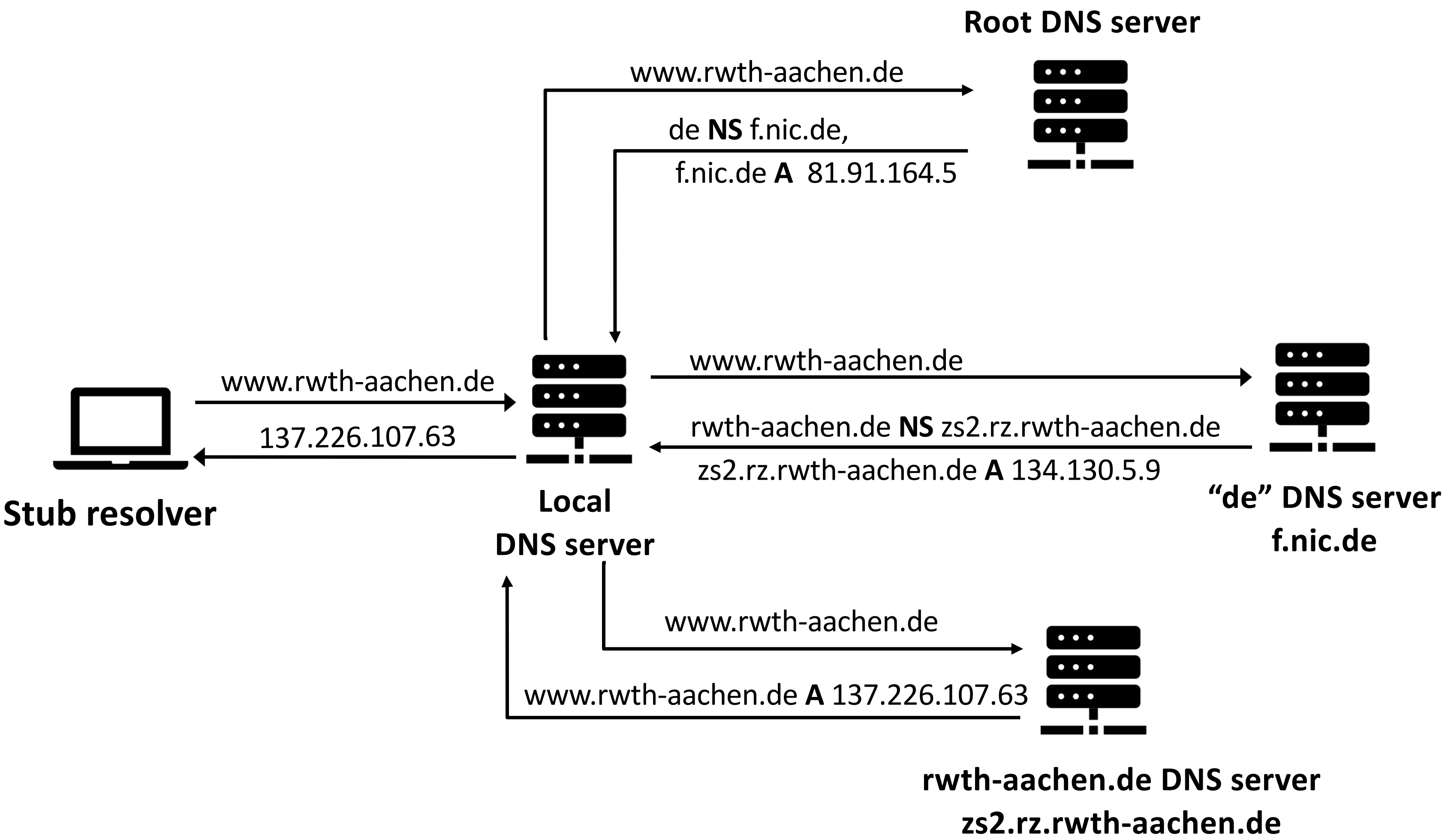
- **domain name**
  - to which the RR applies
- **TTL in seconds**
  - indicates how long RR should be cached
- **class:** IN for Internet

Type	Meaning	Value
A	IPv4 address of a host	32 bit
AAAA	IPv6 address of a host	128 bit
MX	Mail exchange	Domain name of mail server accepting email for this domain
NS	Name Server	Domain name of an authoritative name server for this domain
CNAME	Canonical Name	Maps the domain name (alias) to an other domain name (canonical name)
PTR	Pointer	Used mainly for reverse lookups
SOA	Start of authority	Administrative information on a zone

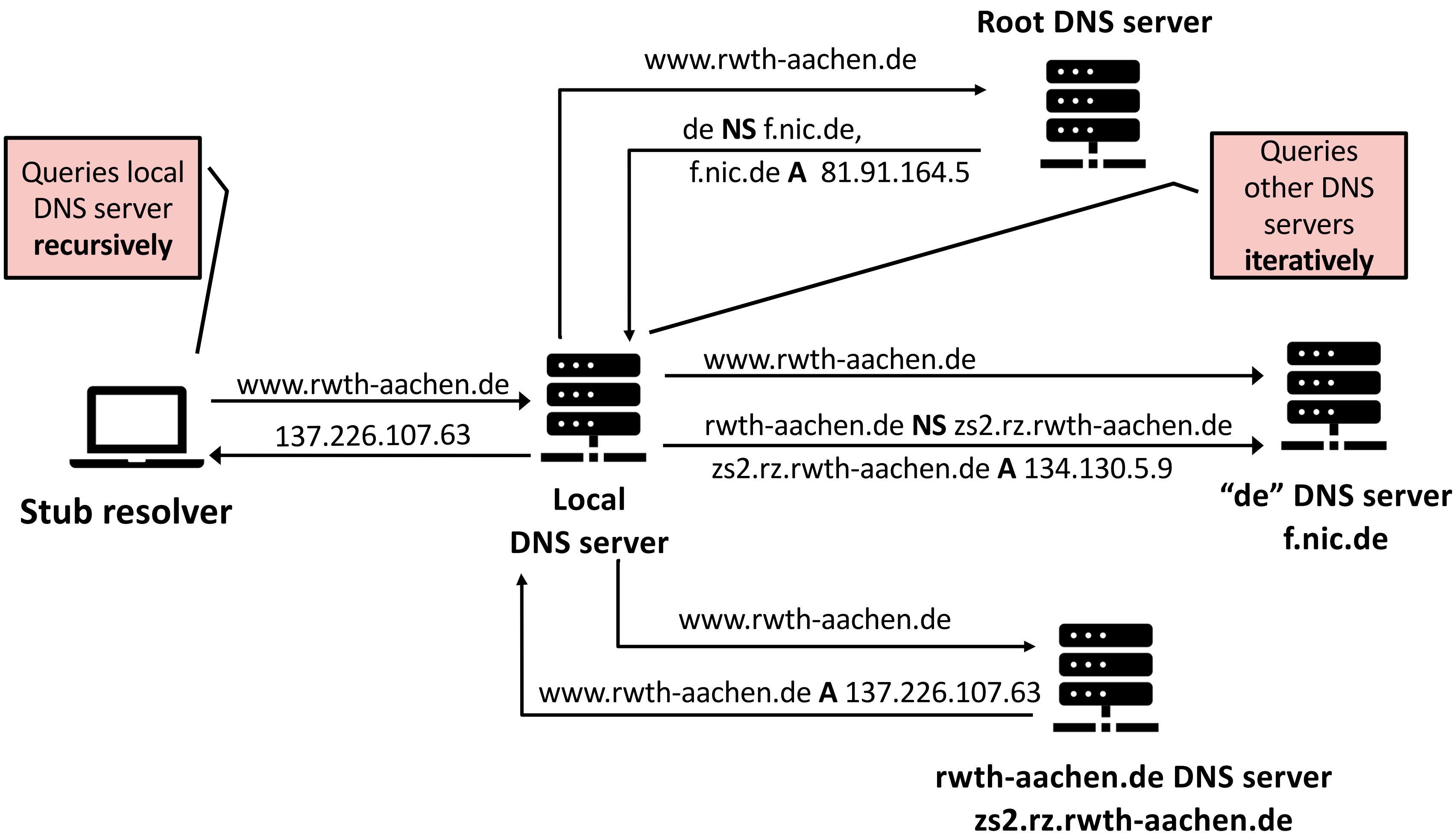
Example

```
MacBook-Pro:~ uli$ dig www.tu-darmstadt.de

; <<>> DiG 9.10.6 <<>> www.tu-darmstadt.de
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 14179
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 2, AUTHORITY: 2, ADDITIONAL: 5

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;www.tu-darmstadt.de.           IN      A

;; ANSWER SECTION:
www.tu-darmstadt.de.    3444    IN      CNAME   cms-sip02.hrz.tu-darmstadt.de.
cms-sip02.hrz.tu-darmstadt.de. 68489 IN A       130.83.47.181

;; AUTHORITY SECTION:
hrz.tu-darmstadt.de.    18179   IN      NS      ans1.net.hrz.tu-darmstadt.de.
hrz.tu-darmstadt.de.    18179   IN      NS      ans2.net.hrz.tu-darmstadt.de.

;; ADDITIONAL SECTION:
ans1.net.hrz.tu-darmstadt.de. 47911 IN  A       130.83.22.61
ans2.net.hrz.tu-darmstadt.de. 47911 IN  A       130.83.56.61
ans1.net.hrz.tu-darmstadt.de. 47911 IN  AAAA    2001:41b8:83f:22::61
ans2.net.hrz.tu-darmstadt.de. 47911 IN  AAAA    2001:41b8:83f:56::61

;; Query time: 56 msec
;; SERVER: 134.130.4.1#53(134.130.4.1)
;; WHEN: Thu Jun 01 12:16:29 CEST 2023
;; MSG SIZE  rcvd: 222
```

