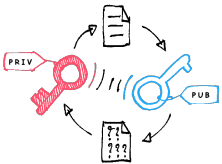
Verschlüsselung, Signatur und Co.

Wie funktioniert's in der Praxis und welche Risiken gibt es?

Prof. Dr.-Ing. Andreas Noack
<andreas.noack@hochschule-stralsund.de>

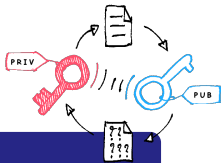
Klassische Kryptographie

Kryptographie ist die **Lehre der Verschlüsselung** bei der es darum geht, den Inhalt von etwas geschriebenem nur gewissen Personen zugänglich zu machen.



Klassische Kryptographie

Kryptographie ist die **Lehre der Verschlüsselung** bei der es darum geht, den Inhalt von etwas geschriebenem nur gewissen Personen zugänglich zu machen.



Historische Verwendung von Kryptographie

Die Kryptographie existiert schon lange, erste Ansätze gab es bereits **2000 Jahre v.Chr.**

- Die Ägypter verwendeten spezielle **Hieroglyphen**, um Informationen zu verbergen.
- Die **Skytala** von Sparta erlaubte es den Griechen, Informationen zu verschlüsseln, indem eine aufgewickelte Papyrusrolle quer beschrieben wurde.
- Cäsar verwendete die **Cäsar-Chiffre**, die Buchstaben durch andere Buchstaben ersetzt hat (z.B. A $\rightarrow$ D).

Wie wird Kryptographie eingesetzt?

Verschiedene Anwendungsfälle für Kryptographie

- **Authentifikation**. Der Nachweis, dass Daten von einer bestimmten Person stammen (*digitale Signatur*) oder der Beweis einer Identität, z.B. durch ein *digitales Zertifikat*.

Wie wird Kryptographie eingesetzt?

Verschiedene Anwendungsfälle für Kryptographie

- **Authentifikation**. Der Nachweis, dass Daten von einer bestimmten Person stammen (*digitale Signatur*) oder der Beweis einer Identität, z.B. durch ein *digitales Zertifikat*.
- **Schlüsselaustausch**. Vereinbarung eines geheimen Schlüssels über einen *unsicheren Kanal* ($\rightarrow$ Internet).

Wie wird Kryptographie eingesetzt?

Verschiedene Anwendungsfälle für Kryptographie

- **Authentifikation**. Der Nachweis, dass Daten von einer bestimmten Person stammen (*digitale Signatur*) oder der Beweis einer Identität, z.B. durch ein *digitales Zertifikat*.
- **Schlüsselaustausch**. Vereinbarung eines geheimen Schlüssels über einen *unsicheren Kanal* ($\rightarrow$ Internet).
- **Transportverschlüsselung**. Die Verschlüsselung von Kommunikationswegen ($\rightarrow$ https).

Wie wird Kryptographie eingesetzt?

Verschiedene Anwendungsfälle für Kryptographie

- **Authentifikation**. Der Nachweis, dass Daten von einer bestimmten Person stammen (*digitale Signatur*) oder der Beweis einer Identität, z.B. durch ein *digitales Zertifikat*.
- **Schlüsselaustausch**. Vereinbarung eines geheimen Schlüssels über einen *unsicheren Kanal* ($\rightarrow$ Internet).
- **Transportverschlüsselung**. Die Verschlüsselung von Kommunikationswegen ($\rightarrow$ https).
- **Ende-zu-Ende-Verschlüsselung**. Die *durchgängige* Verschlüsselung zwischen Quelle und Ziel einer Kommunikation.

Wie wird Kryptographie eingesetzt?

Verschiedene Anwendungsfälle für Kryptographie

- **Authentifikation**. Der Nachweis, dass Daten von einer bestimmten Person stammen (*digitale Signatur*) oder der Beweis einer Identität, z.B. durch ein *digitales Zertifikat*.
- **Schlüsselaustausch**. Vereinbarung eines geheimen Schlüssels über einen *unsicheren Kanal* ($\rightarrow$ Internet).
- **Transportverschlüsselung**. Die Verschlüsselung von Kommunikationswegen ($\rightarrow$ https).
- **Ende-zu-Ende-Verschlüsselung**. Die *durchgängige* Verschlüsselung zwischen Quelle und Ziel einer Kommunikation.
- **Datenverschlüsselung**. Die Verschlüsselung von *persistenten* Daten, z.B. Festplattenverschlüsselung.

Einordnung der Kryptographie

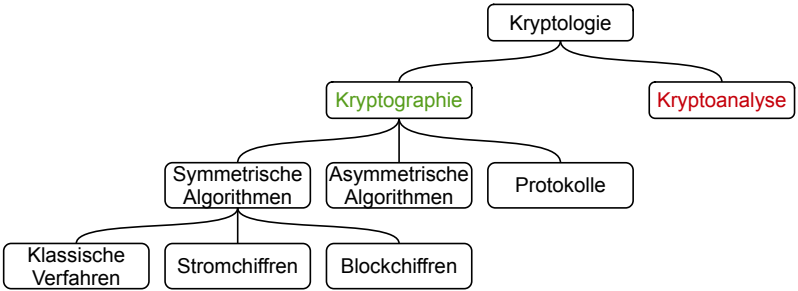


Abbildung: Einordnung der Kryptographie

Die **Kryptographie** beschäftigt sich mit der Verschlüsselung (und Absicherung) von Informationen, wohingegen sich die **Kryptoanalyse** mit dem Brechen von Kryptosystemen beschäftigt.

Symmetrische Algorithmen

Der **Kryptographie** untergeordnet sind drei verschiedene Techniken:

Symmetrische Algorithmen. *Verschlüsselungsverfahren (gen/enc/dec), die für die Ver- und Entschlüsselung den selben Schlüssel **k** verwenden.*

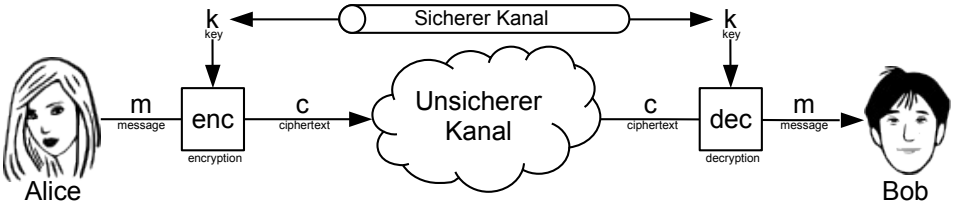


Abbildung: Symmetrische Verschlüsselung

*Den Algorithmus zur Erstellung der Schlüssel (hier **k**) nennt man gen.*

Symmetrische Algorithmen II

Unterteilung der symmetrischen Algorithmen

Die symmetrischen Algorithmen lassen sich wiederum unterteilen in:

1. **Klassische Verfahren**. Historische Kryptographie (z.B. Cäsar-Chiffre).
2. **Stromchiffren**. *Moderne kryptographische* Algorithmen, die einzelne Bits verschlüsseln. XOR-Verknüpfung (*exklusives Oder*) eines generierten Schlüsselstroms mit Klartext *(beide haben die gleiche Länge)*.
3. **Blockchiffren**. *Moderne kryptographische* Algorithmen, die gesamte Blöcke (z.B. 16 Byte) verschlüsseln. Wenn die Blocklänge nicht ausgereizt wird, werden die restlichen Bits aufgefüllt (*Padding*).

