UNIVERSITÄT DUISBURG ESSEN

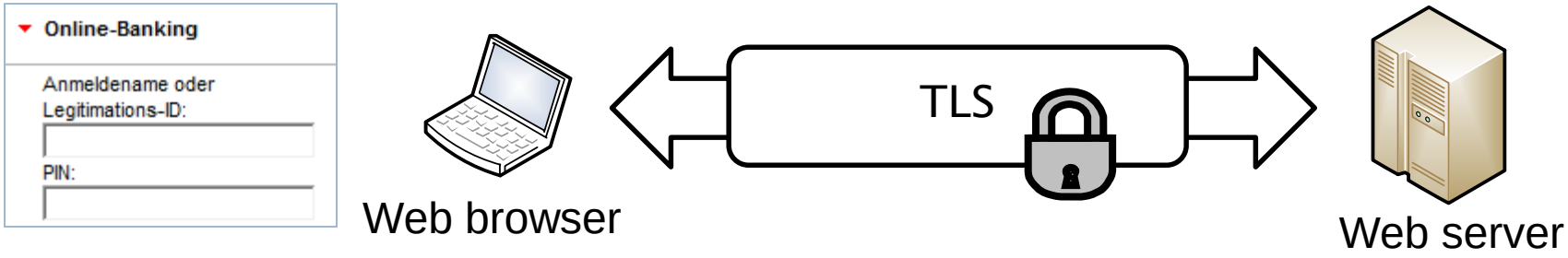
# Sicherheit in Kommunikationsnetzen
## (Network Security)

## Transport Layer Security

Dr.-Ing. Matthäus Wander

Universität Duisburg-Essen

# Transport Layer Security (TLS)




- Cryptographic protocol
  - Provides security on top of reliable transport (TCP)
  - Used to secure HTTP, SMTP, IMAP, XMPP and others
- Security goals
  - Authentication of server and optionally client
  - Data integrity (no manipulation of data)
  - Confidentiality (encryption)

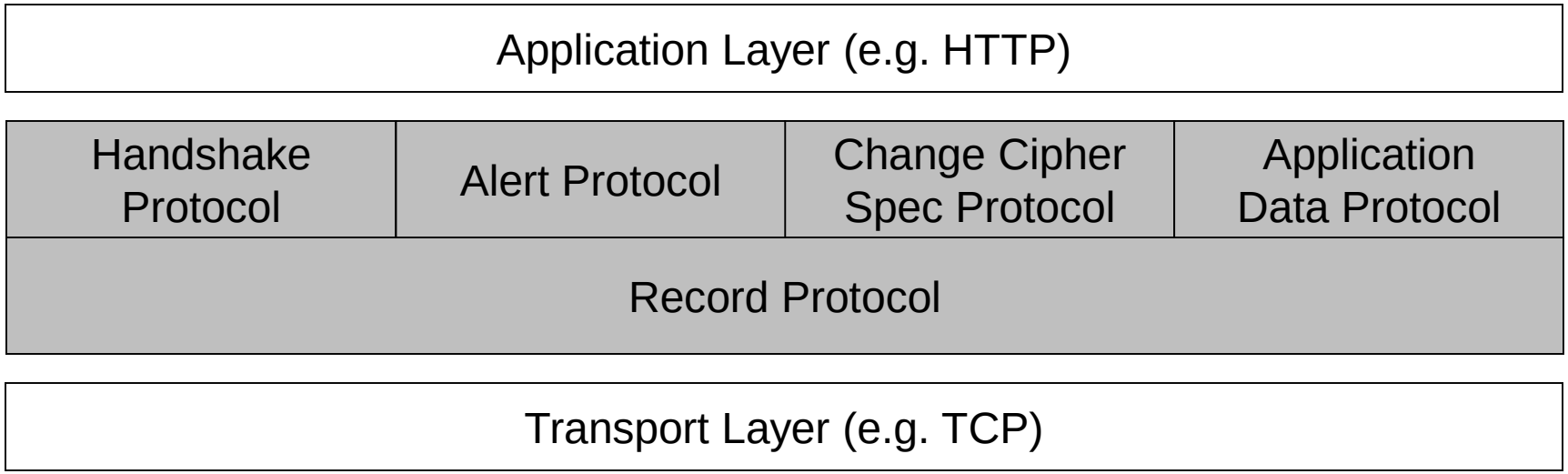
# History

- Predecessor: Secure Sockets Layer (SSL)
  - Developed by Netscape for HTTPS (= HTTP + SSL)
- 1994: SSL 2.0
  - Insecure, major security weaknesses
- 1995: SSL 3.0
  - Insecure, deprecated since 2015 (RFC 7568)
- 1999: TLS 1.0, minor improvements to SSL 3.0
  - Standardization and further development by IETF

# History

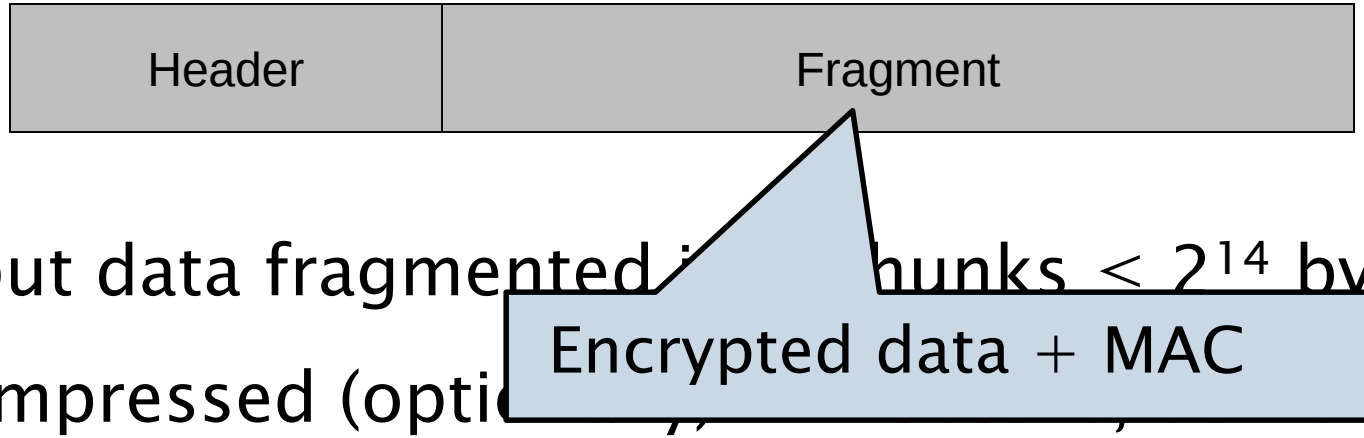
- 2003: TLS Extensions
  - New features or security mechanisms (e.g. Encrypt-then-MAC, RFC 7366)
- 2006: TLS 1.1
  - Security improvements
- 2008: TLS 1.2 — **Presented in this lecture**
  - Added new or replaced old cryptographic algorithms
- TLS 1.3 is work in progress as of 2017
  - Outlook: new handshake and other major changes

# Protocol Overview



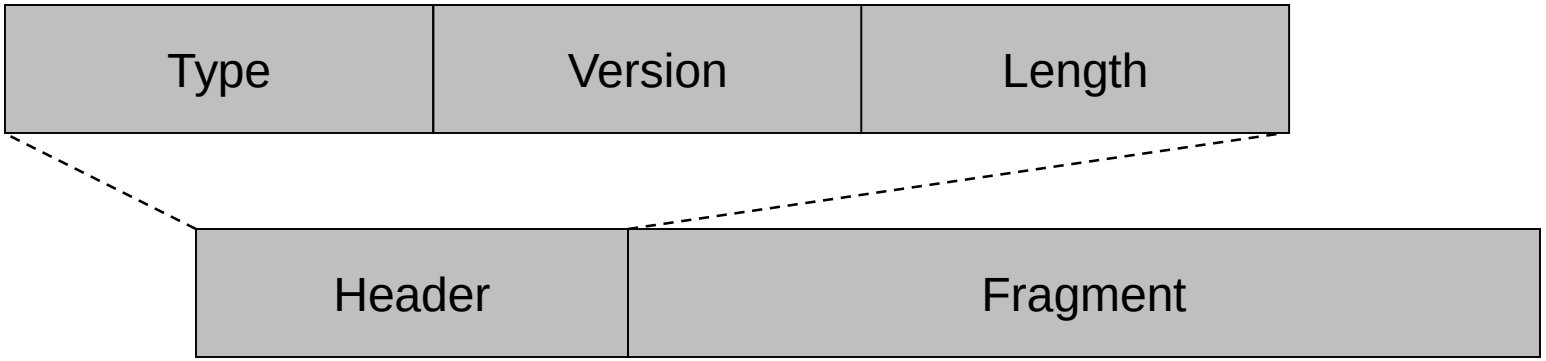

- TLS consists of two layers:
  - Handshake Protocol and other protocols
  - Record Protocol

# Record Protocol



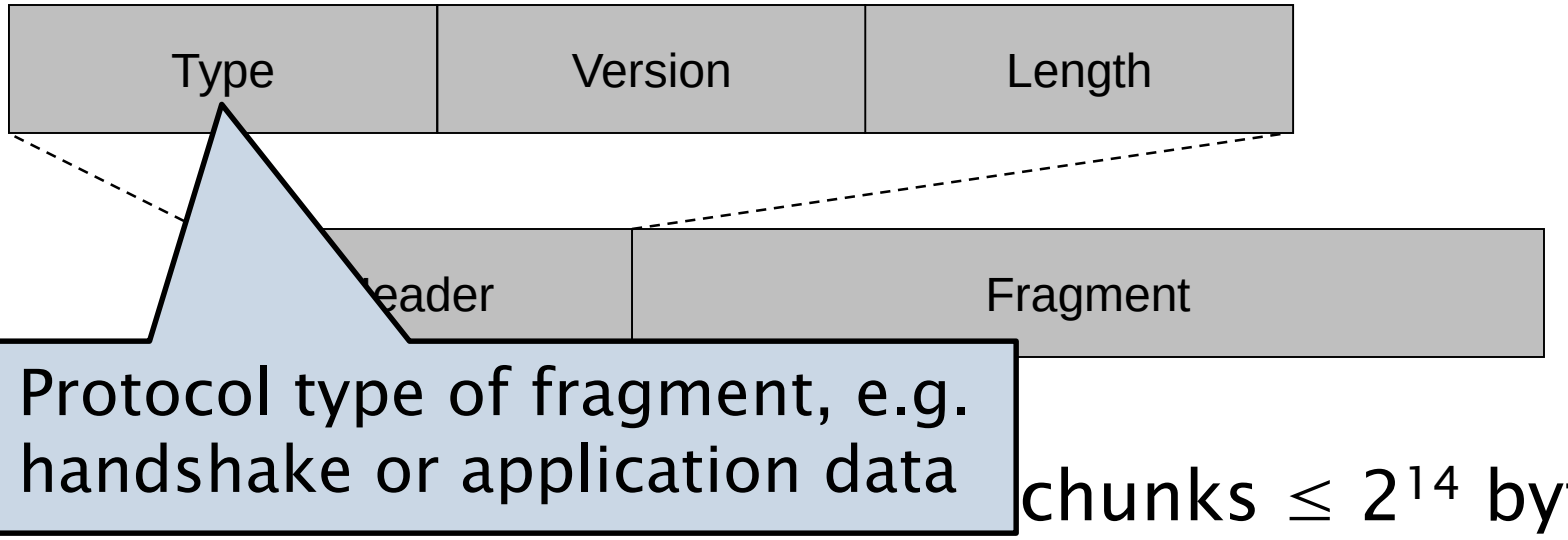

- Input data fragmented i hunks < $2^{14}$ bytes
- Compressed (optio se
- Encrypted and integrity protected (e.g. MAC)
- Prepended with header

# Record Protocol




- Input data fragmented into chunks $\leq 2^{14}$ bytes
- Compressed (optional) ⇨ insecure, deprecated
- Encrypted and integrity protected (e.g. MAC)
- Prepended with header