Example: Typical Address Resolution

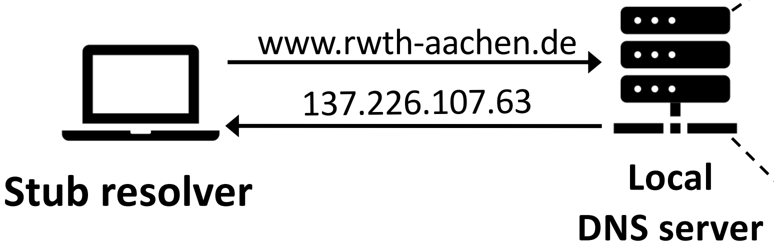


Example: Typical Address Resolution



Caching at DNS Servers

DNS servers cache all RRs they learn from responses for as TTL seconds



Cache of Local DNS server after receiving all responses

name	TTL	Class	Type	Value
www.rwth-aachen.de	600	IN	A	137.226.107.63
zs2.rz.rwth-aachen.de	600	IN	A	134.130.5.9
rwth-aachen.de	86400	IN	NS	zs2.rz.rwth-aachen.de
f.nic.de	600	IN	A	81.91.164.5
de	86400	IN	NS	f.nic.de

Caching accelerates future queries

- same queries as well as queries that may reuse part of the information obtained

DNS Threats Overview

- **Threats to availability**
  - Flood DNS server with fake queries or responses
  - Thereby make it unresponsive for legitimate queries

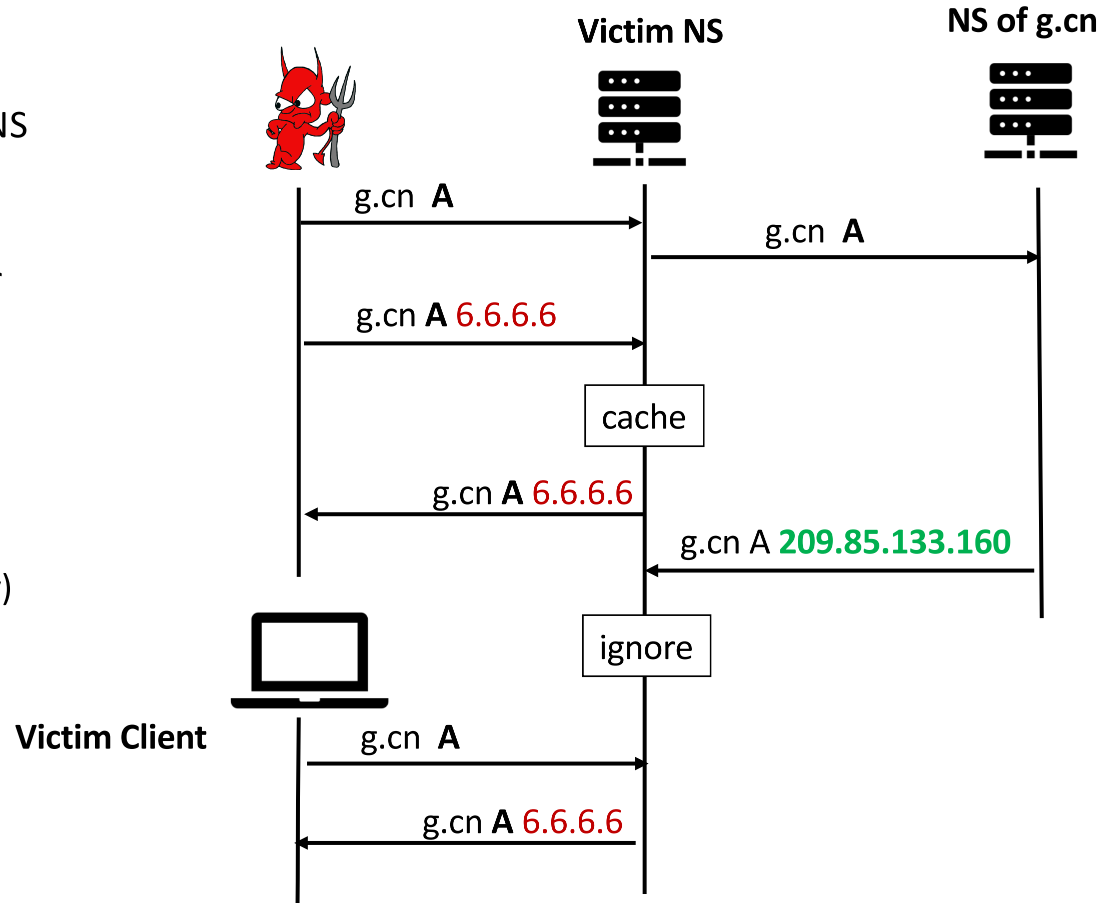
Chapter 8

- **Threats to integrity such as**
  - Provide incorrect RRs to resolvers
    - E.g., by making DNS servers cache fake RRs: **Cache Poisoning Attacks**
    - E.g., by making client machines connect to malicious DNS servers
- **Threats to confidentiality**
  - DNS queries are unencrypted
    - Anyone eavesdropping between client and local DNS server learns queries
    - Local DNS server learns queries

Example Attack Threatening Integrity: Cache Poisoning

Idea:

- Make victim DNS server query another DNS server for RR to be faked
- Send a fake response to victim DNS server
- Only successful if attacker can
  - spoof IP address of queried DNS server
  - guess correct query ID
  - (guess correct source port of victim's query)
  - be faster than real name server



DNSSec

- **Goal of DNSSec**
  - Protect authenticity of resource records in an end-to-end fashion
  - Enable distribution of authentic copies of public keys
  - Still allow for caching
  - Still allow DNS to run on top of UDP

- **Using TLS to protect authenticity of DNS RRs**
  - Would require DNS to run on top of TCP
  - Unnecesssary overhead
    - for one single round of query and response need TCP three-way handshake and TLS handshake
  - Would make caching impossible as direct connection to authoritative name server required

DNSSec New RRs and Keys

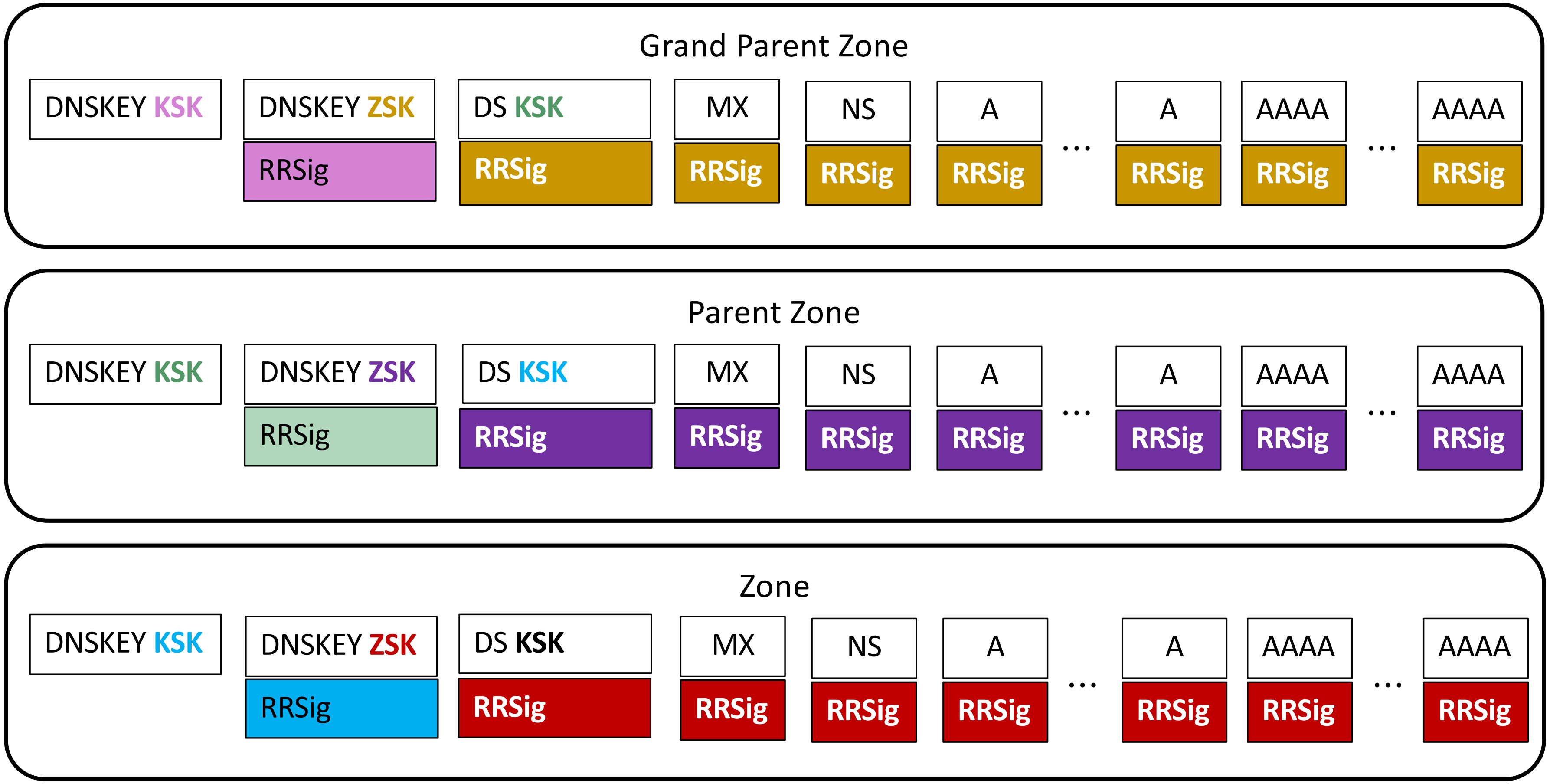
New RR types introduced by DNSSec

- **RRSig RR**: used to carry signatures on a specific other RR referred to by domain name and type
- **DNSKEY RR**: used to carry public ZSKs and KSK of zones
- **DS RR:** (Delegation Signature RR): used to carry a hash of a KSK
- **NSEC RR:** used to help to authenticate negative DNS responses

Two types of keys used in DNSSec

- **Zone Signing Key ZSK**
  - Used to sign resource records of a zone
  - Can be changed without involving parent zone
- **Key Signing Key KSK**
  - Used to sign DNSKEY RRs containing a ZSK
  - Distributed in DNSKEY RRs as well
  - Requires a DS RR in the parent domain and an RRsig RR that carries a signature on the DS RR signed with the zone's ZSK

Zone File with DNSSec Resource Records



New RR Types (1)

- **DNSKEY RR contains**
  - a public key of a zone
  - Key flag that indicates if the key is a KSK (key flag 257) or a ZSK (key flag 256)
  - algorithm field that indicates for which algorithm the key can be used

```
de.    4408    IN    DNSKEY    257 3 8
AwEAAbntyidABgdzt4jx+CVx8RgxEcJYdBFoihl3Ay87saAJsJXCVo6X
yGJWDHlgNFJrVzKL6ePIQ2vtnb/R4opICz1TTLB92MFiWJs6gKlBBHtx
z1+etiRAAWLgakExShzkmWmrFciMpTDIjNMEclpl4diuqgnnqiAtO4jw 97t/C69H ; key-id =39227
```

New RR Types (2)

- **DS RR (Delegation Signature RR) contains**
  - hash of a KSK, indication of hash algorithm used
  - A key tag that points to the KSK, an algorithm field that indicates the algo for which KSK can be used, algo identifier of hash algo used

```
de.    82597    IN    DS    39227 8 2
AAB73083B9EF70E4A5E94769A418AC12E887FC3C0875EF206C3451DC 40B6C4FA
```

New RR Types (3)