Symmetrische Algorithmen II

Unterteilung der symmetrischen Algorithmen

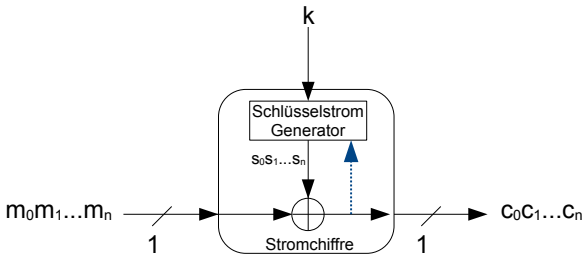
Die symmetrischen Algorithmen lassen sich wiederum unterteilen in:

1. **Klassische Verfahren**. Historische Kryptographie (z.B. Cäsar-Chiffre).
2. **Stromchiffren**. *Moderne kryptographische* Algorithmen, die einzelne Bits verschlüsseln. XOR-Verknüpfung (*exklusives Oder*) eines generierten Schlüsselstroms mit Klartext *(beide haben die gleiche Länge)*.
3. **Blockchiffren**. *Moderne kryptographische* Algorithmen, die gesamte Blöcke (z.B. 16 Byte) verschlüsseln. Wenn die Blocklänge nicht ausgereizt wird, werden die restlichen Bits aufgefüllt (*Padding*).

Beispiele für Chiffren

- **Stromchiffren**: (L)FSR, RC4, A5, Salsa20, ...
- **Blockchiffren**: (3)DES, IDEA, AES, Blowfish, Twofish, ...

Stromchiffren



Das Blockschaltbild einer **Stromchiffre**.
Stromchiffren verschlüsseln **beliebig** viele Bits und sind oft besonders **performant**.

Definition (Stromchiffre)

Eine **Stromchiffre** verschlüsselt Bits **einzeln**. Dazu wird ein Schlüsselstrom ($s_0 s_1 \ldots s_n$) generiert, der **XOR** mit dem Klartext verknüpft wird *(bitweise Addition)*.

Blockchiffren

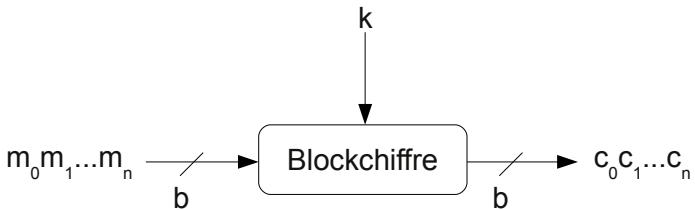


Abbildung: Allgemeines Schaltbild einer Blockchiffre (Blockgröße: b Bits)

Definition (Blockchiffre)

Eine **Blockchiffre** verschlüsselt Blöcke **fester Länge**, z.B. 64 oder 128 Bit. Der Klartext wird zerlegt und danach **Block für Block** verschlüsselt. Wenn der Klartext den letzten Block nicht vollständig ausfüllt, wird dieser aufgefüllt (*Padding*).

Betriebsmodi von Blockchiffren (Operation Modes)

Die **Betriebsmodi** von **Blockchiffren** beschreiben, wie ein Klartext bestehend aus **mehreren Blöcken** verschlüsselt wird.

Betriebsmodi von Blockchiffren (Operation Modes)

Die **Betriebsmodi** von **Blockchiffren** beschreiben, wie ein Klartext bestehend aus **mehreren Blöcken** verschlüsselt wird.

Wie unterscheiden sich verschiedene Betriebsmodi?

- Abhängigkeit der Verschlüsselung eines Blocks von anderen Blöcken
- Schutz der Integrität (z.B. Inhalte oder Reihenfolge der Blöcke)
- Möglichkeit, einzelne Bits zu verschlüsseln (≈Stromchiffre)

Betriebsmodi von Blockchiffren (Operation Modes)

Die **Betriebsmodi** von **Blockchiffren** beschreiben, wie ein Klartext bestehend aus **mehreren Blöcken** verschlüsselt wird.

Wie unterscheiden sich verschiedene Betriebsmodi?

- Abhängigkeit der Verschlüsselung eines Blocks von anderen Blöcken
- Schutz der Integrität (z.B. Inhalte oder Reihenfolge der Blöcke)
- Möglichkeit, einzelne Bits zu verschlüsseln (≈Stromchiffre)

Wir beginnen mit dem naheliegendsten Ansatz! Dieser wird als **Electronic Codebook Mode (ECB)** bezeichnet.

Electronic Codebook Mode (ECB)

Definition (Electronic Codebook Mode (ECB))

Jeder Block wird **unabhängig** voneinander verschlüsselt.
Verschlüsselung eines Blocks m_i für alle $i \geq 0$: $c_i = enc_k(m_i)$

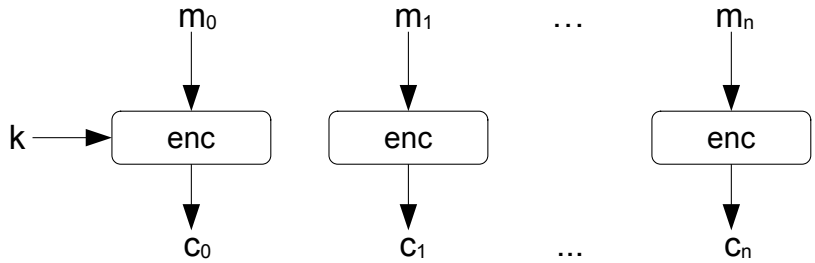


Abbildung: Verschlüsselung mit ECB

Eigenschaft: Die Verschlüsselung mehrerer Blöcke mit dem gleichen Inhalt führt zu identischen Chiffreblöcken.