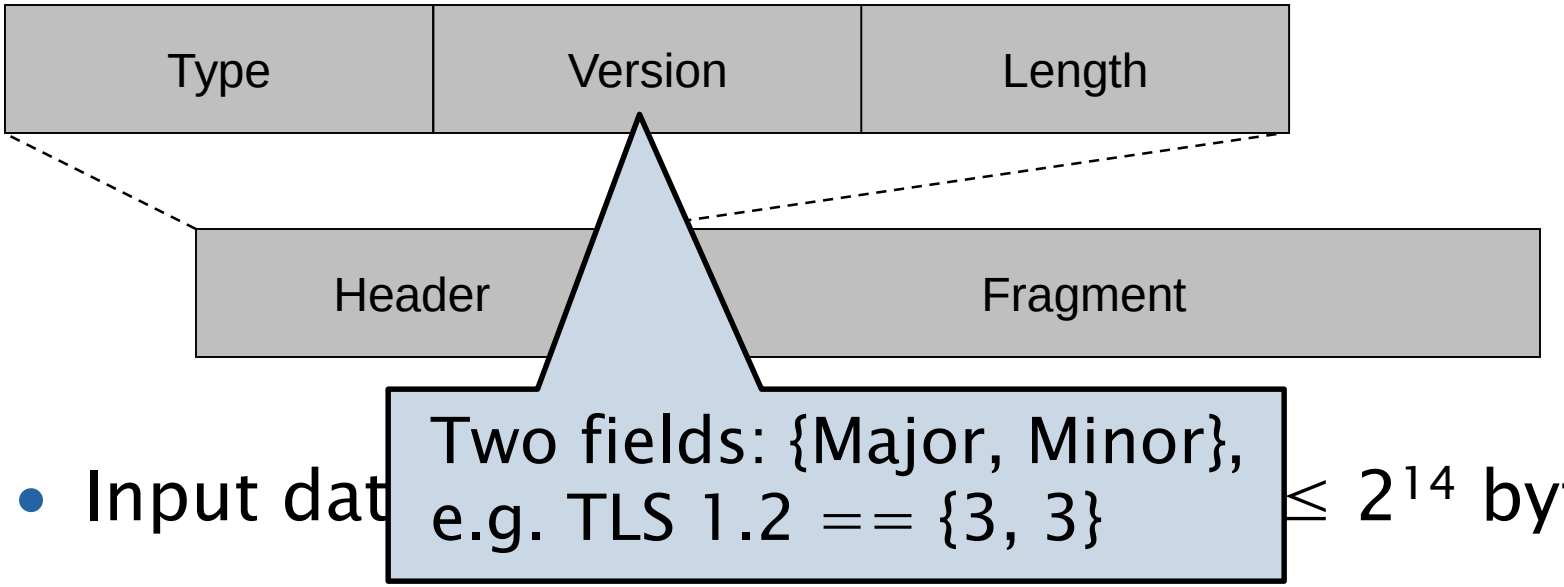
# Record Protocol




chunks $\leq 2^{14}$ bytes

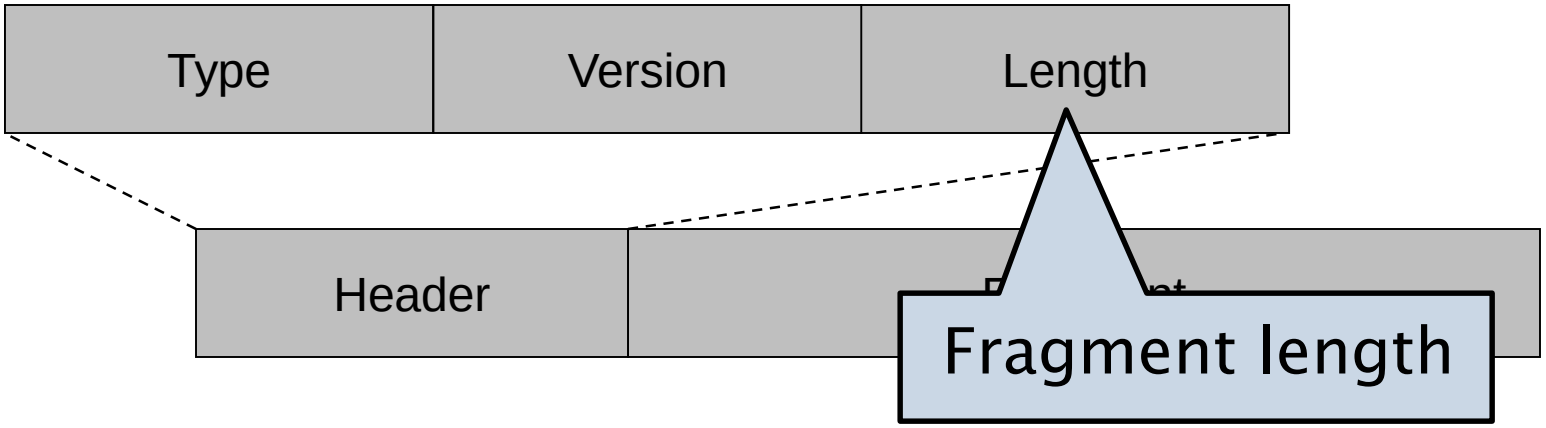
- Compressed (optional) ⇨ insecure, deprecated
- Encrypted and integrity protected (e.g. MAC)
- Prepended with header

# Record Protocol



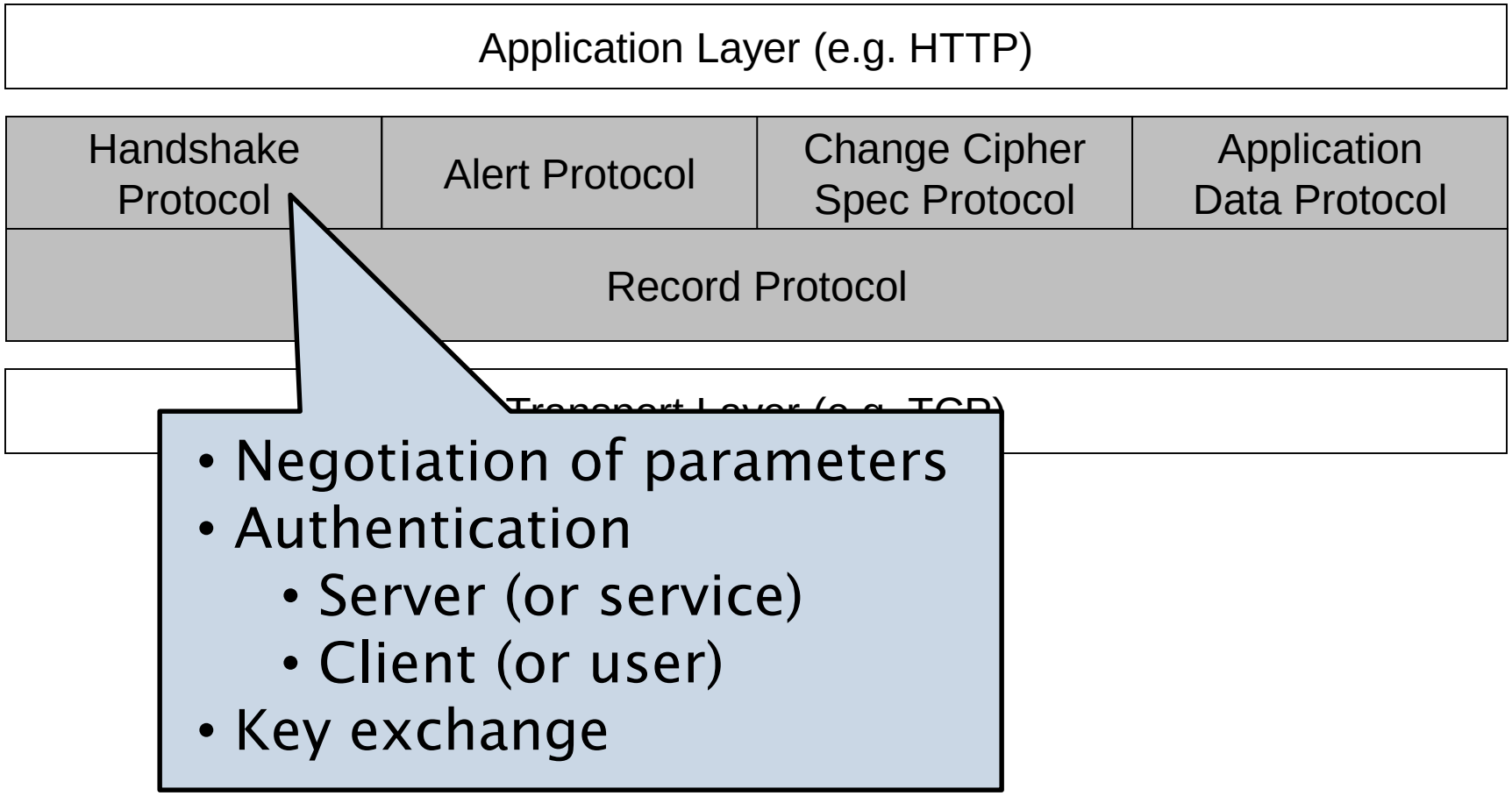

- Input dat ≤ $2^{14}$ bytes
- Compressed (optional) ⇨ insecure, deprecated
- Encrypted and integrity protected (e.g. MAC)
- Prepended with header

# Record Protocol

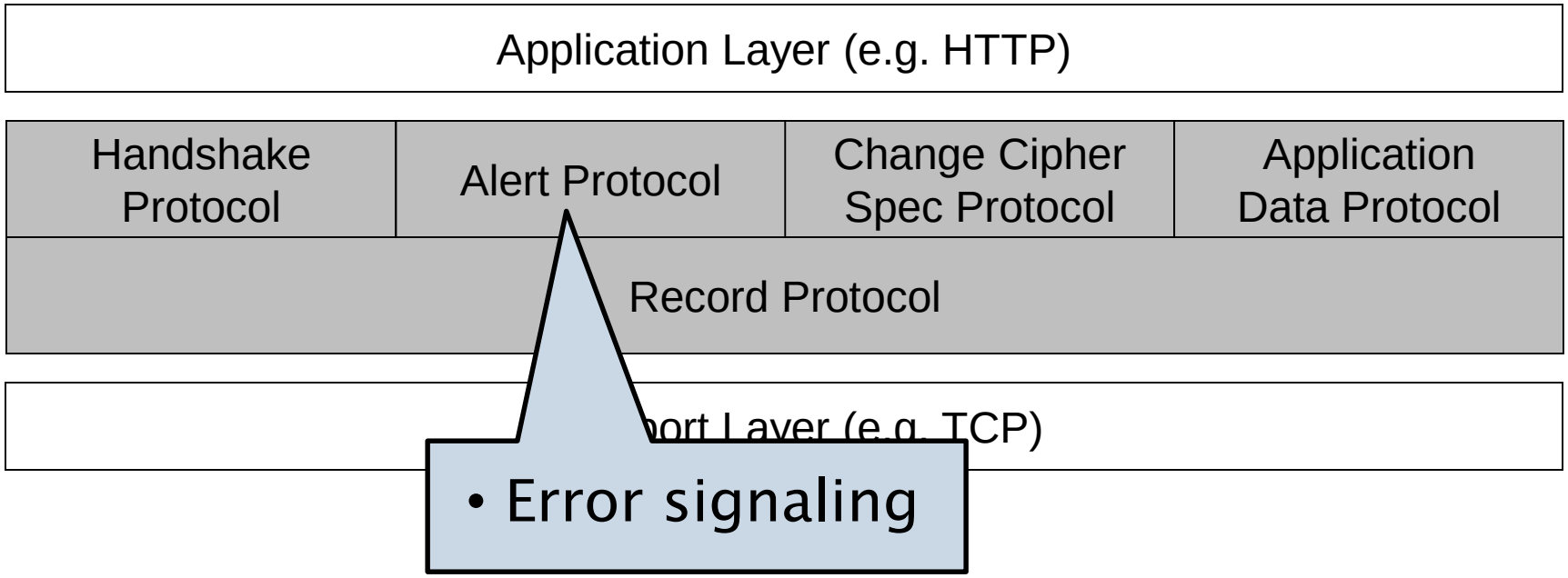



- Input data fragmented into chunks $\leq 2^{14}$ bytes
- Compressed (optional) ⇨ insecure, deprecated
- Encrypted and integrity protected (e.g. MAC)
- Prepended with header