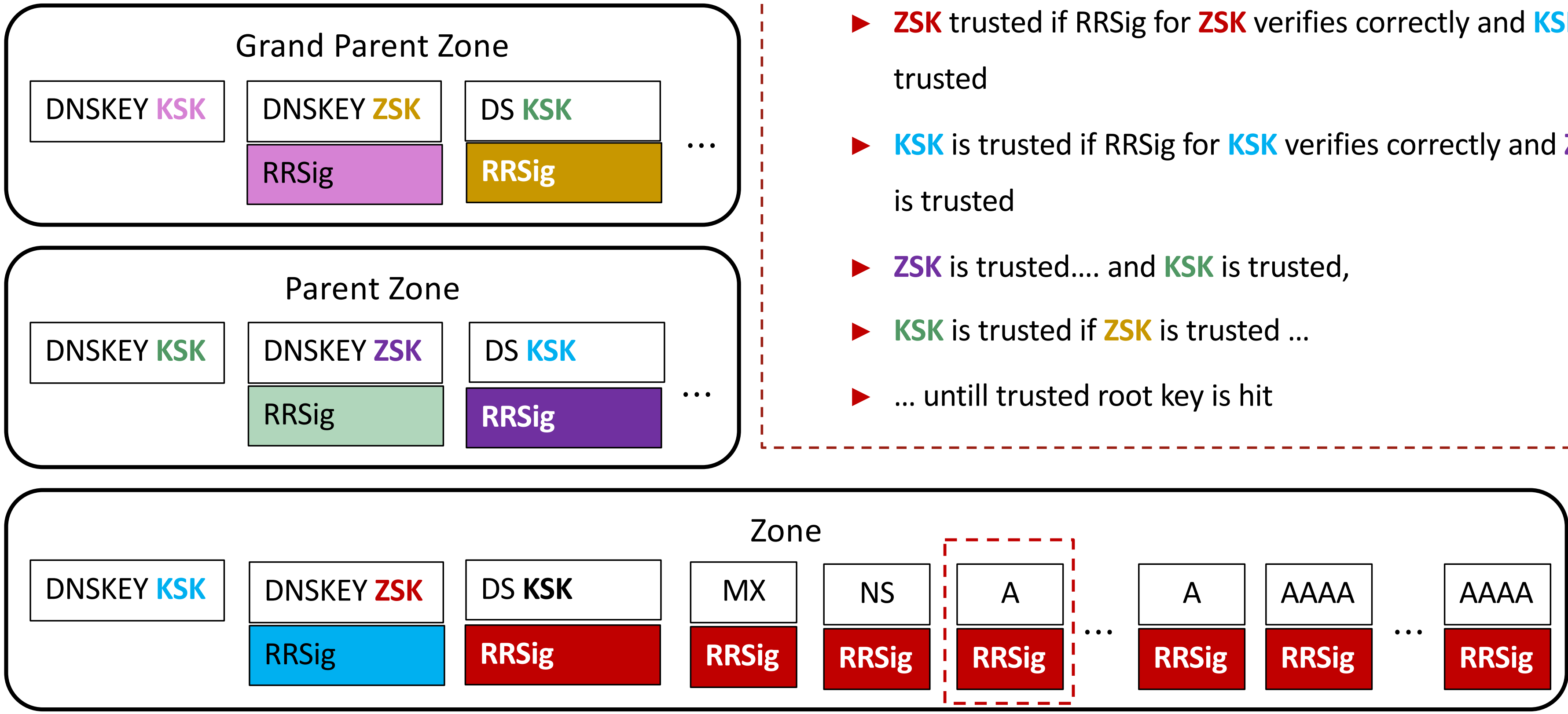
- **RRSIG RR contains**
  - a signature on a hash
  - A type covered field that indicates the type of RR covered by the signature
  - Algorithm field, that indicates the algorithm used
  - A key tag that points to the key used to generates the signature
  - Indication of the validity period of this signature, the signer's name
  - Original TTL of signed RR

de. 5320 IN RRSIG DNSKEY 8 1 7200 20170601120000 20170511120000 39227 de.

HMN5YPRBCtkSxWlR8/eW/3Kqy5AyVSbq/Zbx4fRKbSawf+v7rXHEqXnx
CFS4DlaDWdOs0bOWLfMH748NW8YA1ZT6DSFKecuEkTHzLL4IbFzZcBgG
KuJkp7wmiaamuW2PPGhutuqUcJ3CPy357OpuCJcT0qJh5nkkEURtZGj+
gZCFNWjKIBvnLwDMGkFtdHsiW1DBUJA41KeF3Cbsvk9iJZnkxB+k6mlr
a6A/G7+2fk6JuG4w7TBTfbBAa12VOHMzydwg2IMyweQOu3LssFR/WMF6
h8k7SaOJhwE6WjhVuHhAUxsvnN7TVPF9Ihb2FXMj1j0ckGwBZ4Y/SL7X /JLXhQ==

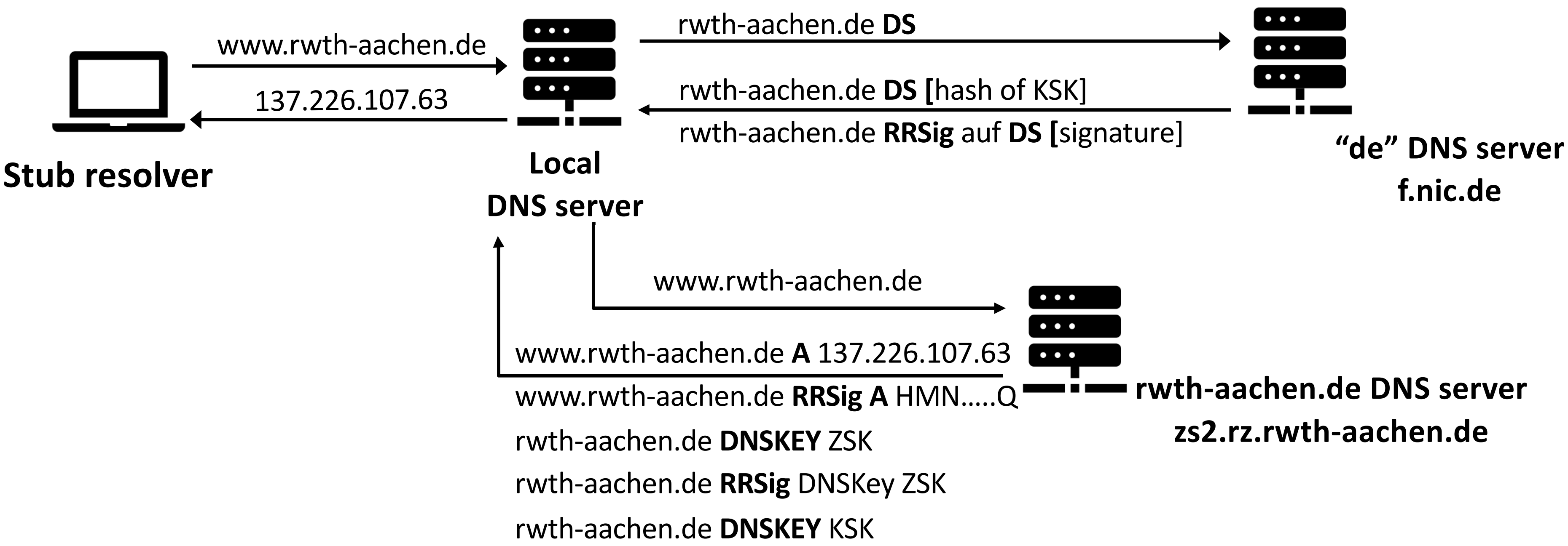
DNSSec Signature Validation

- RR of zone trusted to be authentic if
- RRSig on RR with **ZSK** verifies correctly and **ZSK** trusted
- **ZSK** trusted if RRSig for **ZSK** verifies correctly and **KSK** is trusted
- **KSK** is trusted if RRSig for **KSK** verifies correctly and **ZSK** is trusted
- **ZSK** is trusted.... and **KSK** is trusted,
- **KSK** is trusted if **ZSK** is trusted ...
- ... untill trusted root key is hit



Example: Typical Address Resolution with DNSSec

Signatures are checked at the local DNS server; stub resolvers do often not support DNSSec

