Bundesamt
für Sicherheit in der
Informationstechnik

Deutschland
Digital•Sicher•BSI•

BSI – Technische Richtlinie

Bezeichnung:	**Kryptographische Verfahren: Empfehlungen und Schlüssellängen**
Kürzel:	BSI TR-02102-1
Version:	2025-01
Stand:	31. Januar 2025

Version	Datum	Änderungen
2019-01	22.02.2019	Aufnahme des CCM-Modus unter die empfohlenen Betriebsarten. Aufnahme des PKCS1.5-Paddings unter die Legacy-Verfahren.
2020-01	24.03.2020	Empfehlung von FrodoKEM und Classic McEliece mit geeigneten Sicherheitsparametern für PQC-Anwendungen zusammen mit einem bisher empfohlenen asymmetrischen Verfahren. Empfehlung von Argon2id für passwortbasierte Schlüsselableitung. Übergangsweise Verlängerung der Konformität von RSA-Schlüsseln mit einer Schlüssellänge ab 2000 Bits auf Ende 2023.
2021-01	08.03.2021	Überarbeitung des Kapitels zu Zufallsgeneratoren, insbesondere im Hinblick auf die Verwendung von DRG.3- und NTG.1-Zufallsgeneratoren. PTG.2-Zufallsgeneratoren werden für allgemeine Einsatzzwecke nicht mehr empfohlen. Aufnahme standardisierter Versionen hashbasierter Signaturverfahren.
2022-01	28.01.2022	Grundlegende editorische Überarbeitung des gesamten Textes, geringfügige Anpassungen des Layouts. Aktualisierungen in den Bereichen Seitenkanalanalyse, QKD und Seed-Generierung für Zufallszahlengeneratoren.
2023-01	09.01.2023	Anhebung des Sicherheitsniveaus auf 120 Bit, Aktualisierung im Bereich PQ-Kryptographie.
2024-01	02.02.2024	Grundlegende Umstrukturierung im Zusammenhang mit quantensicherer Kryptographie, Abkündigung der Empfehlung von DSA ab 2029, Aufnahme des MLS-Protokolls.
2025-01	31.01.2025	Aktualisierung der PQ-Empfehlungen nach Verabschiedung der NIST Standards, Aufnahme von AES-GCM-SIV und Key-Wrapping, Aktualisierung der Zufallszahlenerzeugung nach AIS 20/31-Update.

Bundesamt für Sicherheit in der Informationstechnik
Postfach 20 03 63, 53133 Bonn, Germany
E-Mail: tr02102@bsi.bund.de
Internet: https://www.bsi.bund.de

Inhaltsverzeichnis

Tabellenverzeichnis

Notationen und Glossar

ggT Der größte gemeinsame Teiler (ggT) (englisch greatest common divisor (gcd)) $\mathrm{ggT}(a, b)$ zweier ganzer Zahlen $a, b \in \mathbb{Z}$ ist diejenige natürliche Zahl mit der Eigenschaft, dass sie sowohl a als auch b teilt und dass jede andere natürliche Zahl, die die Zahlen a und b ebenfalls teilt, bereits Teiler von $\mathrm{ggT}(a, b)$ ist.

$\mathbb{F}_n$ Körper mit n Elementen, auch als Galoiskörper $\mathrm{GF}(n)$ bezeichnet.

$\mathbb{Z}_n$ Ring der Restklassen modulo n in $\mathbb{Z}$, auch als $\mathbb{Z}/n\mathbb{Z}$ bezeichnet.

kgV Das kleinste gemeinsame Vielfache (englisch lowest or least common multiple (lcm)) $\mathrm{kgV}(a, b)$ zweier ganzer Zahlen $a, b \in \mathbb{Z}$ ist die kleinste positive natürliche Zahl, die sowohl Vielfaches von a als auch Vielfaches von b ist.

φ Eulersche Phi-Funktion $\varphi\colon \mathbb{Z} \to \mathbb{Z}$, definiert als $\varphi(n) := \mathrm{Card}(\{a \in \mathbb{N}\colon 1 \leq a \leq n,\ \mathrm{ggT}(a, n) = 1\}) = \mathrm{Card}(\mathbb{Z}_n^*)$.

R^* Einheitengruppe des kommutativen Rings R.

Card Anzahl der Elemente $\mathrm{Card}(M)$ (auch $|M|$) einer endlichen Menge M.

Ceiling-Funktion Ceiling-Funktion $\lceil\cdot\rceil\colon \mathbb{R} \to \mathbb{Z}$, definiert als $\lceil x\rceil := \min\{z \in \mathbb{Z}\colon z \geq x\}$.

Floor-Funktion Floor-Funktion $\lfloor\cdot\rfloor\colon \mathbb{R} \to \mathbb{Z}$, definiert als $\lfloor x\rfloor := \max\{z \in \mathbb{Z}\colon z \leq x\}$.