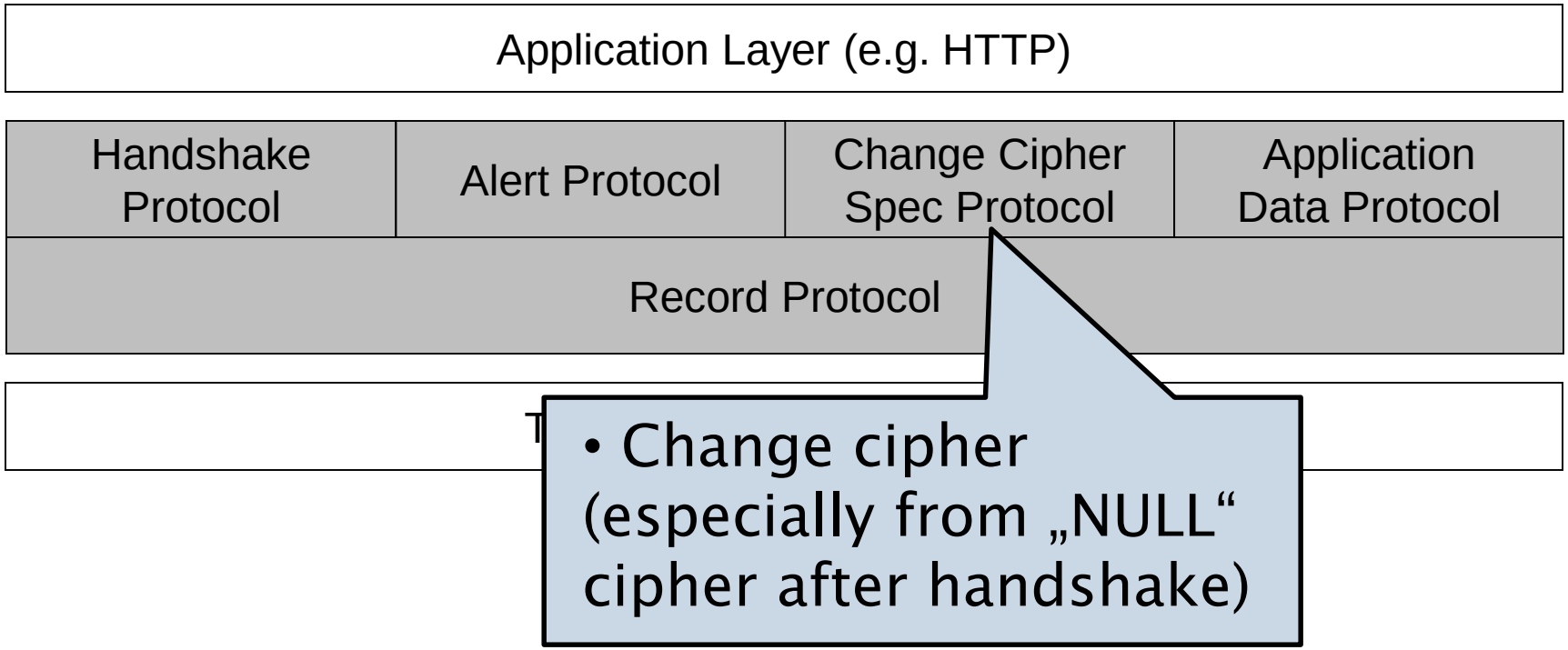
# Protocol Overview



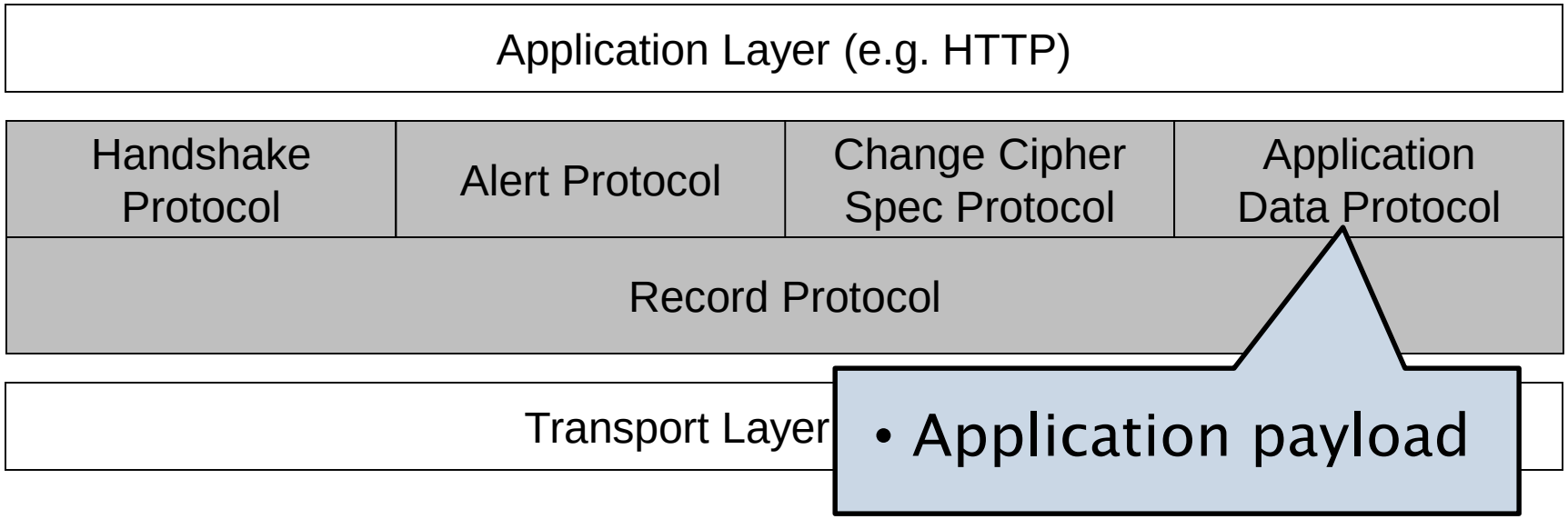
# Protocol Overview



# Protocol Overview



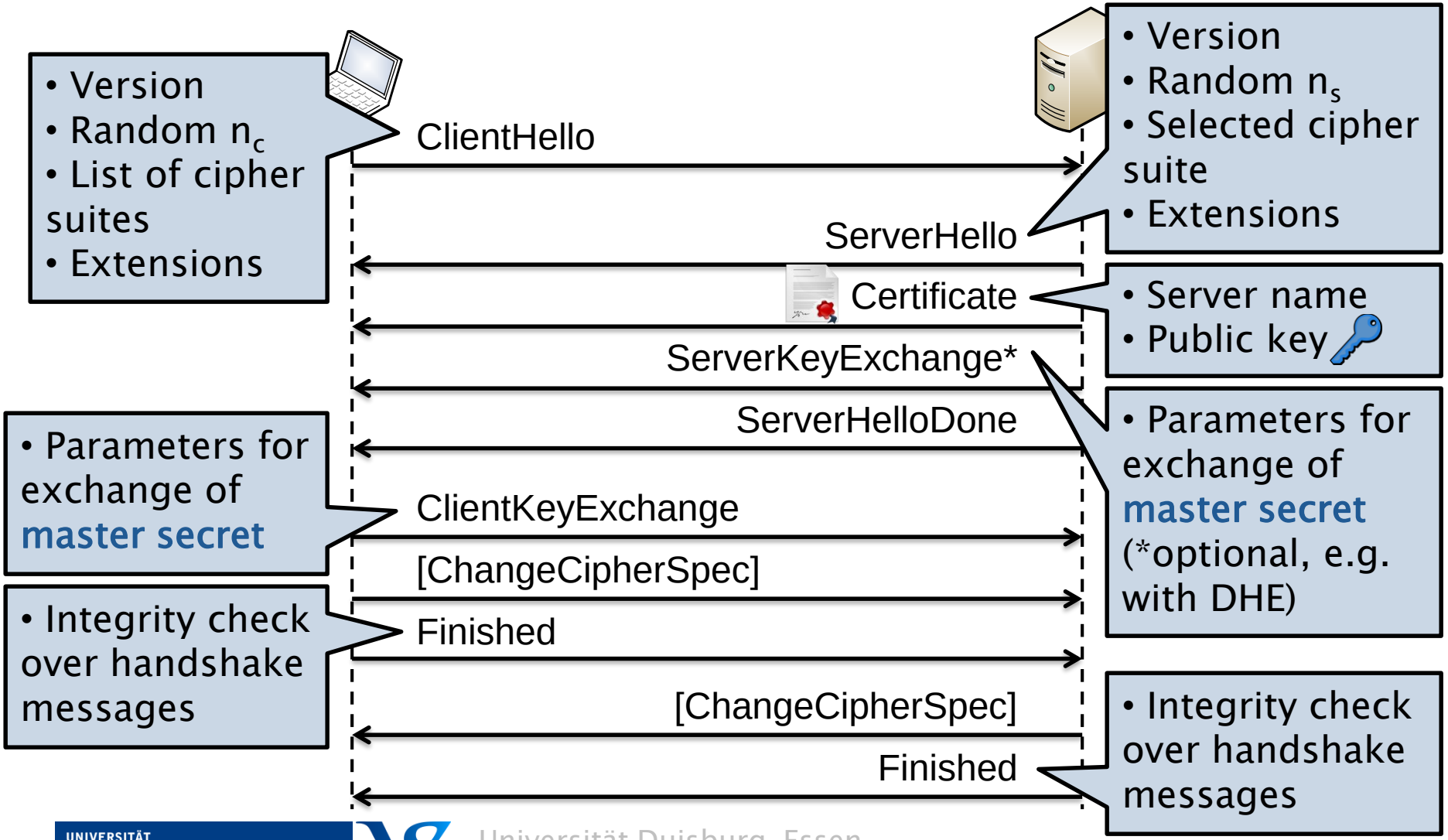
# Protocol Overview



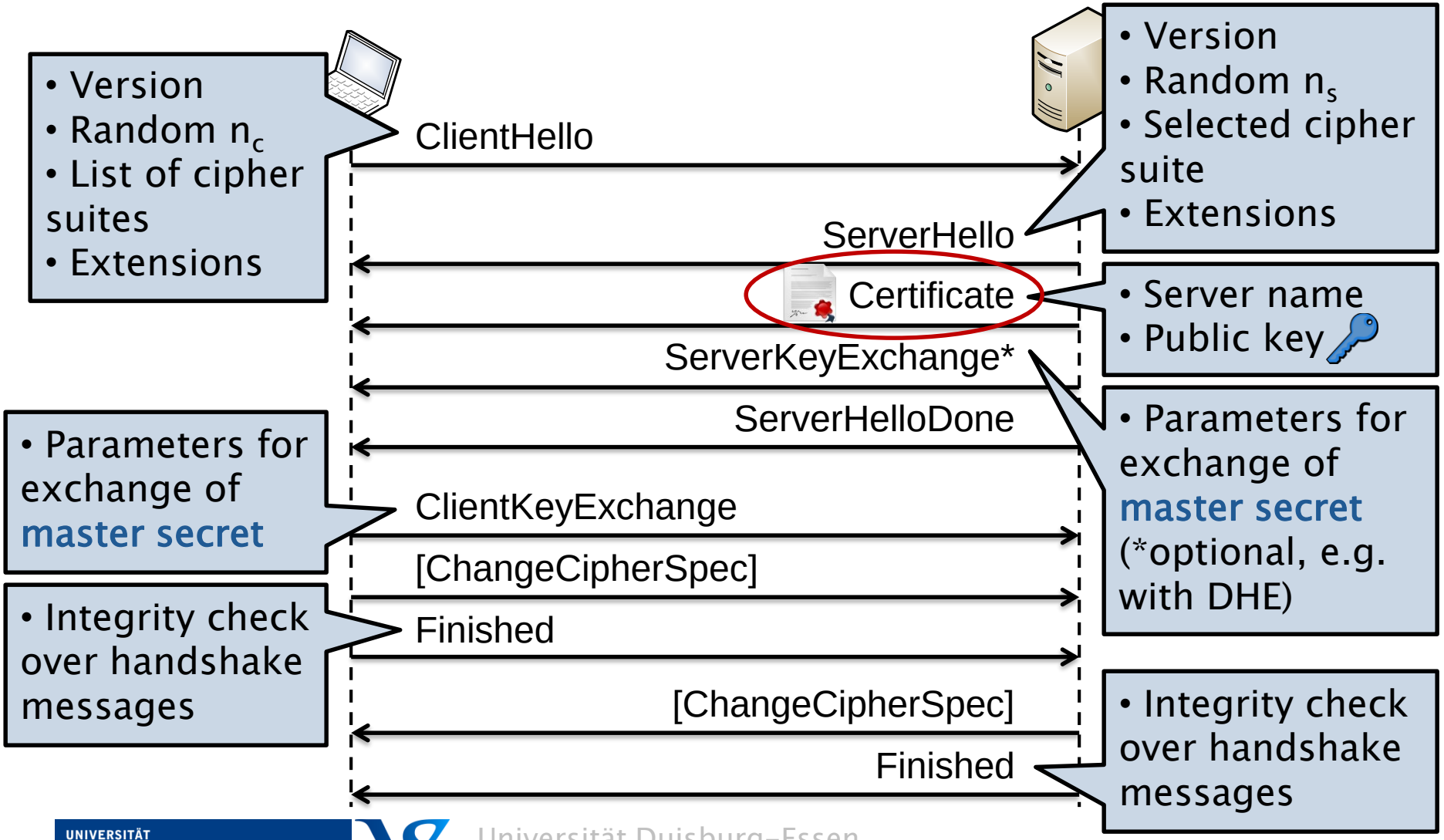
# Handshake

- Create new session or resume established one
  - Full or abbreviated handshake (faster)
- Exchange capabilities and negotiate parameters
  - e.g. cipher suites, extensions
- Authentication
  - Anonymous (no authentication): insecure
  - Server only (via certificate): most common
  - Client and server (via certificates): occasionally
- Key exchange: agree on shared master secret