Cipher Block Chaining Mode (CBC)

Definition (Cipher Block Chaining Mode (CBC))

XOR-Verknüpfung des Klartextblocks mit **vorherigem** Chiffreblock, **dann** Verschlüsselung.
Für den **ersten** Block wird ein zufälliger **Initialisierungsvektor (IV)** genutzt, dieser wird oft gemeinsam mit dem Chiffretext versendet (allerdings unverschlüsselt).

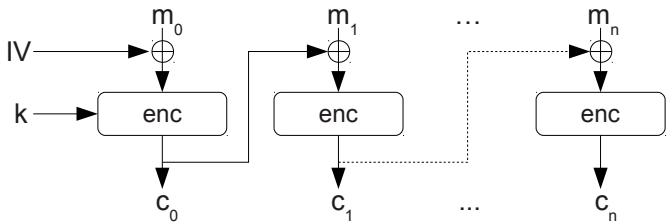


Abbildung: Verschlüsselung mit CBC

Verschlüsselung:

- $c_0 = enc_k(m_0 \oplus IV)$
- $c_i = enc_k(m_i \oplus c_{i-1})$

Counter Mode (CTR)

Definition (Counter Mode (CTR))

XOR-Verknüpfung des Klartextblocks mit **Verschlüsselung** eines **Counters** (Zähler). Initial besteht der Counter oft aus einem **festgelegten Zufallswert**, der für jede Operation **inkrementiert** wird. Der CTR-Modus erzeugt eine **Stromchiffre**.

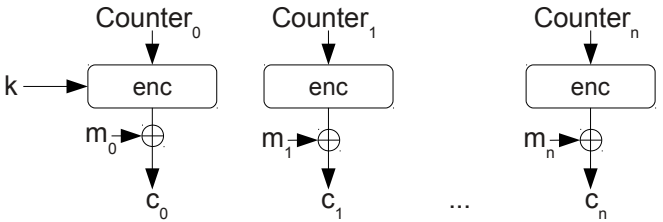


Abbildung: Verschlüsselung und Entschlüsselung mit CTR

Verschlüsselung:

- $c_i = enc_k(counter_i) \oplus m_i$

Entschlüsselung:

- $m_i = \mathbf{enc}_k(counter_i) \oplus c_i$

Asymmetrische Algorithmen

Asymmetrische Algorithmen. *Verschlüsselungsverfahren (gen/enc/dec), die für die Ver- und Entschlüsselung unterschiedliche Schlüssel **e** und **d** verwenden.*
*Asymmetrische Algorithmen sind **erheblich** langsamer als symmetrische Algorithmen!*

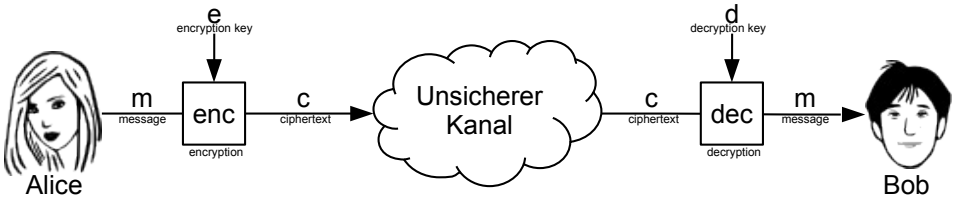


Abbildung: Asymmetrische Verschlüsselung

Asymmetrische Algorithmen

Asymmetrische Algorithmen. *Verschlüsselungsverfahren (gen/enc/dec), die für die Ver- und Entschlüsselung unterschiedliche Schlüssel **e** und **d** verwenden.*
*Asymmetrische Algorithmen sind **erheblich** langsamer als symmetrische Algorithmen!*

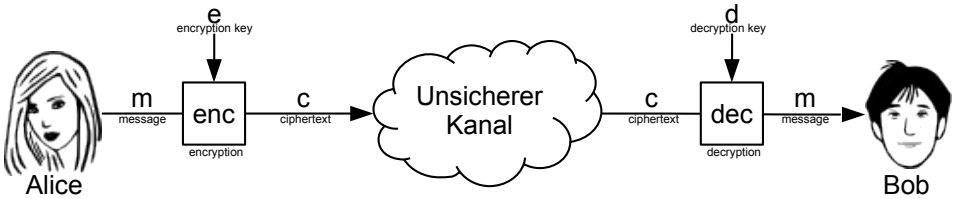


Abbildung: Asymmetrische Verschlüsselung

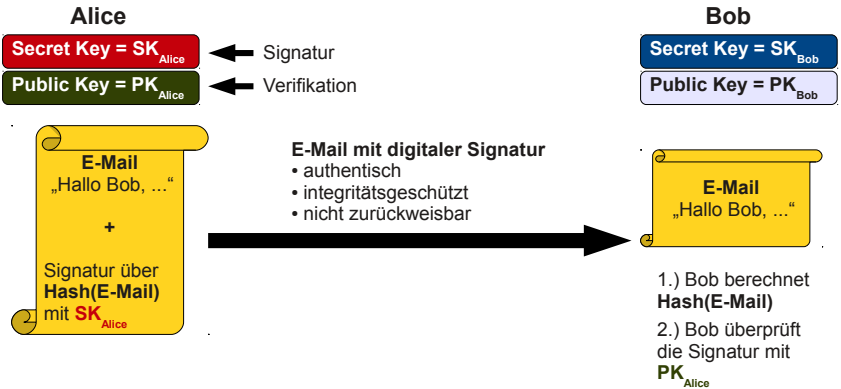
Beispiele für asymmetrische Algorithmen

- **Chiffren**: RSA, ElGamal, ECIES, ...
- **Signaturverfahren**: RSA, DSA, ECDSA, ...
- **Schlüsselaustausch**: DH, ECDH(E), BD1, BD2, TBKA, ...