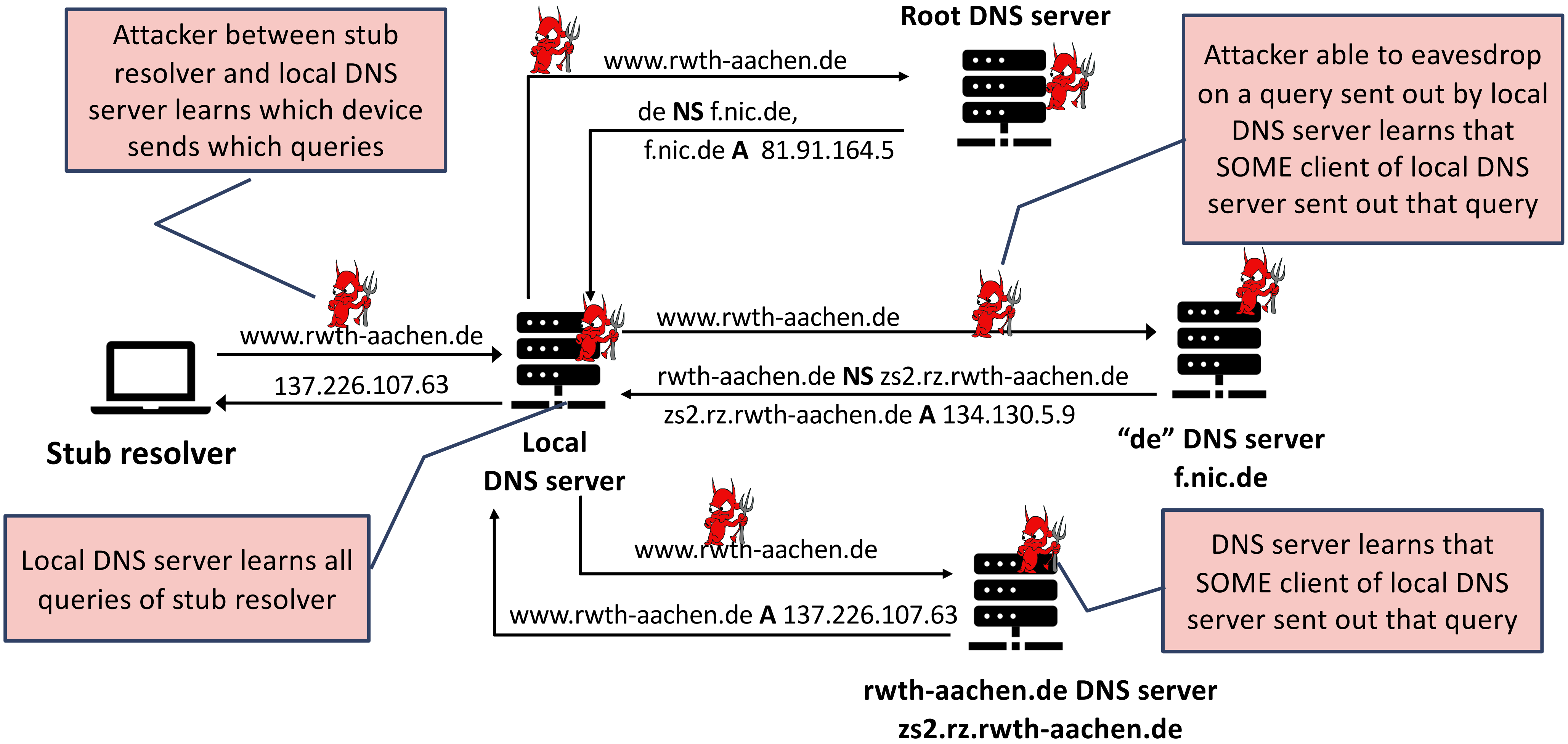


The DS record may also be cached by RWTH's name server and directly provided

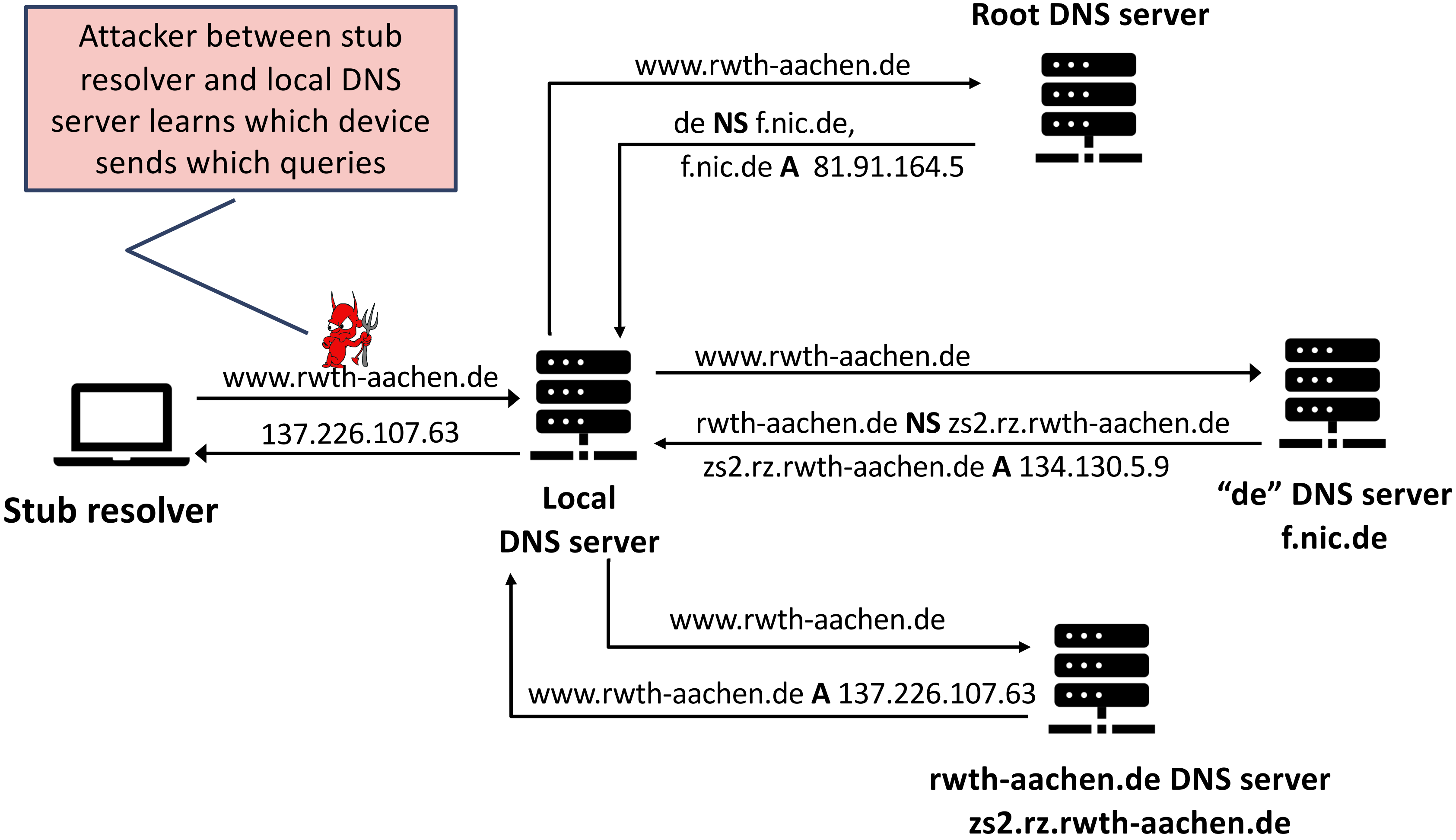
Confidentiality of DNS Queries

- **DNSSec only protects authenticity of DNS queries**
  - Implicitly assumes that DNS queries are not confidential
    - DNS responses only contain public information!
- **However, what someone queries and how reveals**
  - The websites he or she is interested in
  - Operating system used by the client
    - OS specific queries such as DNS queries to windows domains for updating
  - The anti-virus program you use
  - The productivity programs you use
    - Acrobat reader, all MS-office products etc. make specific DNS queries
  - …