# Handshake Sequence



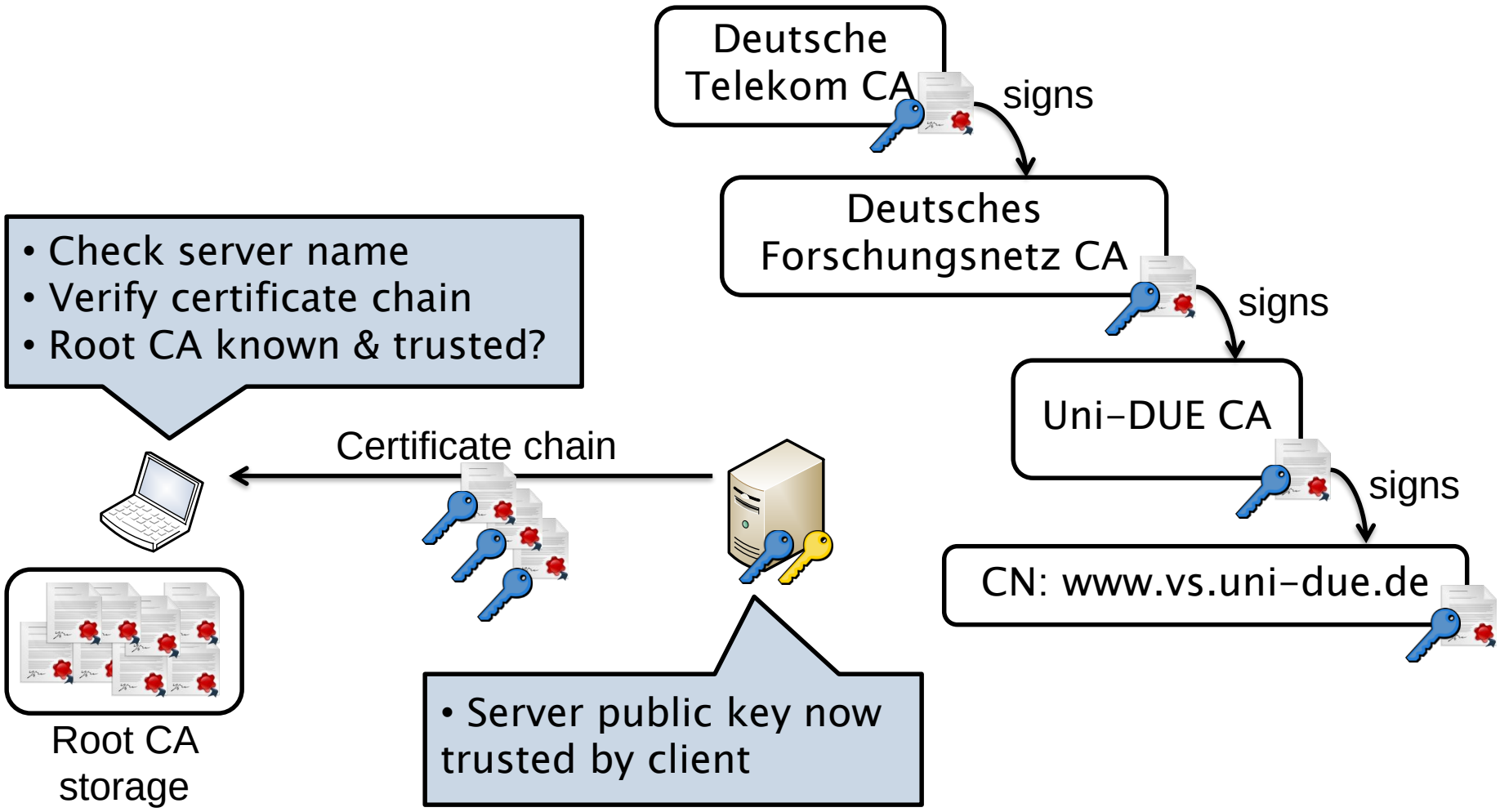
# Handshake Sequence



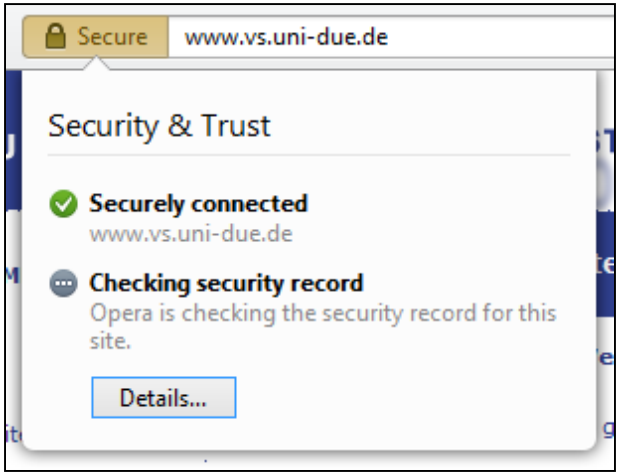
# X.509 Certificates

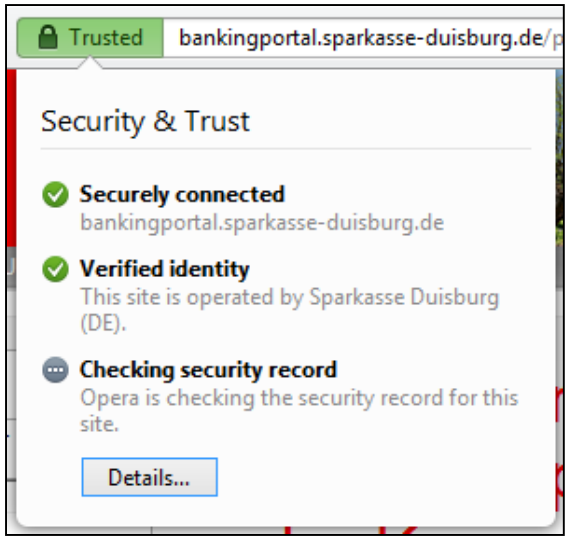
- TLS uses the X.509 public-key infrastructure
- Certificate  has public key  and server name
  - Issued and signed by certificate authority (CA)
  - or self-signed, i.e. untrusted (⇨browser warning)
- Applications checks:
  - Server name matches „Common Name“ in certificate?
  - ... or any of the „Subject Alternative Name“?
  - Issuer CA known trusted by client?
  - Certificate expiration time? Revocation status?