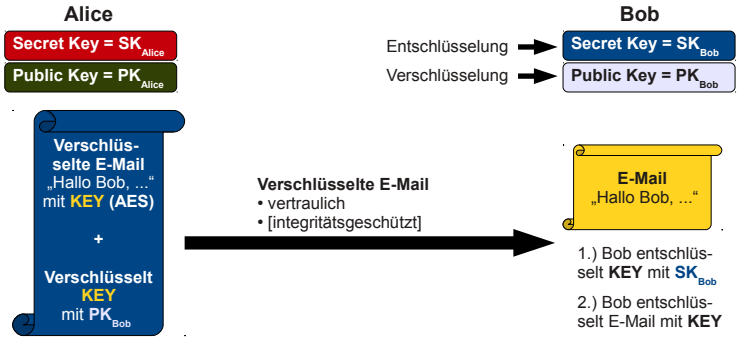
Asymmetrische Algorithmen: E-Mails signieren

Digitale Schlüsselpaare

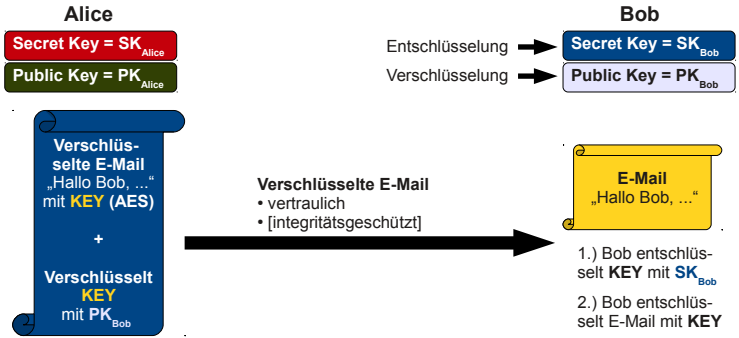
Jeder Nutzer verfügt über ein Schlüsselpaar (SK_{Nutzer}, PK_{Nutzer}).
Der *SK* (**Secret Key**) bleibt geheim und der *PK* (**Public Key**) ist öffentlich.



Asymmetrische Algorithmen: E-Mails verschlüsseln



Asymmetrische Algorithmen: E-Mails verschlüsseln



Hybride Verschlüsselung

Hybride Verschlüsselung. Die E-Mail wird mit einer **schnellen** symmetrischen Chiffre (z.B. AES) und einem zufälligen Schlüssel **KEY** verschlüsselt.

KEY wird asymmetrisch mit $PK_{\text{Gegenseite}}$ verschlüsselt.

Asymmetrische Algorithmen: Elliptische Kurven

Elliptic Curve Cryptography (ECC)

Mit elliptischen Kurven (EC) können **asymmetrische Kryptosysteme** mit den Sicherheitseigenschaften der klassischen Verfahren (RSA, DL-based) erzeugt werden.

Asymmetrische Algorithmen: Elliptische Kurven

Elliptic Curve Cryptography (ECC)

Mit elliptischen Kurven (EC) können **asymmetrische Kryptosysteme** mit den Sicherheitseigenschaften der klassischen Verfahren (RSA, DL-based) erzeugt werden.

Der Vorteil von ECC liegt dabei in einer **deutlich kürzeren Schlüssellänge** bei äquivalenter Sicherheit im Vergleich mit klassischen Verfahren ($\approx 160 - 256$ Bit vs. $1024 - 3072$ Bit).

Blockchiffre	**MAC**	**RSA**	**DH $\mathbb{F}_p$**	**ECDH**	**ECDSA**
128	128	3000	3000	250	250

Empfehlungen des Bundesamt für Sicherheit in der Informationstechnik (BSI) für die Jahre 2023+ in Bit [5]

Kryptographische Protokolle

Protokolle. *Austausch von mehreren kryptographischen Nachrichten in einer festgelegten Form. Beispiele: Sicherer Schlüsselaustausch oder Authentifikation.*

Kryptographische Protokolle

Protokolle. *Austausch von mehreren kryptographischen Nachrichten in einer festgelegten Form. Beispiele: Sicherer Schlüsselaustausch oder Authentifikation.*

Kryptographische Protokolle

Kryptographische Protokolle tauschen Informationen zwischen mehreren Parteien in einer festgelegten Form aus. Sie nutzen bekannte **symmetrische** und/oder **asymmetrische** Algorithmen als Bausteine, die kombiniert neue Funktionen umsetzen, z.B. einen *authentifizierten Schlüsselaustausch* wie in TLS/SSL oder SSH.

Kryptographische Protokolle

Protokolle. *Austausch von mehreren kryptographischen Nachrichten in einer festgelegten Form. Beispiele: Sicherer Schlüsselaustausch oder Authentifikation.*

Kryptographische Protokolle

Kryptographische Protokolle tauschen Informationen zwischen mehreren Parteien in einer festgelegten Form aus. Sie nutzen bekannte **symmetrische** und/oder **asymmetrische** Algorithmen als Bausteine, die kombiniert neue Funktionen umsetzen, z.B. einen *authentifizierten Schlüsselaustausch* wie in TLS/SSL oder SSH.

Beispiele für kryptographische Protokolle

- TLS (vormals SSL), SSH, OAuth, Kerberos, Signal (X3DH), ...

Public Key Infrastruktur (PKI)

Im Internet werden mittlerweile viele Webseiten

sicher via `https://` ausgeliefert.

Public Key Infrastruktur (PKI)

Im Internet werden mittlerweile viele Webseiten

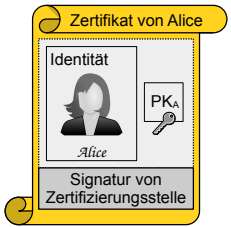
sicher via `https://` ausgeliefert.

Sicherheitsziel: Entity Authentication

Wie erkennt man, dass man mit dem richtigen Server und nicht mit einem Angreifer spricht?

- Der Server authentifiziert sich mit einem digitalen **Zertifikat**!
- Das Zertifikat ist Teil einer **Public Key Infrastructure (PKI)**.

Digitale Zertifikate

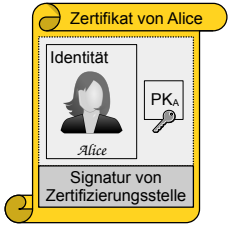


Digitales Zertifikat

- **Identität**: Name oder FQDN für Server + *weitere Informationen wie Organisation, E-Mail, ...*
- **Public Key**: RSA, ECDSA
- **Digitale Signatur** von Zertifizierungsstelle (CA) oder übergeordneter Instanz, Gültigkeit (Datum)

Ein übliches Format für digitale Zertifikate ist X.509

Digitale Zertifikate



Digitales Zertifikat

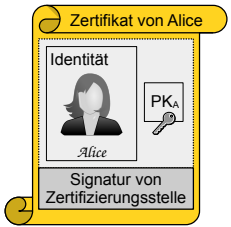
- **Identität**: Name oder FQDN für Server + *weitere Informationen wie Organisation, E-Mail, ...*
- **Public Key**: RSA, ECDSA
- **Digitale Signatur** von Zertifizierungsstelle (CA) oder übergeordneter Instanz, Gültigkeit (Datum)

Ein übliches Format für digitale Zertifikate ist X.509

Wie funktioniert eine Authentifikation mit Zertifikat?