Example Threats against Confidentiality



Threat countered by DNS over TLS (DoT)



Overview

Email Security

- Email Architecture
- Threats
- End-to-end protection
  - PGP and S/MIME
- Backbone protection
  - SMTPs
  - …

DNS Security

- DNS System
- Threats
- DNSSec
- DoT / DoH

Remote Login with SSH

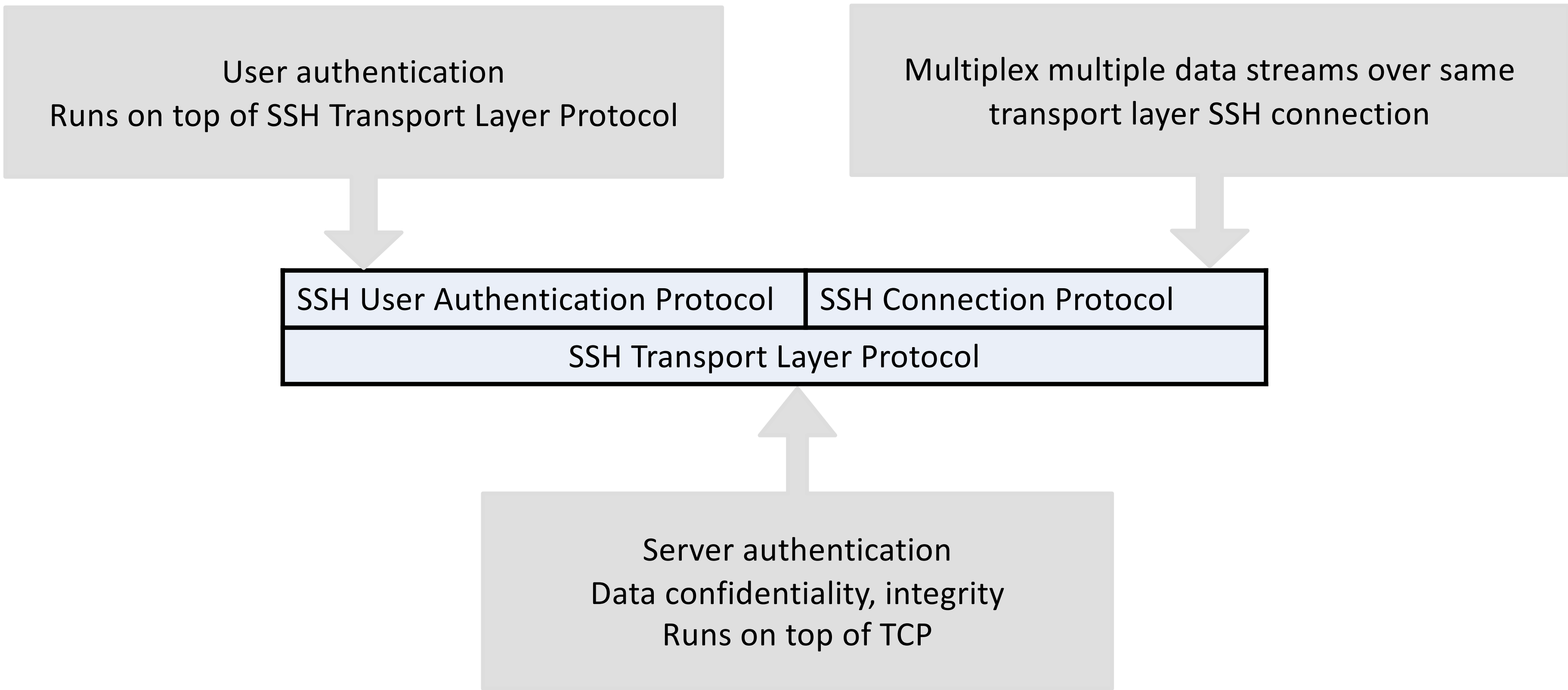
- Primary use case
- Security services offered
- TCP payload protection

SSH Main use Case and Operation

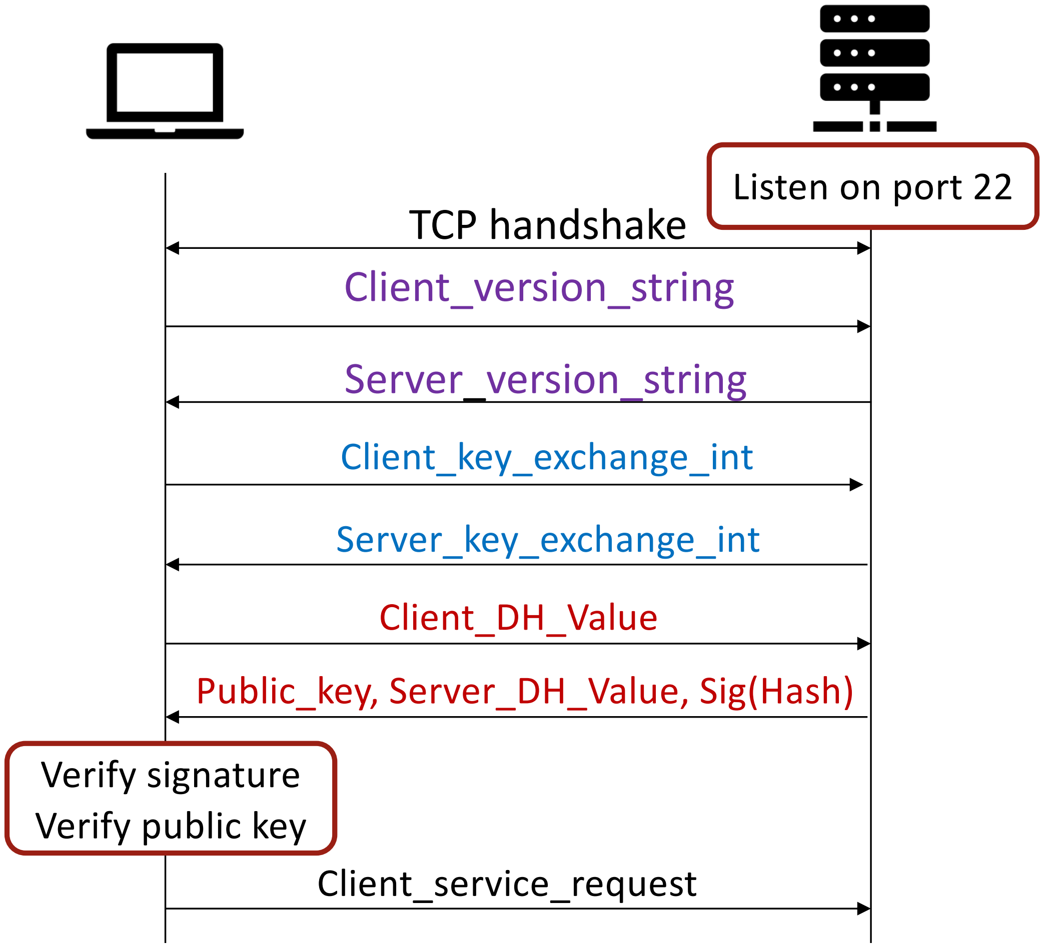
- **Originally designed to secure network services over an insecure network**
  - Specifically: remote login onto a machine for administrative purposes or in order to run applications on a remote host
- **Developed at the same time as the first SSL version**
  - Some overlap between SSL and TLS w.r.t use cases supported
  - But original main use cases quite different
- **Operates on the application layer**
  - Does not make use of TLS