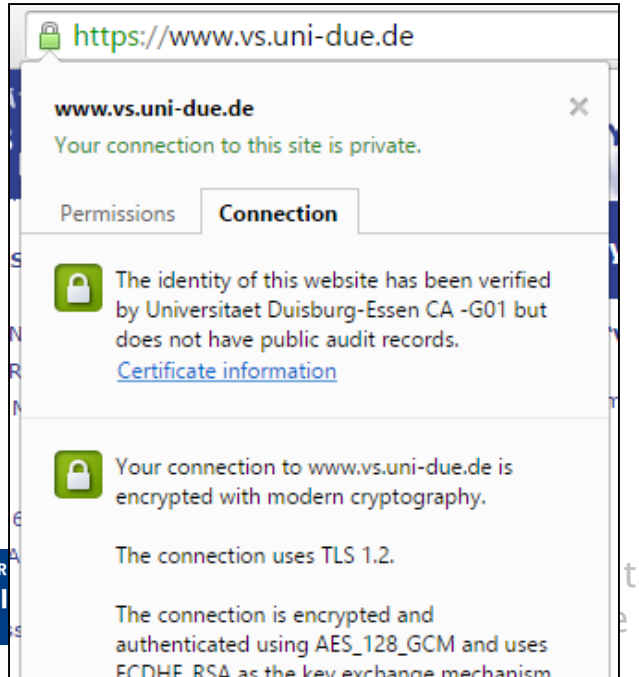
# X.509 Public Key Infrastructure

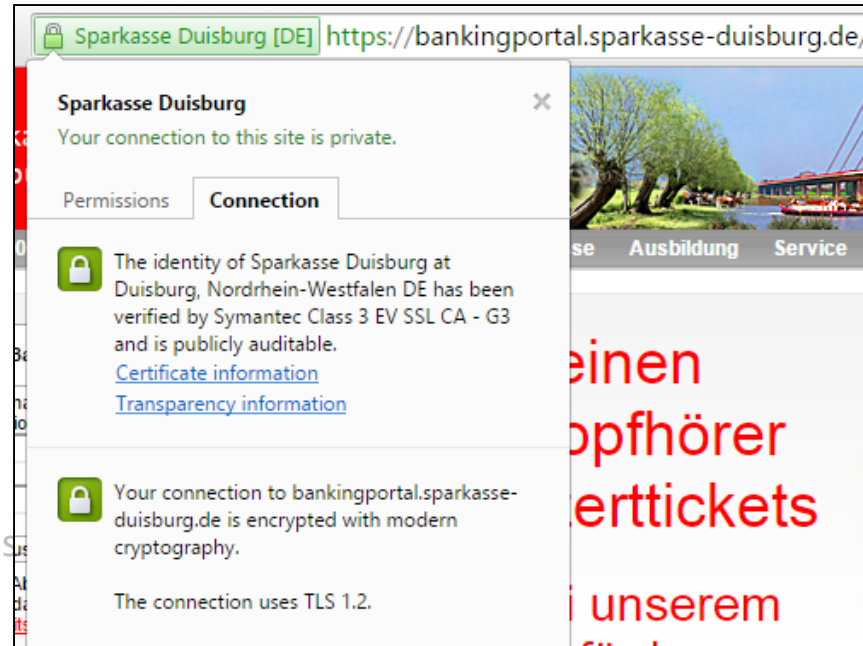


# Example Certificates in Web Browser



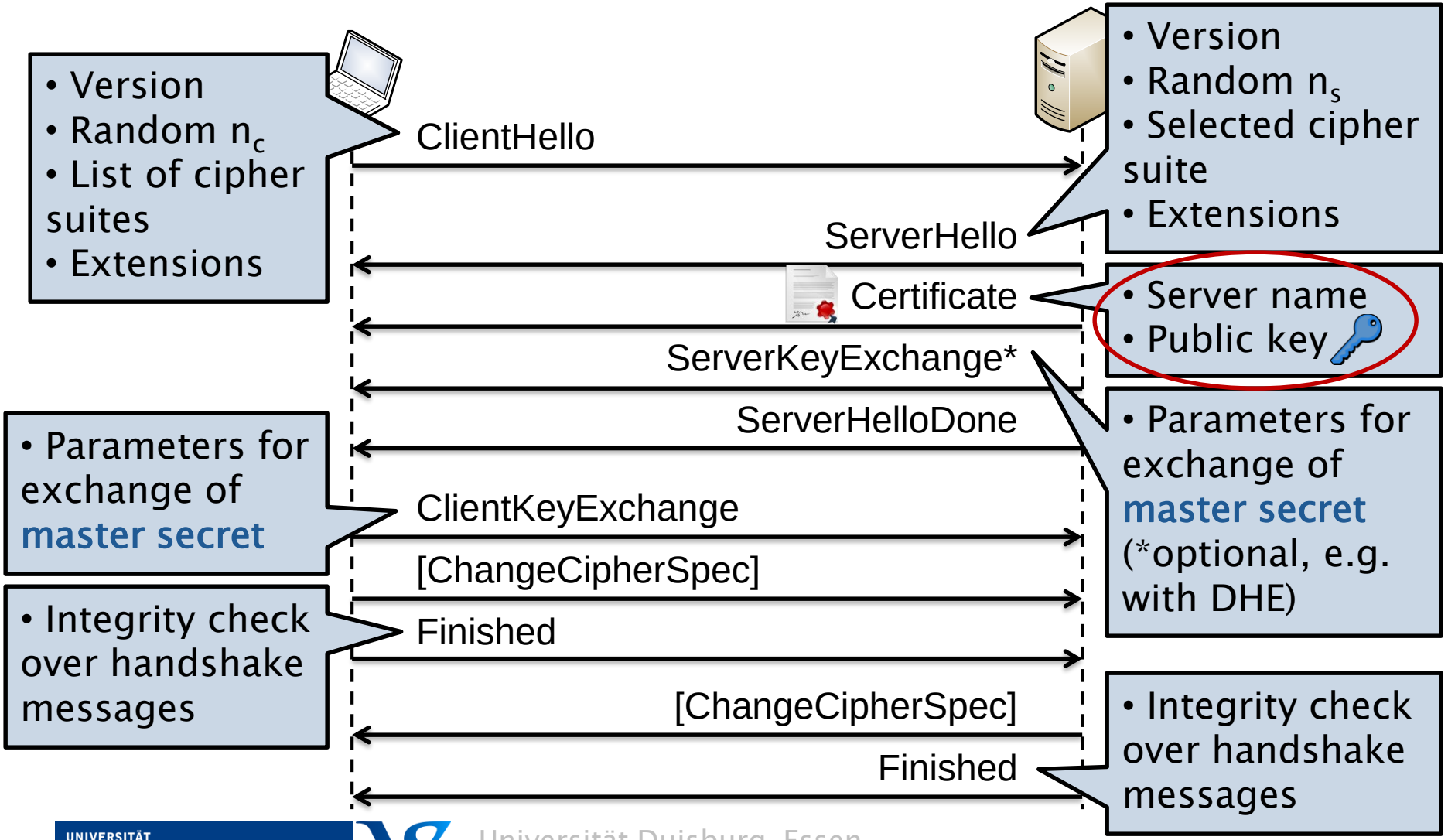



# Domain Validation and Extended Validation

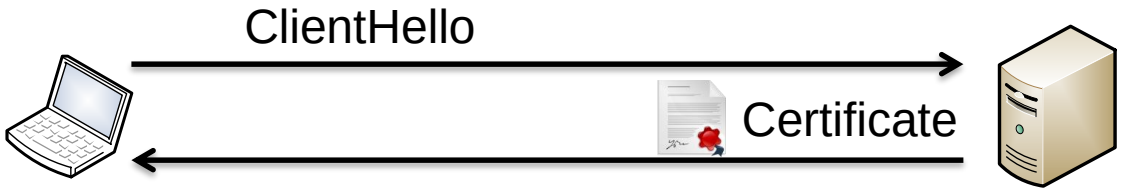
- Domain validation (DV)
  - CA sends email to domain owner
  - ... or requires setup of a token (unique string) in DNS zone or on a web server
  - This is an automatic check mostly, maybe in addition with a simple identity check (e.g. photo of ID card)
- Extended validation (EV)
  - Extensive identity check (e.g. phone call, bills, ID card)
  - Businesses must provide legal proofs, e.g. registration of company name

# Handshake Sequence

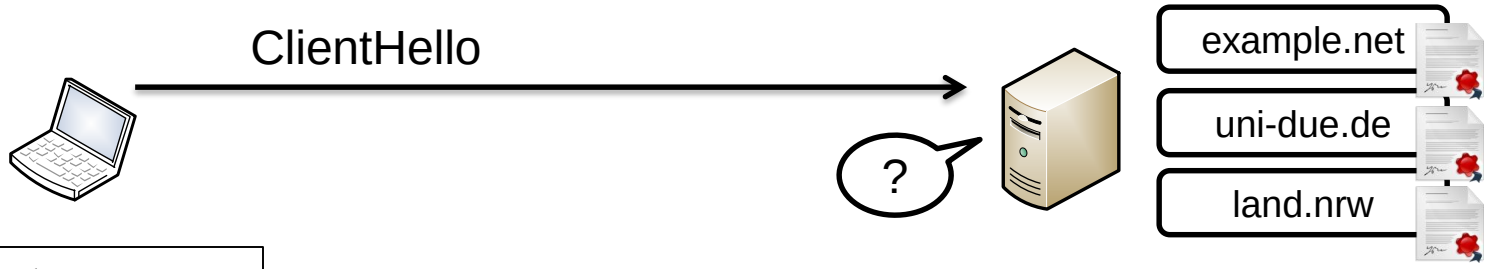


# Server Name Indication

- TLS server returns certificate before client sends application data




- Problem: what if server hosts multiple domains (virtual hosts) with different certificates?

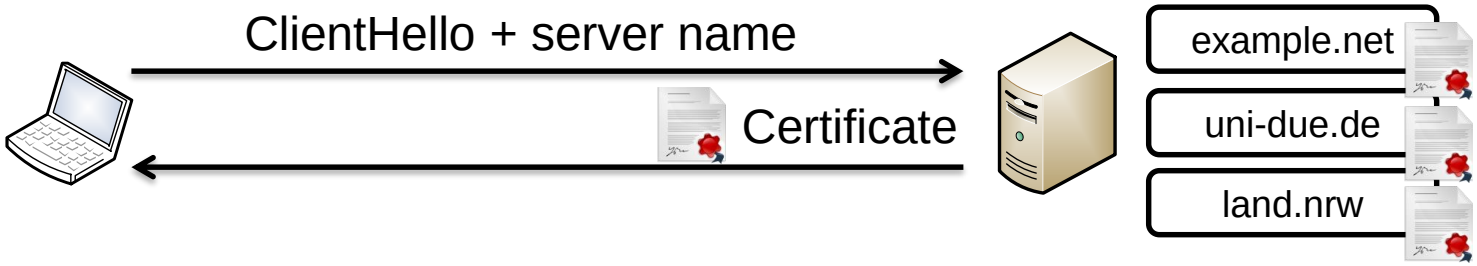



Not sent yet:

```
GET / HTTP/1.1
Host: www.example.net
```

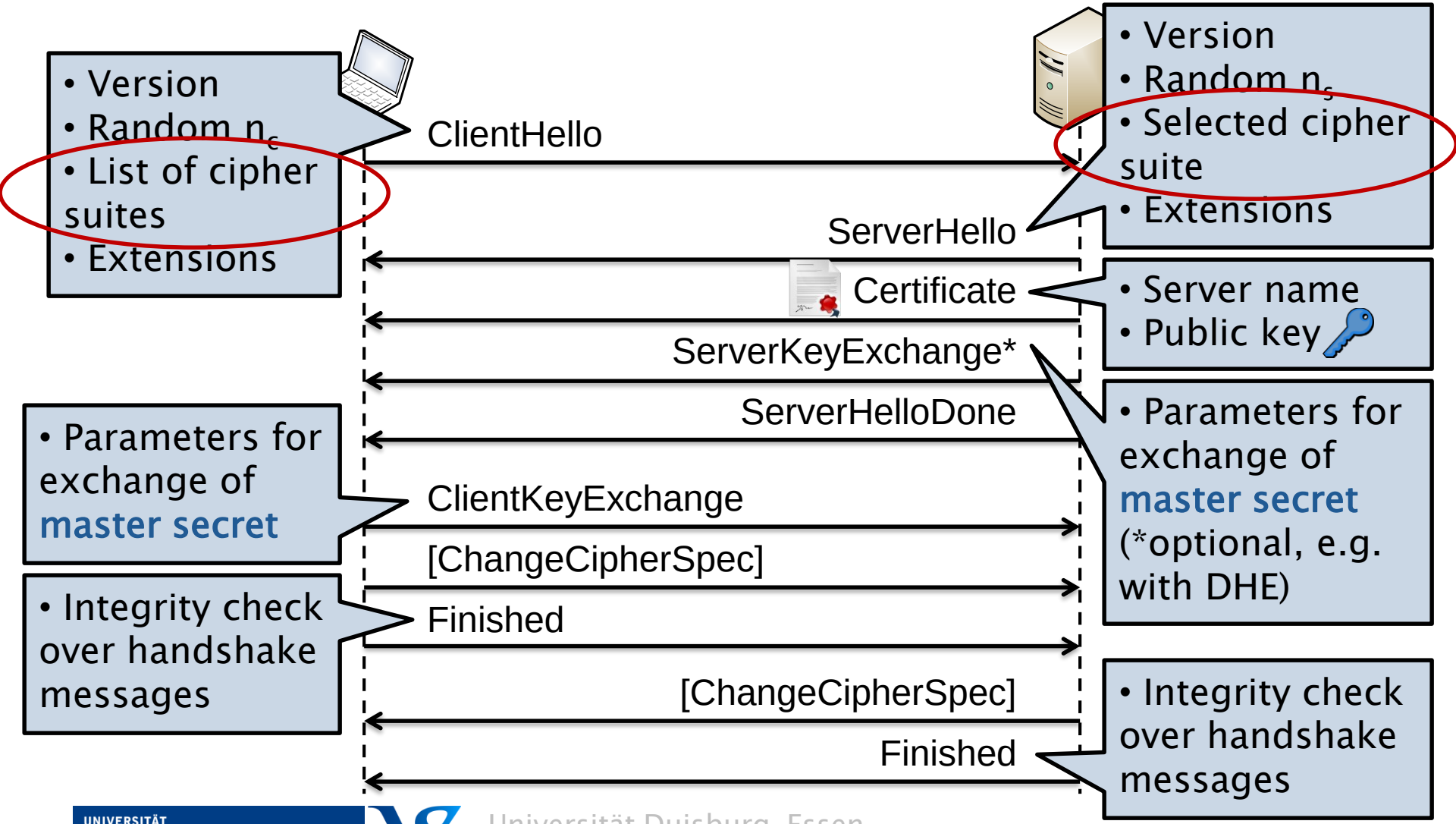
# Server Name Indication (2)

- Solution: include requested server name in TLS connection request




- Server returns certificate that matches server name requested by client
- TLS extension „Server Name Indication“ (SNI)
  - Supported by all current web browsers and servers

# Handshake Sequence



# Cipher Suites

**TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256**

- Cipher suite: set of ciphers used for TLS session
  - >300 specified, a lot of them insecure and deprecated
- Recommended in RFC 7525 (with TLS ≥1.2):

TLS_DHE_RSA_WITH_AES_128_GCM_SHA256
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
TLS_DHE_RSA_WITH_AES_256_GCM_SHA384
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384

# Cipher Suites

**TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256**

Key exchange and authentication

- Cipher s… set of ciphers used for TLS session
  - …d deprecated
- …≥1.2):
  - …6
  - …4

- ECDHE_RSA
  - Elliptic Curve
  - Diffie-Hellman
  - Ephemeral (with Forward Secrecy)
  - RSA authentication (DH agrees on key but does not authenticate peer)

# Cipher Suites

**TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256**

Key exchange and authentication

Encryption cipher (algorithm, key size, mode)

- Cipher suite: set of ciphers us TLS session
-
  - AES_128_GCM
    - AES symmetric cipher
    - 128 bit key length
    - Galois/Counter Mode (authenticated encryption)