Alice besitzt ein **Zertifikat** und einen geheimen **Secret Key**: $Cert_A := \left(ID_A, PK_A, sig_{CA}(ID_A, PK_A)\right)$ und SK_A

Digitale Zertifikate



Digitales Zertifikat

- **Identität**: Name oder FQDN für Server + *weitere Informationen wie Organisation, E-Mail, ...*
- **Public Key**: RSA, ECDSA
- **Digitale Signatur** von Zertifizierungsstelle (CA) oder übergeordneter Instanz, Gültigkeit (Datum)

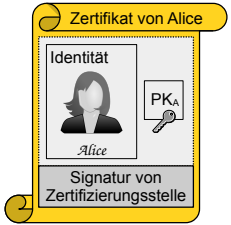
Ein übliches Format für digitale Zertifikate ist X.509

Wie funktioniert eine Authentifikation mit Zertifikat?

Alice besitzt ein **Zertifikat** und einen geheimen **Secret Key**: $Cert_A := \left(ID_A, PK_A, sig_{CA}(ID_A, PK_A)\right)$ und SK_A
Folgende Schritte für **Entity Authentication** (Alice→Bob):

1. Alice **sendet** $Cert_A$ an Bob, dieser **prüft** ob die Signatur der CA stimmt

Digitale Zertifikate



Digitales Zertifikat

- **Identität**: Name oder FQDN für Server + *weitere Informationen wie Organisation, E-Mail, ...*
- **Public Key**: RSA, ECDSA
- **Digitale Signatur** von Zertifizierungsstelle (CA) oder übergeordneter Instanz, Gültigkeit (Datum)

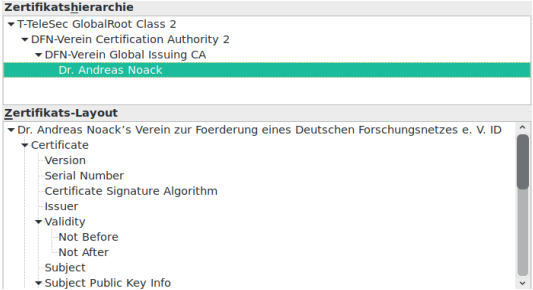
Ein übliches Format für digitale Zertifikate ist X.509

Wie funktioniert eine Authentifikation mit Zertifikat?

Alice besitzt ein **Zertifikat** und einen geheimen **Secret Key**: $Cert_A := \left(ID_A, PK_A, sig_{CA}(ID_A, PK_A)\right)$ und SK_A
Folgende Schritte für **Entity Authentication** (Alice→Bob):

1. Alice **sendet** $Cert_A$ an Bob, dieser **prüft** ob die Signatur der CA stimmt
2. Alice führt eine **Operation** mit ihrem SK_A durch, z.B.:
   - Alice signiert: sig_{SK_A}(Zufall), Bob verifiziert mit PK_A **oder**
   - Bob verschlüsselt enc_{PK_A}(Zufall), Alice entschlüsselt korrekt mit SK_A

Public Key Infrastructure



Zertifikatshierarchie für Dr. Andreas Noack

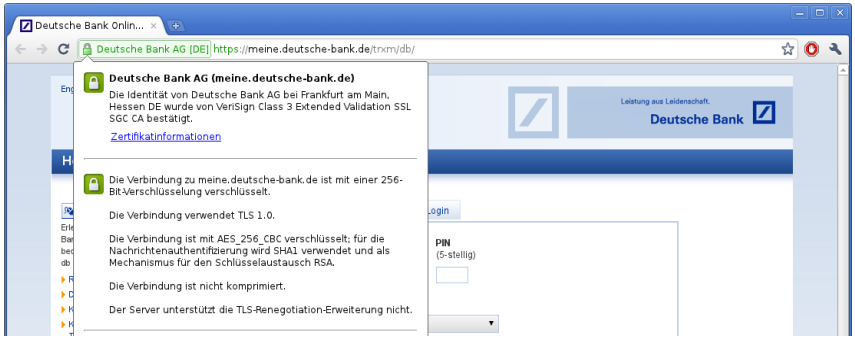
- Nicht jeder kennt den *PK* von "DFN-Verein Global Issuing CA", aber den *PK* von "T-TeleSec GlobalRoot Class 2" ($\rightarrow$ *im OS/Browser gepeichert*).
- Im Zertifikat sind enthalten:
  1. Signatur von "T-TeleSec GlobalRoot Class 2" für Zertifikat von "DFN-Verein Certification Authority 2"
  2. Signatur von "DFN-Verein Certification Authority 2" für Zertifikat von "DFN-Verein Global Issuing CA"
  3. Signatur von "DFN-Verein Global Issuing CA" für "Dr. Andreas Noack"

Jura: Digitale Zertifikate vs. Elektronische Signatur

Bisher haben wir ausschließlich technische Aspekte kennengelernt, nun kommen einige rechtliche hinzu...

	Beschreibung	**Juristische Bedeutung**
Elektronische Signatur	**Keine digitale Signatur!** Ein einfacher Text: *Vorname Name (+Adresse)* unter einem digitalen Dokument	Keine rechtliche Bindung
Fortgeschrittene Signatur	Digitale Signatur! CA ist **kein** vom BSI akkreditierter Anbieter	Keine rechtliche Bindung, für formfreie Vereinbarungen geeignet
Qualifizierte Signatur	Digitale Signatur! CA ist **akkreditierter** Anbieter, eine sichere Signaturerstellungseinheit (SEEE, Hardware oder Software) ist erforderlich	Kann Schriftform zur Korrespondenz (z.B. mit Ämtern) ersetzen.

Transport Layer Security (TLS)

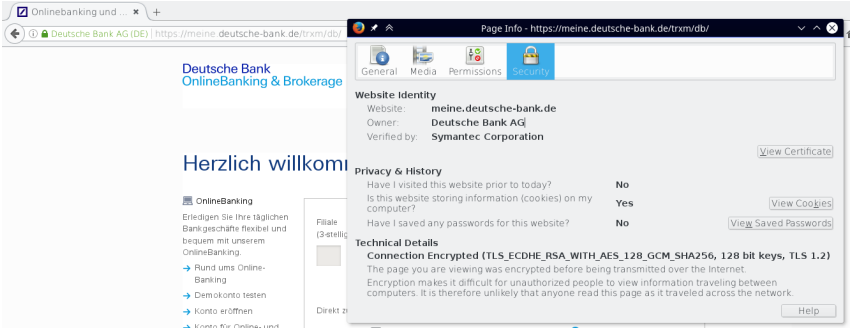


Quelle: `https://meine.deutsche-bank.de`

Was ist Transport Layer Security (TLS)?