Remote Login with SSH
TCP
IP
Data Link
Physical

SSH Protocol Family

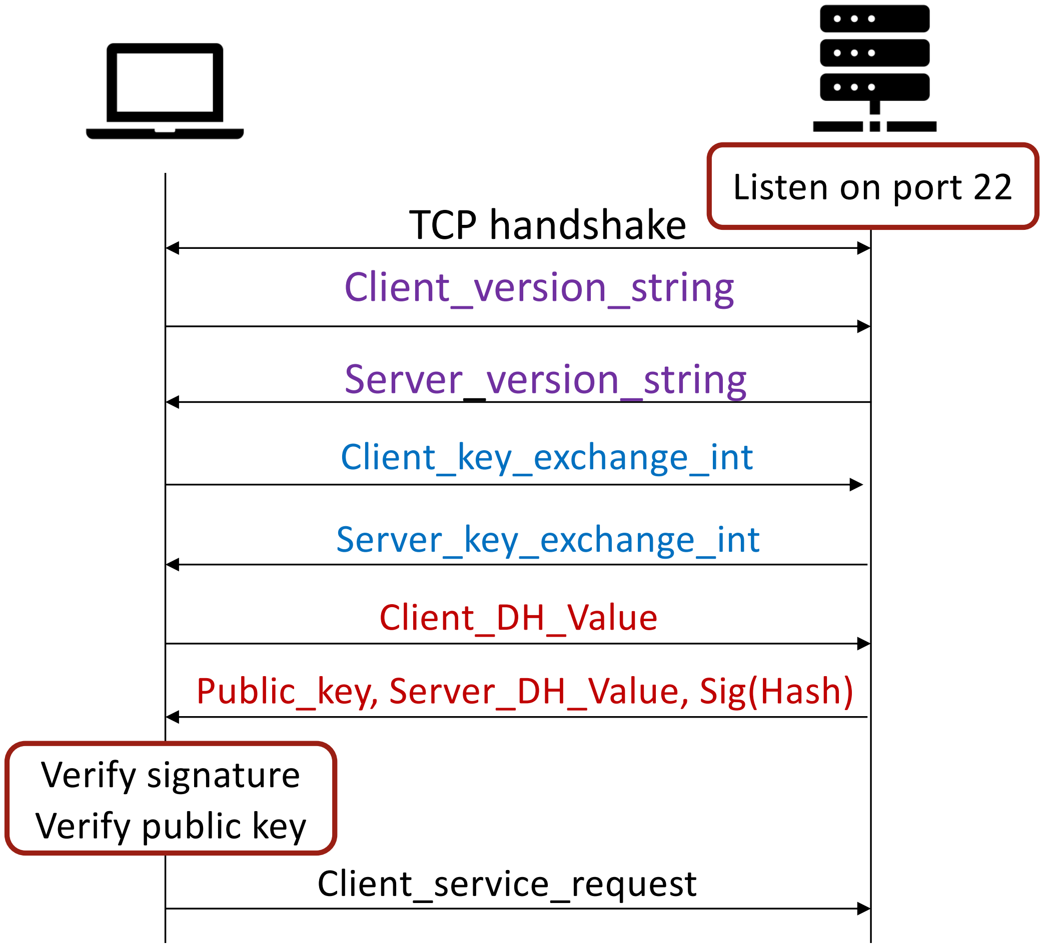


SSH Transport Layer Protocol (1)



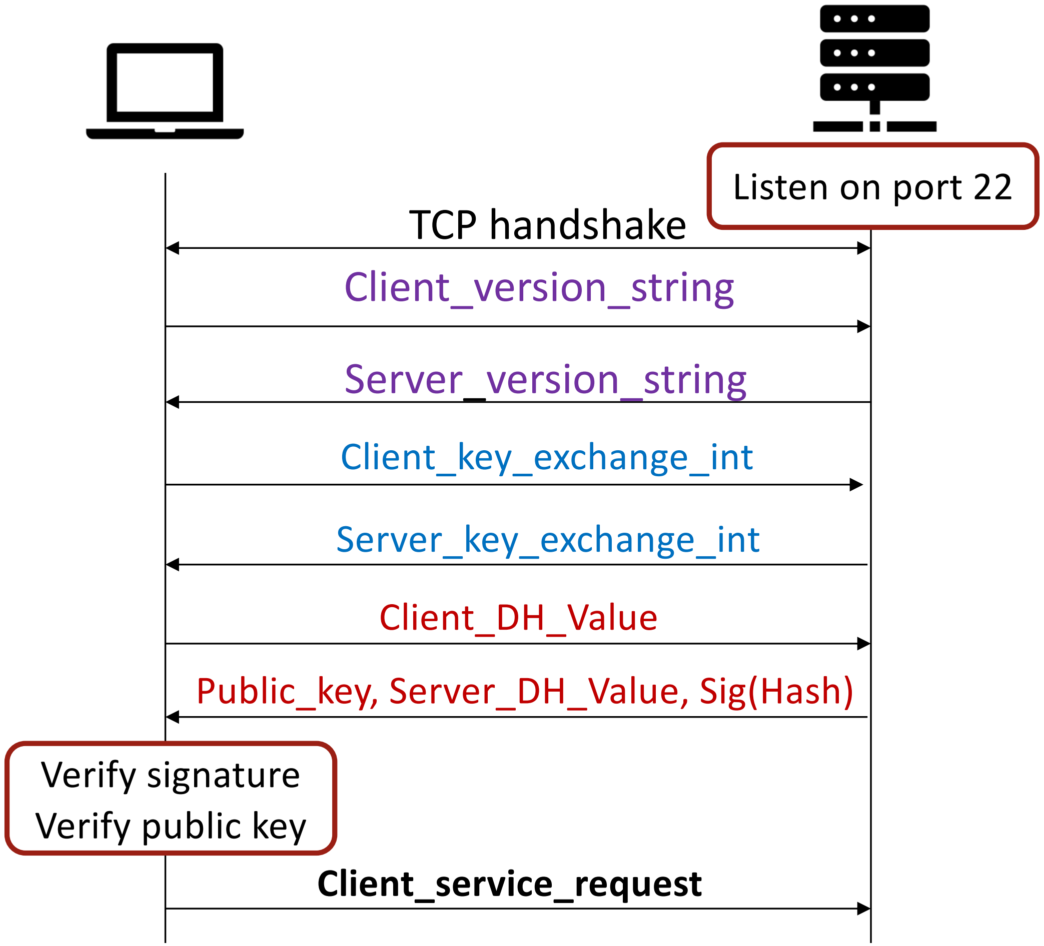
- **Version string**
  - SSH version, SSH software version, optional comments
- **Key exchange init**
  - RAND, list of supported key exchange methods, symmetric encryption algorithms, supported MACs, supported compression algos
- **Algorithm selection**
  - Client list of algorithms ordered according to preference
    - Most preferred one first
  - First algo in client list that is also on server's list is chosen

SSH Transport Layer Protocol (2)