TLS_DHE_RSA_WITH_AES_256_GCM_SHA384

TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384

# Cipher Suites

**TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256**

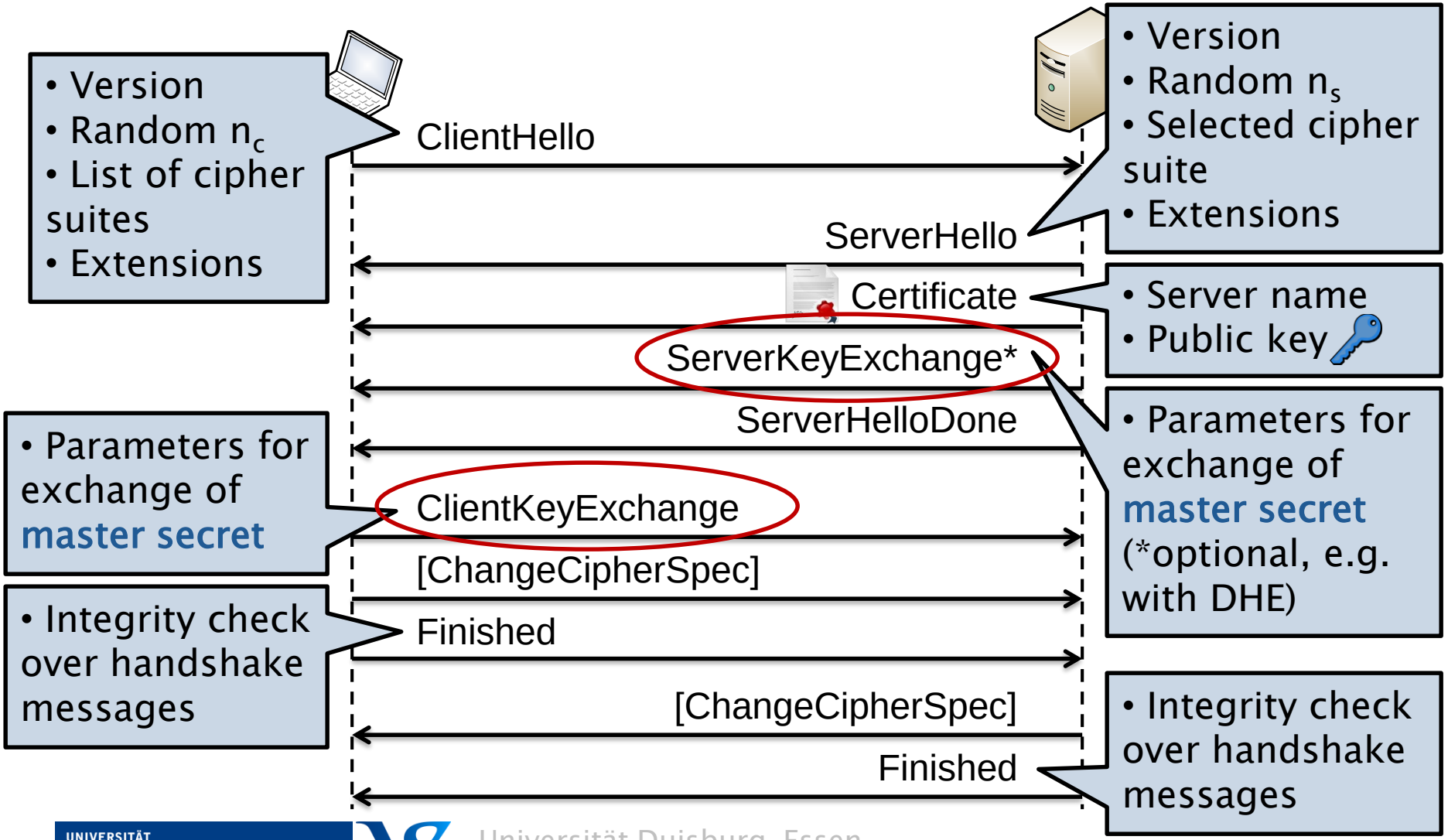
- Key exchange and authentication
- Encryption cipher (algorithm, key size, mode)
- Hash function

- Cipher suite: set of ciphers used f[illegible]ession
-
  - SHA256
    - HMAC-SHA256 keyed-hash function
    - used in pseudorandom function (PRF)
    - PRF derives keys from secret, random data

TLS_DHE_RSA_WITH_AES_256_GCM_SHA384

TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384

# Handshake Sequence



# Key Exchange

- Objective: shared 48-byte master secret (**MS**)
  - Used to derive cipher keys for integrity and encryption
- Key exchange: agree on premaster secret (**PMS**)
  - RSA transport: client generates **PMS**, sends to server encrypted with server public key 
  - Diffie-Hellman: client and server negotiate **PMS**
- **MS** = PRF(**PMS**, „master secret“, $n_c$ || $n_s$)
  - „master secret“ is a constant label
  - Client and server nonce are given as random seed

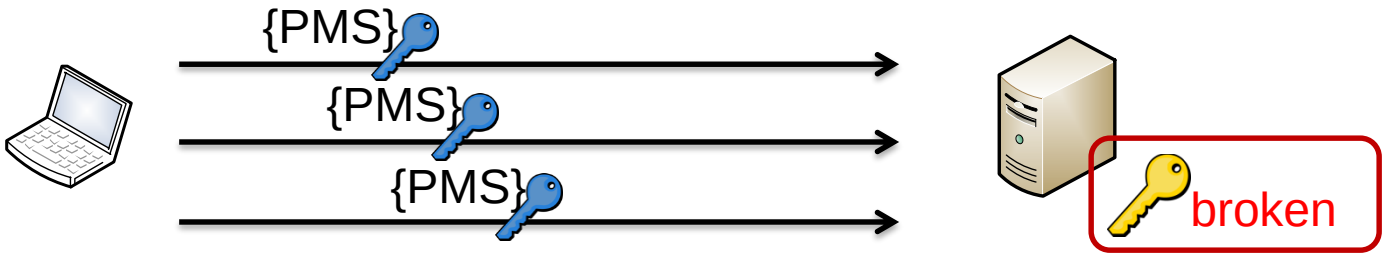
# Pseudorandom Function

- Definition: a pseudorandom function (PRF) is a deterministic function that:
- Input: receives a seed and context-specific data
- Output: produces a pseudorandom sequence of fixed length
  - Similar to PRNG, but for fixed-length output
- Purpose: e.g. key expansion from small secret
- In TLS, the PRF is essentially a keyed-hash function based on HMAC

# Pseudorandom Function in TLS

- PRF(secret, label, seed) = P(1) || P(2) || ... || P(i)
  - label is constant, seed is random
  - Output length depends on how much data is needed
- $P(i) = HMAC_{secret}(A_i \,||\, label \,||\, seed)$
  - with $A_0 = label \,||\, seed$
  - with $A_i = HMAC_{secret}(A_{i-1})$
  - each P(i) call consists of two nested HMAC calls
- Hash function for PRF specified by cipher suite
  - Typical choice: SHA-256

# Perfect Forward Secrecy




- Scenario: attacker collects encrypted TLS traffic
  - Server private key broken or exposed in future
- RSA-encrypted key transport:
  - Attacker can decrypt all TLS sessions (past and future) with RSA private key
- No perfect forward secrecy with RSA transport

# Perfect Forward Secrecy (2)

- Definition: a protocol has perfect forward secrecy if compromise of long-term key ⇏ compromise of past session keys
  - Sessions in the past remain secure even with a key compromise now or in the future
  - Deutsch etwa: „nach vorne gerichtete Geheimhaltung“
- We cannot protect future sessions after a broken key from man-in-the-middle attacker
  - But sessions that were once secure from impersonation should remain confidential
  - Note: still breakable, but with high effort per session

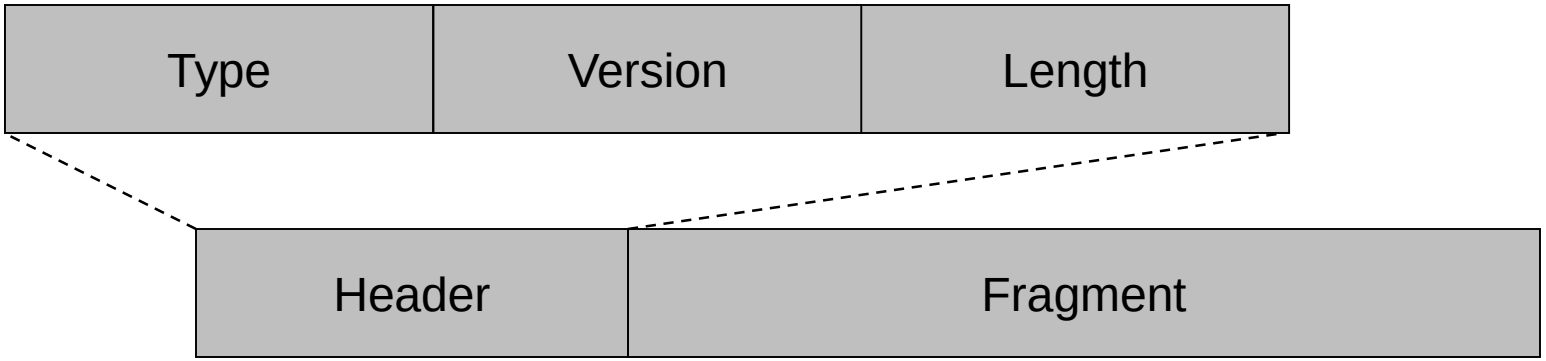
# Perfect Forward Secrecy (3)


- Scenario: attacker collects encrypted TLS traffic
  - Server private key 🔑 broken or exposed in future
- Ephemeral Diffie-Hellmann key exchange:
  - Attacker can break authenticity of signatures
  - Broken private key ⇏ broken Diffie-Hellman
  - Perfect forward secrecy: session keys remain private

# Perfect Forward Secrecy (4)

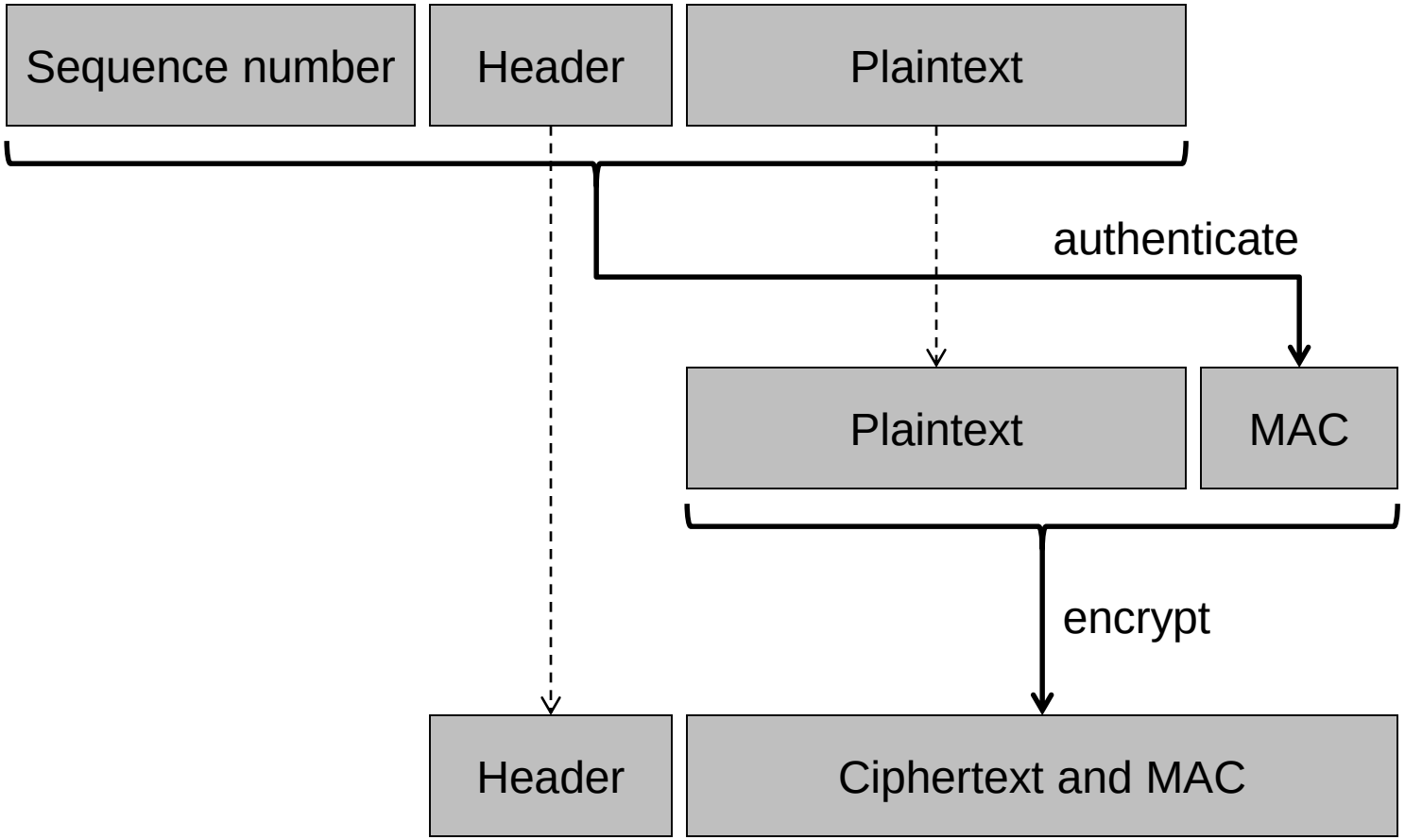

- Perfect forward secrecy is given only if DH exponents are chosen dynamically per session
  - Thus ephemeral („kurzlebig“)
- Another variant is to use a static $g^a=X$ to save computation time (static/fixed DH)
  - Does not provide perfect forward secrecy