- Protokoll für **Verschlüsselung**, **Daten**- und **Teilnehmerauthentifizierung**.
- TLS unterstützt eine Vielzahl von kryptographischen Algorithmen (*Ciphersuites*) zur Verschlüsselung und Authentifizierung (Daten/Teilnehmer).
- TLS setzt eine zuverlässige Verbindung (≈TCP) voraus, z.B. https.
- TLS 1.0 ≈ SSL 3.1 (*Secure Socket Layer*).

Transport Layer Security (TLS)



Quelle: `https://meine.deutsche-bank.de` (02/2017)

Was ist Transport Layer Security (TLS)?

- Protokoll für **Verschlüsselung**, **Daten**- und **Teilnehmerauthentifizierung**.
- TLS unterstützt eine Vielzahl von kryptographischen Algorithmen (*Ciphersuites*) zur Verschlüsselung und Authentifizierung (Daten/Teilnehmer).
- TLS setzt eine zuverlässige Verbindung (≈TCP) voraus, z.B. https.
- TLS 1.0 ≈ SSL 3.1 (*Secure Socket Layer*). Aktuell ist TLS 1.3 (RFC 8446)

Die Protokolleigenschaften von TLS und DTLS

- **Entity Authentication**: Client ← Server
  mit X.509 Zertifikat und Nachweis des Besitzes
- **Entity Authentication**: Client → Server
  optional, ebenfalls mit X.509 Zertifikat
- **Schlüsselaustausch**: Client ↔ Server
  mit oder ohne (Perfect) Forward Secrecy
- **Verschlüsselte** und **integritätsgeschützte** Übertragung von Daten: Client ↔ Server
  TLS ermöglicht Komprimierung

Die Protokolleigenschaften von TLS und DTLS

- **Entity Authentication**: Client ← Server
  mit X.509 Zertifikat und Nachweis des Besitzes
- **Entity Authentication**: Client → Server
  optional, ebenfalls mit X.509 Zertifikat
- **Schlüsselaustausch**: Client ↔ Server
  mit oder ohne (Perfect) Forward Secrecy
- **Verschlüsselte** und **integritätsgeschützte** Übertragung von Daten: Client ↔ Server
  TLS ermöglicht Komprimierung
- **Ciphersuite** ist aushandelbar: Client ⇔ Server
  Unterschiedliche kryptographische Verfahren
- **Sitzungswiederaufnahme** und **Schlüsselerneuerung**
  Zurückliegende Sitzungen können wieder aufgenommen werden, Sitzungsschlüssel werden nach gewisser Zeit erneuert.

TLS Ciphersuites

Die Chiffresammlungen: TLS-Ciphersuites

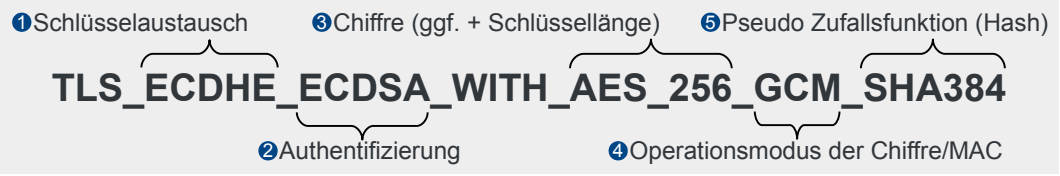
Die TLS-Ciphersuites (≈ Katalog von kryptographischen Verfahren) haben starke Auswirkungen auf den TLS-Protokollablauf:
Je nach Ciphersuite werden Schlüssel (z.B. PMS) anders berechnet und verschiedene Protokollnachrichten (z.B. ClientKeyExchange) anders belegt.

TLS Ciphersuites

Die Chiffresammlungen: TLS-Ciphersuites

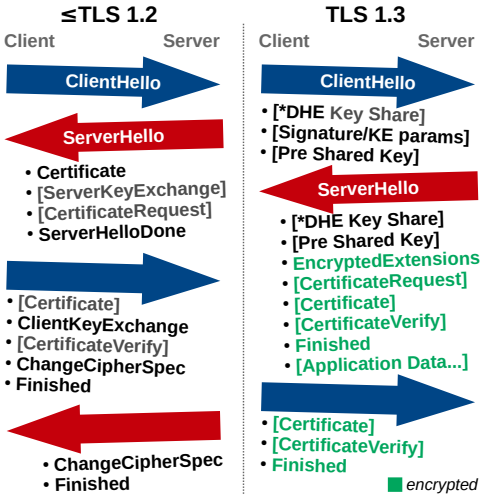
Die TLS-Ciphersuites (≈ Katalog von kryptographischen Verfahren) haben starke Auswirkungen auf den TLS-Protokollablauf:
Je nach Ciphersuite werden Schlüssel (z.B. PMS) anders berechnet und verschiedene Protokollnachrichten (z.B. ClientKeyExchange) anders belegt.

Die Namensgebung der Ciphersuites [3, 7]



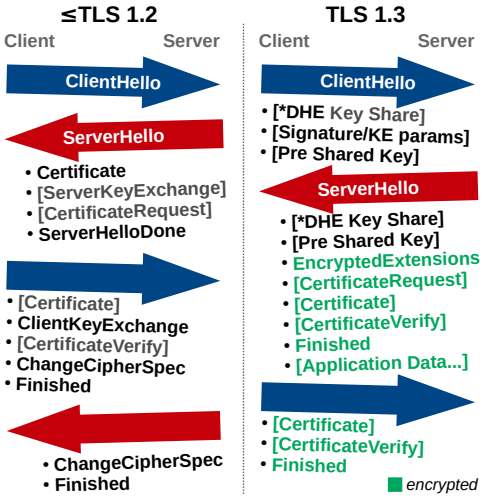
1. z.B. `ECDHE`, `DHE` (*mit Forward Secrecy*) oder `ECDH`, `DH` (*ohne FS*), ...
2. z.B. `ECDSA`, `RSA`, `DSS`, ...
3. z.B. `AES256`, `AES128`, `CAMELLIA256`, `3DES-EDE` (168 Bit), `DES` (56 Bit), ...
4. z.B. `GCM`, `CBC`, ...
5. z.B. `SHA384`, `SHA256`, `SHA`, `MD5`, ...