- **DH-Value**
  - Public DH value selected by client resp. server
- **Public key**
  - Bare public key of server
  - Alternative: public key certificate
- **Hash**
  - Computed on all information exchanged so far: Client_version_string || … || Server_DH_Value
- **Sig**
  - Signature computed by server with private key corresponding to public key

SSH Transport Layer Protocol (3)



- **Session key derived from DH key**
  - MAC keys, one per direction
  - Encryption keys, one per direction
  - IVs, one per direction if required for mode of encryption selected
- **Client_service_request**
  - First message protected by the selected algorithms
  - services
    - ssh-userauth
    - ssh-connection

Verifying Server's Public Key

Pre-stored

Client has local database that associates each host name with the corresponding public key of the host

Certificate

The host name – key association is certified by a trusted CA and the server provides an appropriate certificate to the client instead of the public key alone

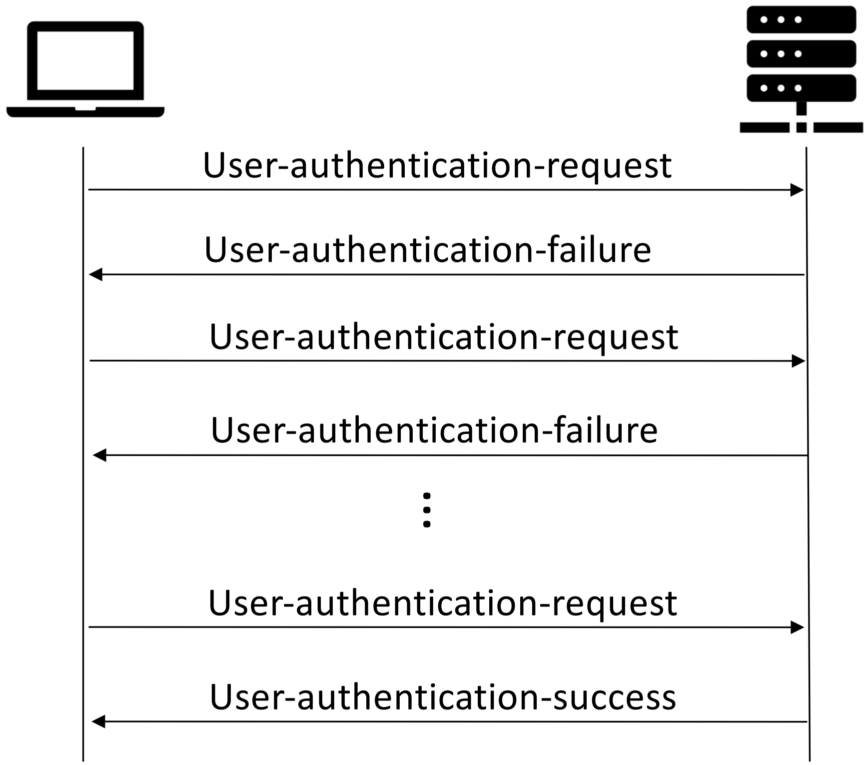
Fingerprint

A fingerprint of the key is shown to the user and can be checked by the user over an external channel

Best effort

Accept host key without checking when first connecting to the server. Save the host key in a local database and use the pre-stored method from there on for this server

SSH User Authentication Protocol



- **User-auth-request**
  - Username
  - Service name
  - Method name
  - Method specific fields
- **User-auth-failure**
  - List of auth. methods that can be used next
  - partial success flag
    - True: request successful but additional auth. required
    - False: previous request not successful
- **User-auth-success**
  - Server starts requested service

User Authentication Request per Method

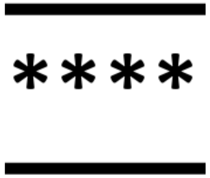
Public key method

public key or public key certificate of user

Signature algorithm

signature on

- session ID (hash of messages exchanged during ssh transport establishment)
- all other data in the request



Password method

send user password over the ssh transport

Host based key method

public key or public key certificate of host

Signature algorithm

signature on

- session ID (hash of messages exchanged during ssh transport establishment)
- all other data in the request

SSH – Connection Protocol

- **Provides**
  - Interactive sessions, i.e., a remote execution of a program
    - E.g., a shell, an application, a system command
    - May involve forwarding of X11 connections
  - Forwarding TCP/IP connections aka port forwarding aka TCP/IP tunneling
    - Comes in different flavors explained on the following slides
- **All these applications can be multiplexed into a single encrypted and integrity protected tunnel**

Summary (1)

- **Many applications can be protected with the help of TLS**
  - HTTP, FTP, …
- **Some applications nevertheless have special needs**
  - Email: asynchronous natures makes use of TLS end-to-end impossible
  - DNS: caching makes use of TLS end-to-end undesirable
- **TLS may help to protect data transfer in a hop-by-hop fashion**
  - E.g., Email from mail server to user's mail client
- **End-to-end protection of Email can be provided by PGP or S/MIME**
  - Both support encryption with symmetric key for end-to-end confidentiality
  - Digital signatures on hash of message for non-repudiation
  - Encryption of symmetric key with public key of receiver

Summary (2)

- **PGP or S/MIME mainly differ in how public keys are distributed**
  - S/MIME uses CA and classical certificates
  - PGP originally designed to use user-signed certificates
    - Supports use of CAs as well
    - Authenticity of keys “computed” from individually assigned trust levels
- **End-to-end protection does not prevent spoofing if it is optional to use**
  - We still see a lot of email spoofing especially in the context of phishing and SPAM
  - DKIM, SPF, and DMARC aim at making it harder
- **DNS system is mainly used to map domain names to IP addresses**
  - Also, to map domain names to name servers and mail servers responsible for the domain

Summary (3)

- **DNS is originally unprotected**
  - DNS queries and responses are neither encrypted nor integrity-protected
  - Consequently, DNS responses can be forged (see cache poisoning for example)
    - **DNSSec** ensures that only authentic RRs are cached
    - Public keys required to verify authenticity of RRs are distributed via DNS
  - DNS queries can be eavesdropped on
    - Addressed by **DoT** which protects connection between client and local DNS server with TLS
- **Remote login**
  - SSH offers authentication, confidentiality, and integrity for remote login
  - SSH was developed in parallel to the first SSL versions
  - Telnet over TLS offers the similar security services as SSH but SSH offers more additional features

References

Book Chapters

- Stallings Chapter 16

RFCs related to Email Security

- PGP: RFC 4880 OpenPGP Message Format
- S/MIME: RFC 5751
- DKIM: RFC 6376
- SPF: RFC 7208
- DMARC: RFC 7489

Root server

- https://root-servers.org

Book

- Allan Liska and Geoffrey Stowe: “DNS Security: Defending the Domain Name System”, Elsevier 2016

RFCs related to DNS

- RFC 1034: Domain Names – Concepts and Facilities
- RFC 1035: Domain Names – Implementation and Specification
- RFC 4033: DNS Security Introduction and Requirements
- RFC 4034: Resource Records for the DNS Security Extensions
- RFC 4035: Protocol Modifications for the DNS Security Extensions

Key Signing ceremony for the root KSK

- https://www.iana.org/dnssec/ceremonies

References

- **W. Stallings, Cryptography and Network Security: Principles and Practice, 8th edition, Pearson 2022**
  - Chapter 19: Electronic Mail Security, DNS, DNSSec
  - Chapter 17.4: SSH

SSH Specification Details

- The secure shell transport layer protocol RFC 4253
- Latest Algorithm recommendations for SSH
  - RFC 9142