# Encryption and Data Integrity

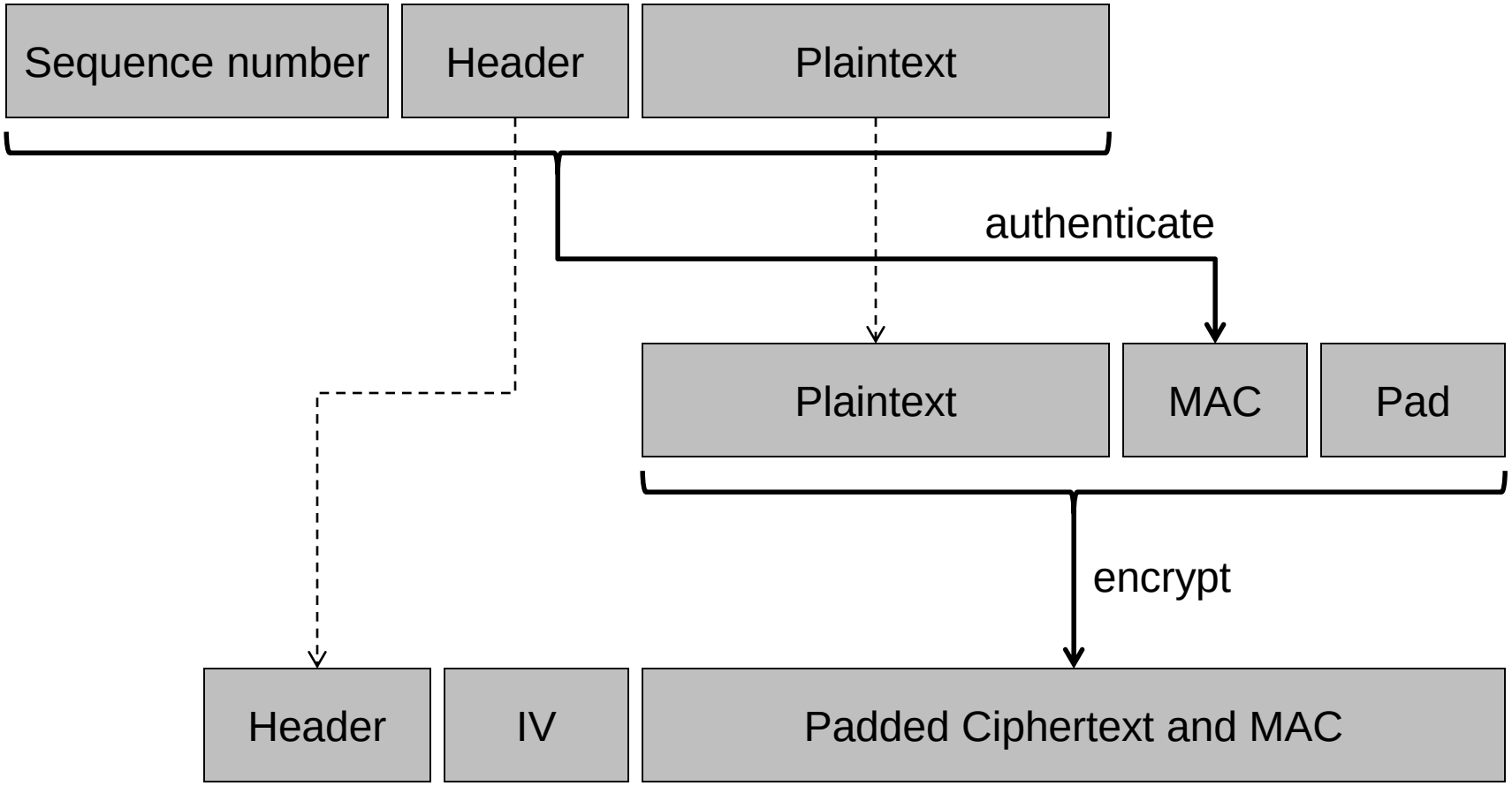



- Once handshake is finished, we can send encrypted and authenticated application data
- Structure of application data “fragment” depends on the negotiated cipher suite
  - Stream cipher, CBC block cipher, AEAD block cipher