Was ist neu in TLS 1.3?



Vergleich der Protokollversionen TLS $\leq$1.2 und 1.3

Was ist neu in TLS 1.3?



Vergleich der Protokollversionen TLS $\leq$1.2 und 1.3

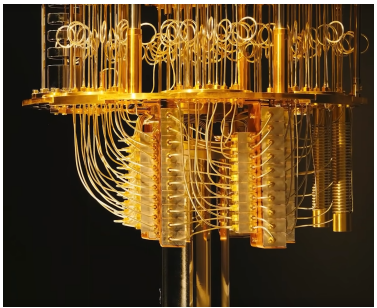
Transport Layer Security TLS 1.3 [4]

- **Forward Secrecy** ist verpflichtend ($\rightarrow$ [EC]DHE)
- **Pre Shared Key** Auth./KE möglich! $\rightarrow$ z.B. alte Session
- Handshake bereits **verschlüsselt**
- App-Data schon in Handshake (schnell+unsicher)
- keine Kompression

Was ist ein Quantencomputer?

Was ist ein Quantencomputer?

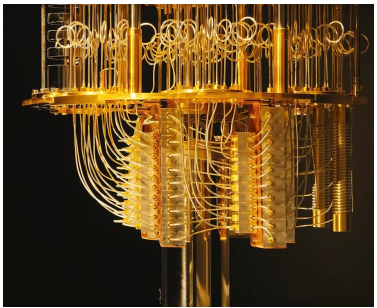
- Computer, der mit **Qubits** $|\Psi\rangle = a \cdot |0\rangle + b \cdot |1\rangle$ mit $a, b \in \mathbb{C}$ anstatt mit **Bits** $\{0, 1\} \in \mathbb{Z}$ rechnet.



Was ist ein Quantencomputer?

Was ist ein Quantencomputer?

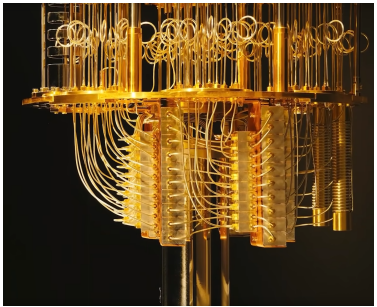
- Computer, der mit **Qubits** $|\Psi\rangle = a \cdot |0\rangle + b \cdot |1\rangle$ mit $a, b \in \mathbb{C}$ anstatt mit **Bits** $\{0, 1\} \in \mathbb{Z}$ rechnet.
- Qubits haben gleichzeitig mehrere Zustände aus $\{0, 1\}$, Einschränkung: $|a|^2 + |b|^2 = 1$. (**Superposition**)
- Nutzt ein Algorithmus mehrere Qubits (**Quantenregister**, z.B. $|00\rangle, |01\rangle, |10\rangle, |11\rangle$), können durch die **Verschränkung** von Qubits alle Linearkombinationen gleichzeitig evaluiert werden!



Was ist ein Quantencomputer?

Was ist ein Quantencomputer?

- Computer, der mit **Qubits** $|\Psi\rangle = a \cdot |0\rangle + b \cdot |1\rangle$ mit $a, b \in \mathbb{C}$ anstatt mit **Bits** $\{0, 1\} \in \mathbb{Z}$ rechnet.
- Qubits haben gleichzeitig mehrere Zustände aus $\{0, 1\}$, Einschränkung: $|a|^2 + |b|^2 = 1$. (**Superposition**)
- Nutzt ein Algorithmus mehrere Qubits (**Quantenregister**, z.B. $|00\rangle, |01\rangle, |10\rangle, |11\rangle$), können durch die **Verschränkung** von Qubits alle Linearkombinationen gleichzeitig evaluiert werden!
- Quantencomputer machen **Fehler**: Die häufigste Lösung ist das richtige Ergebnis.



Teil eines Quantencomputers (Quelle: Youtube, Clixoom)

Effizienz von Quantencomputern

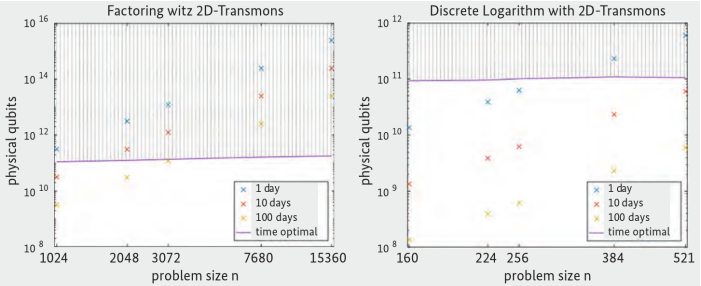


Abbildung: *Größe eines 2D-Transmonen-Quantencomputers zur Faktorisierung und Berechnung eines diskreten Logarithmus bei heutiger Fehlerrate von 1:100.*
Quelle: *BSI-Studie „Entwicklungsstand Quantencomputer“*

Quelle: BSI [6]

Wenn ein Quantencomputer mit **ausreichend Qubits** existiert! (Aktuell: 53 Qubits?)

- ...sind **RSA**, **DH**, **ECDH**, mit üblichen Bitlängen **gebrochen**! (Shor)
- ...sind die meisten **symmetrischen** Chiffren und **Hashfunktionen** nur noch **halb** so sicher. (Grover)

Post-Quanten-Kryptographie

Definition (Post-Quantum-Cryptography (PQC))

Kryptographische Verfahren, die trotz der Existenz von leistungsfähigen **Quantencomputern**, eine hohe Sicherheit bieten!

Dazu zählen vor allem **asymmetrische** kryptographische Verfahren, wie:

- Chiffren *(PKE – Public Key Encryption)*
- Schlüsselaustauschverfahren *(KEM – Key Encapsulation Mechanism)*
- Signaturverfahren

Post-Quanten-Kryptographie

Definition (Post-Quantum-Cryptography (PQC))

Kryptographische Verfahren, die trotz der Existenz von leistungsfähigen **Quantencomputern**, eine hohe Sicherheit bieten!