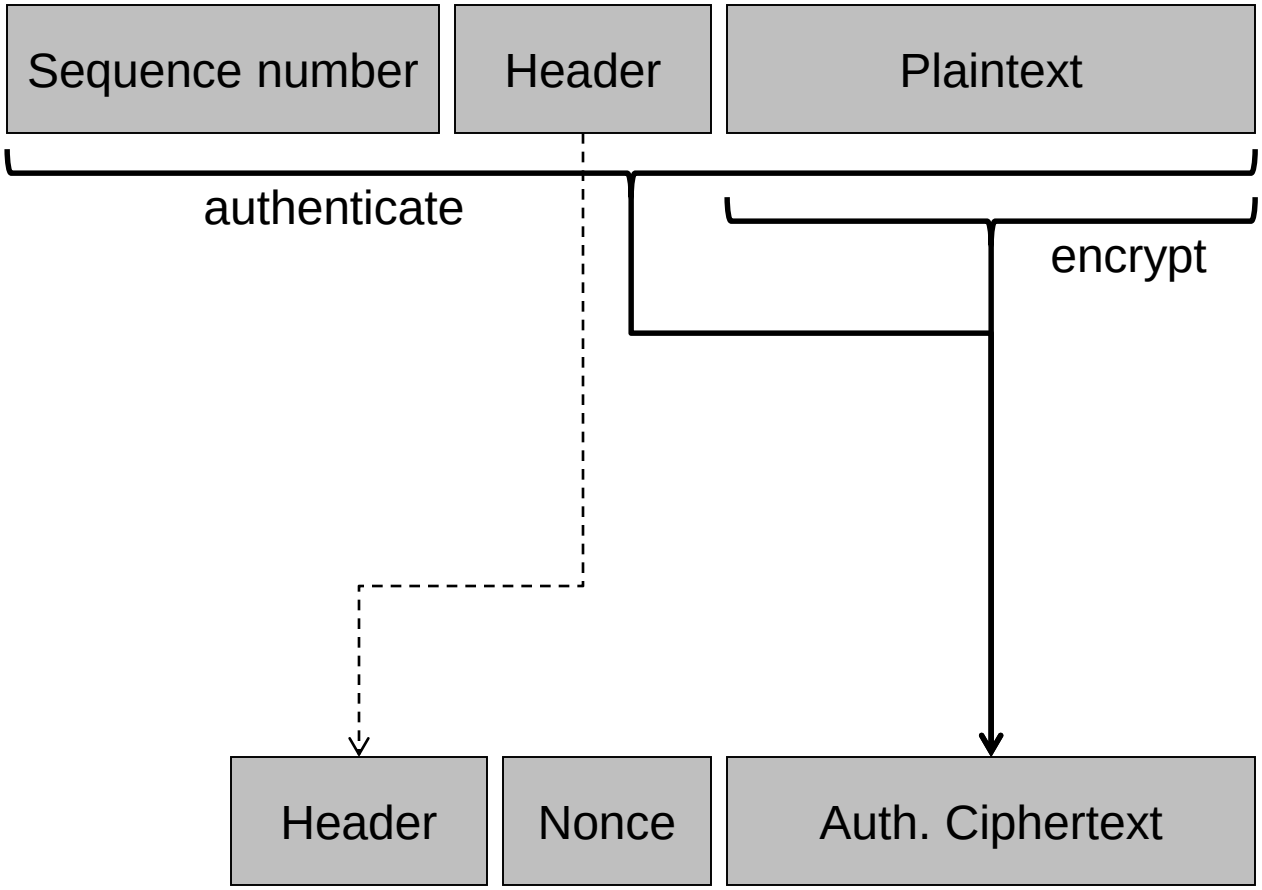
# Stream Cipher Record



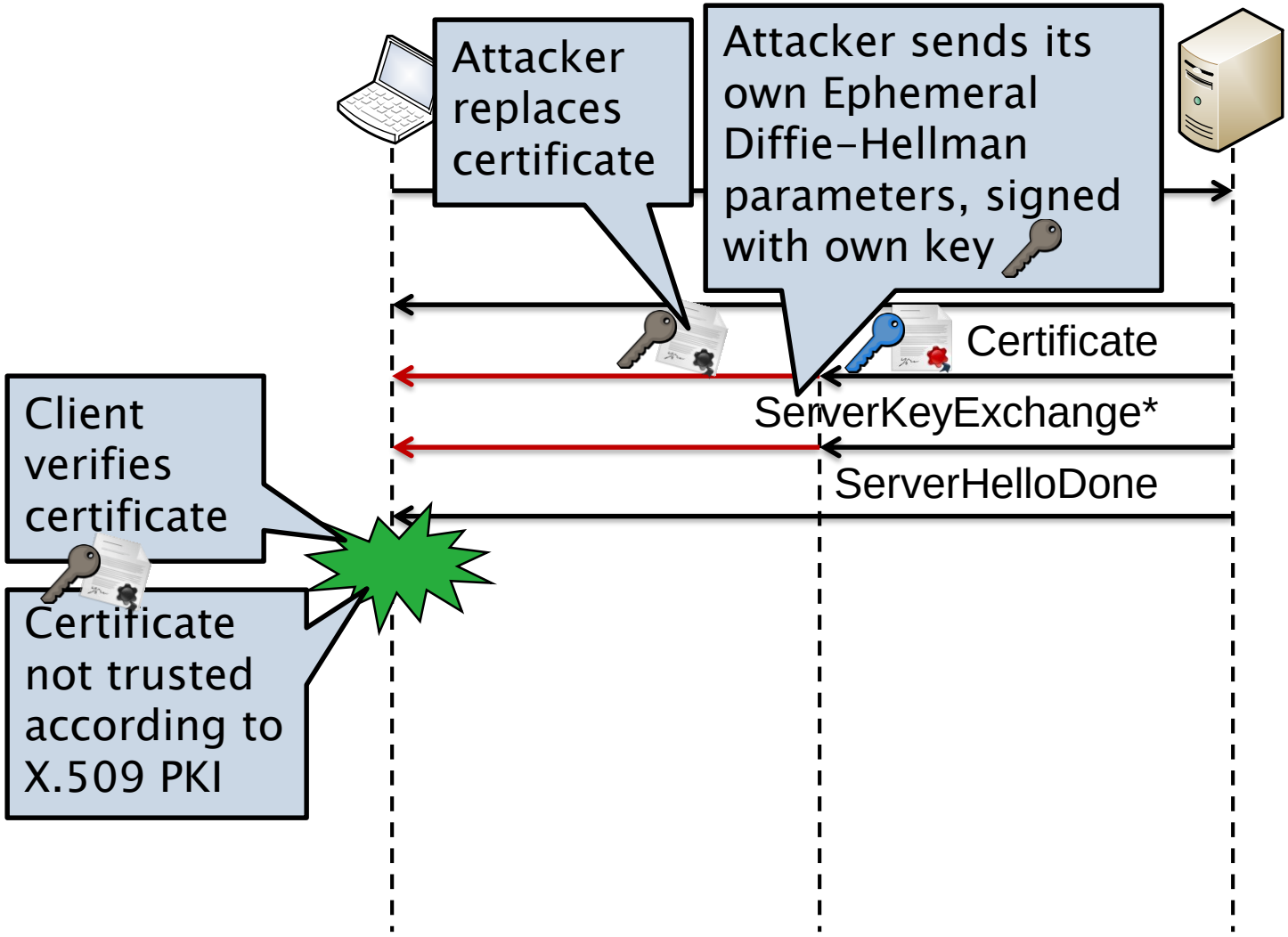
# CBC Block Cipher Record



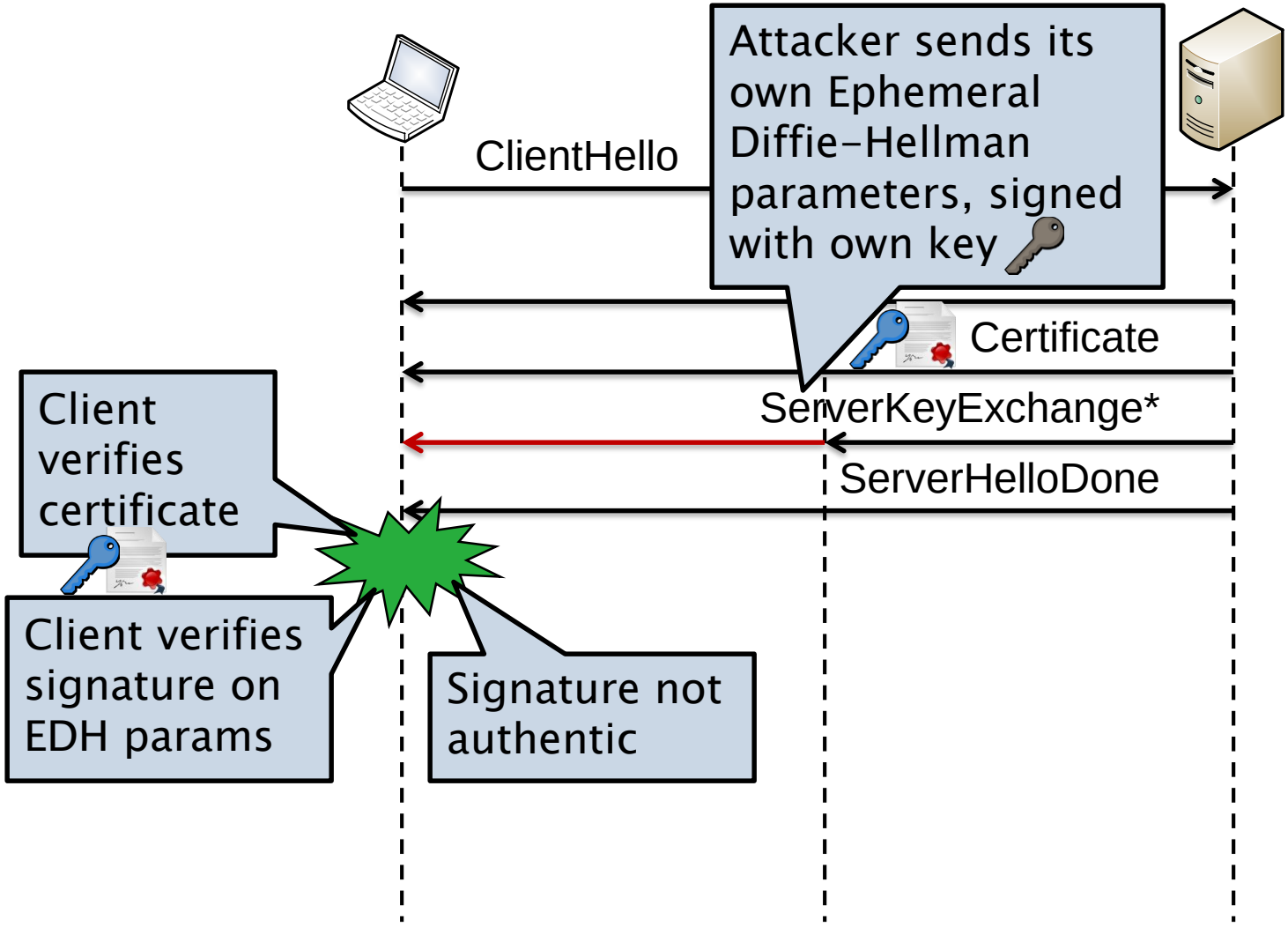
# AEAD Block Cipher Record



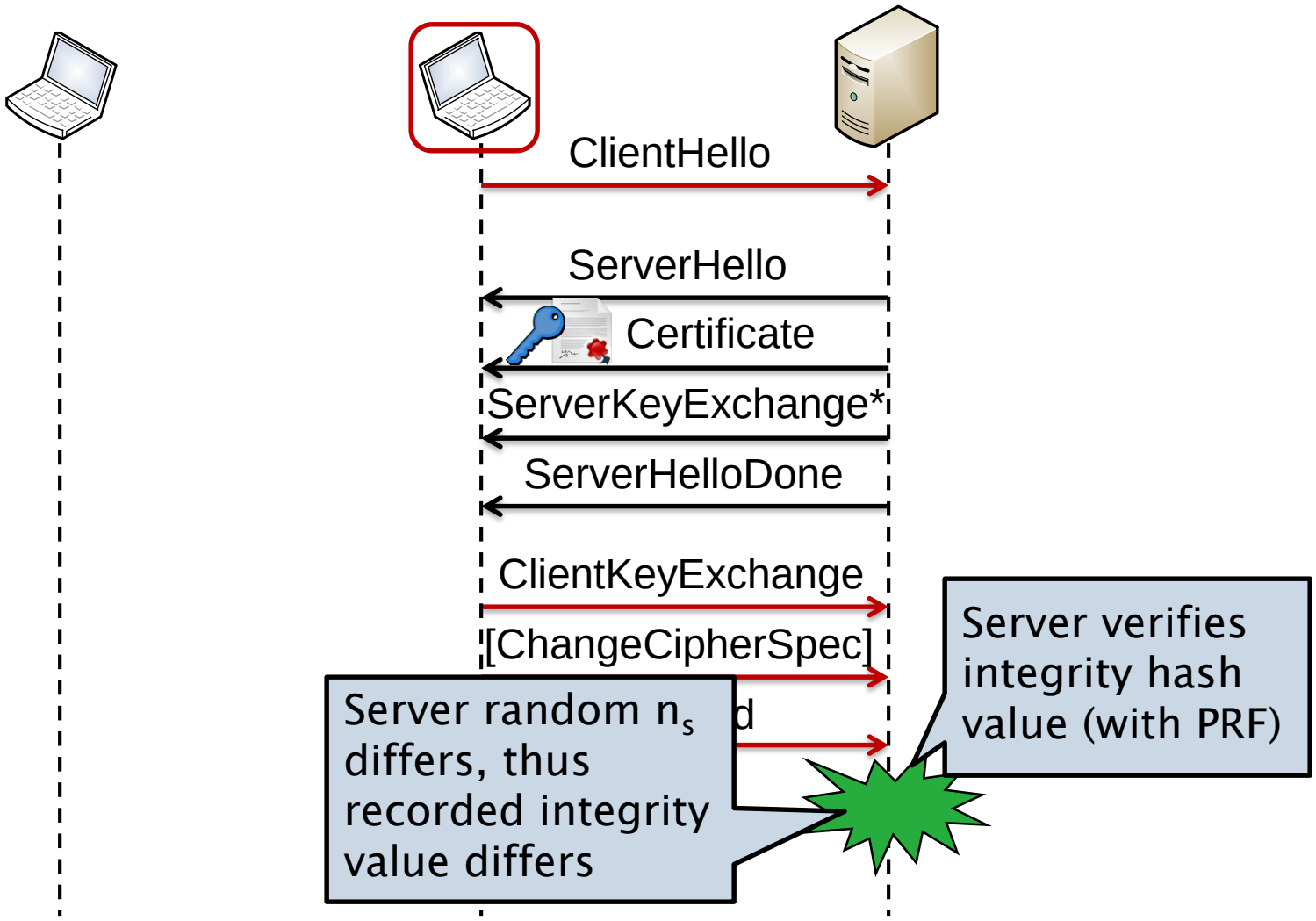
# Man-in-the-Middle Attack I



# Man-in-the-Middle Attack II



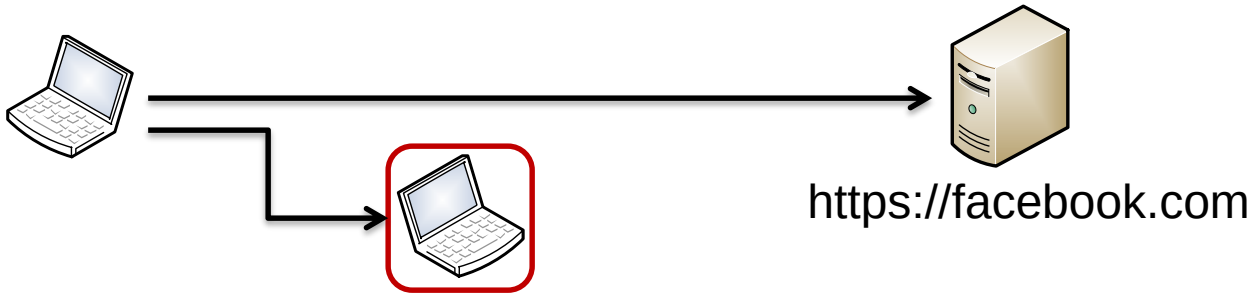
# Replay Attack



# (Some) Successful Attacks on TLS

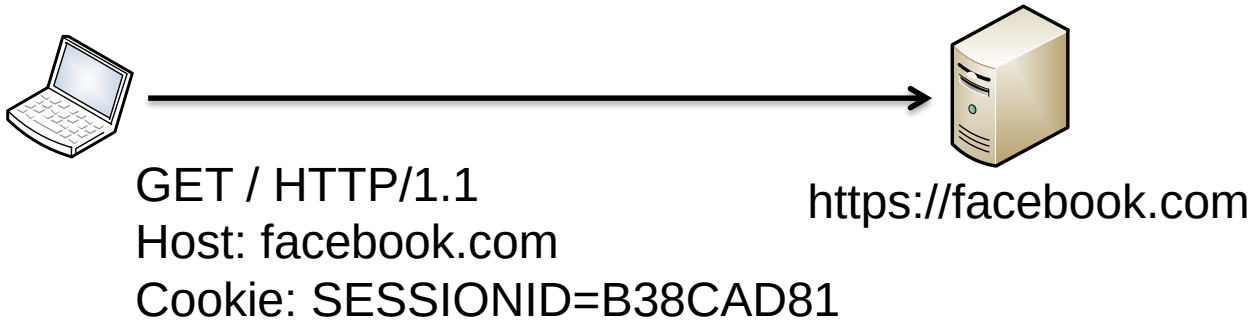
- Padding oracle attacks
  - Server reveals during decryption whether plaintext is padded correctly
  - By error message or through timing differences
  - Attacker can recover small part of plaintext by sending manipulated ciphertexts
  - Example attacks: BEAST, Lucky 13, POODLE
  - Mitigation: avoid CBC mode, avoid MAC-then-encrypt
- Downgrade attacks: force old version of TLS
- Compression attacks, e.g. CRIME

# CRIME Attack




- Scenario: user accesses website over HTTPS
  - User also accesses attacker website
- TLS compression is used
- Attacker watches encrypted traffic (public WiFi)
- JavaScript on attacker website

# CRIME Attack (2)




- Objective: determine secret cookie
- Attacker generates HTTP requests
  - <img src="https://facebook.com/SESSIONID=A" alt="">
  - <img src="https://facebook.com/SESSIONID=B" alt="">

# CRIME Attack (3)

https://facebook.com

```
GET /SESSIONID=A HTTP/1.1
Host: facebook.com
Cookie: SESSIONID=B38CAD81
```

Compressed size: 66 bytes

```
GET /SESSIONID=B HTTP/1.1
Host: facebook.com
Cookie: SESSIONID=B38CAD81
```

Compressed size: 65 bytes

- Attacker guesses secret cookie characters
- Watches size of encrypted, compressed query
  - Message is shorter ⇨ character guessed

# Summary

- TLS is a general purpose cryptographic protocol
  - Originally for HTTPS, but also used for other protocols
  - Cipher suite negotiation
- Security goals:
  - Authenticity/integrity
  - Confidentiality
  - Perfect forward secrecy with ephemeral DH
- X.509 public key infrastructure for certificates
- Outlook: TLS 1.3 provides shorter handshake