Dazu zählen vor allem **asymmetrische** kryptographische Verfahren, wie:

- Chiffren *(PKE – Public Key Encryption)*
- Schlüsselaustauschverfahren *(KEM – Key Encapsulation Mechanism)*
- Signaturverfahren

Standardisierung von PQC-Verfahren

Das *National Institute of Standards and Technology (NIST)* sucht seit 2016 geeignete PQC-Verfahren zur Standardisierung [1]. Für PKE/KEM empfehlt das NIST das Verfahren **Crystals-Kyber**, für digitale Signaturen **Crystals-Dilithium** [2].

PQC-Verfahren

Finalisten der dritten Runde des NIST-PQC-Wettbewerbs [1]

- **Schlüsselaustauschverfahren** (KEM)
  - **NTRU** (**SVP**) nutzt je nach Security Level (1, 3, 5) unterschiedliche Schlüsselgrößen:
    Public Key: 930 - 2401 Byte, Private Key: 1234 - 2983 Byte, Chiffretext: 930 - 2401 Byte
  - Die **LWR**-basierenden **Light Saber** (L1), **Saber** (L3), **Fire Saber** (L5) Verfahren nutzen:
    Public Key: 672 - 1312 Byte, Private Key: 832 - 1664 Byte, Chiffretext: 736 - 1472 Byte
  - **KYBER** (**LWE**) nutzt je nach Security Level (1, 3, 5) folgende Größen:
    Public Key: 800 - 1568 Byte, Private Key: 1632 - 3168 Byte, Chiffretext: 768 - 1568 Byte

PQC-Verfahren

Finalisten der dritten Runde des NIST-PQC-Wettbewerbs [1]

- **Schlüsselaustauschverfahren** (KEM)
  - **NTRU** (**SVP**) nutzt je nach Security Level (1, 3, 5) unterschiedliche Schlüsselgrößen:
    Public Key: 930 - 2401 Byte, Private Key: 1234 - 2983 Byte, Chiffretext: 930 - 2401 Byte
  - Die **LWR**-basierenden **Light Saber** (L1), **Saber** (L3), **Fire Saber** (L5) Verfahren nutzen:
    Public Key: 672 - 1312 Byte, Private Key: 832 - 1664 Byte, Chiffretext: 736 - 1472 Byte
  - **KYBER** (**LWE**) nutzt je nach Security Level (1, 3, 5) folgende Größen:
    Public Key: 800 - 1568 Byte, Private Key: 1632 - 3168 Byte, Chiffretext: 768 - 1568 Byte
- **Signaturverfahren**
  - **Dilithium** (**LWE**) nutzt je nach Security Level (2, 3, 5) unterschiedliche Schlüsselgrößen:
    Public Key: 1312 - 2592 Byte, Private Key: 2528 - 4864 Byte, Signatur: 2420 - 4595 Byte

PQC-Verfahren

Finalisten der dritten Runde des NIST-PQC-Wettbewerbs [1]

- **Schlüsselaustauschverfahren** (KEM)
  - **NTRU** (**SVP**) nutzt je nach Security Level (1, 3, 5) unterschiedliche Schlüsselgrößen:
    Public Key: 930 - 2401 Byte, Private Key: 1234 - 2983 Byte, Chiffretext: 930 - 2401 Byte
  - Die **LWR**-basierenden **Light Saber** (L1), **Saber** (L3), **Fire Saber** (L5) Verfahren nutzen:
    Public Key: 672 - 1312 Byte, Private Key: 832 - 1664 Byte, Chiffretext: 736 - 1472 Byte
  - **KYBER** (**LWE**) nutzt je nach Security Level (1, 3, 5) folgende Größen:
    Public Key: 800 - 1568 Byte, Private Key: 1632 - 3168 Byte, Chiffretext: 768 - 1568 Byte
- **Signaturverfahren**
  - **Dilithium** (**LWE**) nutzt je nach Security Level (2, 3, 5) unterschiedliche Schlüsselgrößen:
    Public Key: 1312 - 2592 Byte, Private Key: 2528 - 4864 Byte, Signatur: 2420 - 4595 Byte

Performance: ***KYBER***, ***Dilithium*** *und* ****Saber*** *gehören zu den schnellsten Verfahren!*

Lessons learned

Lessons learned

1. **Symmetrische** Algorithmen (Chiffren, Hashfunktionen, MAC): Ein gemeinsamer Schlüssel k, aktuell sicher $\approx$ 128 Bit
2. Bei Blockchiffren muss auf einen sicheren **Operationsmodus** geachtet werden!
3. **Asymmetrische** Algorithmen (Verschlüsselung, Signatur, Schlüsselaustausch): Schlüsselpaare (SK, PK), aktuell sicher $\approx$ 2048 Bit
4. **Elliptische Kurven** verbessern die Performance der asymmetrischen Algorithmen
5. Symmetrische Algorithmen sind deutlich **schneller** als asymmetrische Algorithmen.
6. TLS 1.3 > TLS 1.2, alles davor bitte **abschalten**!
7. Zukunftssicherheit: Es gibt bereits effiziente PQC-Verfahren, z.B. **KYBER** und **Dilithium**!

Quelle: Stralsunder Altstadt

Vielen Dank für Ihre Aufmerksamkeit!

Literatur I

[1] Gorjan Alagic u. a. “Status Report on the Third Round of the NIST Post-Quantum Cryptography Standardization Process”. In: (2022).

[2] Hanno Böck. *NIST gibt Post-Quanten-Algorithmen bekannt*. 2022. URL: https://www.golem.de/news/kryptographie-nist-gibt-post-quanten-algorithmen-bekannt-2207-166648.html.

[3] Silke Holtmanns Miguel Angel Reina Ortega Pasi Eronen. *Transport Layer Security (TLS) Parameters*. https://www.iana.org/assignments/tls-parameters/tls-parameters.xhtml. 2022.

[4] E. Rescorla. *The Transport Layer Security (TLS) Protocol Version 1.3*. https://tools.ietf.org/html/rfc8446. 2018.

Literatur II

[5] Bundesamt für Sicherheit in der Informationstechnik (BSI). *BSI TR-02102-1 – Kryptographische Verfahren: Empfehlungen und Schlüssellängen*. https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/TechnischeRichtlinien/TR02102/BSI-TR-02102.pdf. 2022.

[6] Bundesamt für Sicherheit in der Informationstechnik (BSI). *Kryptografie quantensicher gestalten*. 2021. URL: https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/Broschueren/Kryptografie-quantensicher-gestalten.pdf?__blob=publicationFile&v=5.

[7] Dr. Dirk Wetter. *OpenSSL-RFC Mapping*. https://testssl.sh/openssl-rfc.mappping.html. 2017.