A

AES Advanced Encryption Standard, von der NIST in FIPS 197 [88] standardisierte Blockchiffre mit einer Blockgröße von 128 Bits. Entsprechend der Länge der verwendeten Schlüssel werden AES-128, AES-192 sowie AES-256 unterschieden. Abgesehen von Related-Key-Angriffen gegen AES-192 und AES-256 sind keine Angriffe gegen AES bekannt, die einen wesentlichen Vorteil gegenüber generischen Angriffen auf Blockchiffren erzeugen.

Asymmetrische Kryptographie Oberbegriff für kryptographische Verfahren, in denen die Ausführung mancher kryptographischer Operationen (etwa die Verschlüsselung einer Nachricht oder die Prüfung einer Signatur) durch Parteien erfolgen kann, die keine geheimen Daten kennen.

Authenticated Encryption with Associated Data (AEAD)) Kategorie von Betriebsmodi von Blockchiffren, die neben Vertraulichkeit auch Authentizität und Integrität sicherstellen.

Authentication Tag Kryptographische Prüfsumme über Daten, die dem Zweck dient, zufällige Fehler oder absichtliche Veränderungen der Daten aufzuzeigen.

Authentisierte Verschlüsselungsverfahren Verschlüsselungsverfahren, die nicht nur die Vertraulichkeit, sondern auch die Integrität der zu verschlüsselnden Daten gewährleisten.

Authentisierung Verfahren mit dem Ziel der sicheren Identifikation einer Person oder eines informationsverarbeitenden Systems. Im Kontext der vorliegenden Technischen Richtlinie geht es dabei um Personen oder Systeme, die Quelle oder Ziel einer Kommunikationsverbindung darstellen und die Authentisierung erfolgt unter Ausnutzung eines kryptographischen Geheimnisses.

Authentizität Eigenschaften der Echtheit, Überprüfbarkeit und Vertrauenswürdigkeit einer Person, eines Systems oder von Daten.

B

Backward Secrecy (eines kryptographischen Protokolls) Auch Future Secrecy oder Post-Compromise Security („Selbstheilung“) genannt, Sicherheitseigenschaft eines kryptographischen Protokolls, welche garantiert, dass verschlüsselte Nachrichten auch dann geheim bleiben, wenn in der Vergangenheit ein Schlüssel kompromittiert wurde.

Betriebsmodus einer Blockchiffre Ein Betriebsmodus oder eine Betriebsart ist ein Verfahren, das beschreibt, wie mit einer Blockchiffre Nachrichten verschlüsselt werden. Erst die Kombination von Blockchiffre und Betriebsmodus erlaubt es, Nachrichten zu verschlüsseln, die länger als die Blocklänge sind. Üblicherweise wird dazu die Nachricht in mehrere Blöcke aufgeteilt und durch sogenanntes Padding auf eine passende Länge gebracht. Ein Initialisierungsvektor kann das Verfahren zusätzlich unabhängig vom verwendeten Schlüssel randomisieren.

Blockchiffre Schlüsselabhängige, effizient berechenbare, umkehrbare Abbildung, die Klartexte einer festen gegebenen Bitlänge n auf Chiffrate der gleichen Länge abbildet. Ohne Kenntnis des Schlüssels sollte es nicht praktisch möglich sein, die Ausgabe der Blockchiffre von der Ausgabe einer zufällig gewählten bijektiven Abbildung zu unterscheiden.

Brute-Force-Angriff Wörtlich Methode der rohen Gewalt, auch Exhaustionsmethode (kurz Exhaustion); Angriffsmethode, die auf einem automatisierten, oftmals systematischen Ausprobieren aller möglichen Fälle beruht, um beispielsweise geheime Schlüssel oder Passwörter zu ermitteln. Bei Verwendung ausreichend langer Schlüssel sind Angriffe dieser Art auf moderne Verschlüsselungsalgorithmen in der Praxis aussichtslos, da der erforderliche Rechenaufwand (und damit der Zeit- und/oder Kostenaufwand) zu groß wäre. Da die Leistung moderner Hardware kontinuierlich steigt und sich der Zeitaufwand für das Durchprobieren aller Schlüssel einer bestimmten Länge dadurch reduziert, muss die minimale Schlüssellänge ausreichend groß gewählt und regelmäßig angehoben werden, um Angriffen durch vollständige Exhaustion vorzubeugen.

C

Challenge-Response-Verfahren Ein Challenge-Response-Verfahren (übersetzt etwa Aufforderung-Antwort-Verfahren) ist ein Verfahren zur Authentifizierung eines Gegenübers auf Basis von Wissen. Hierbei stellt ein Prüfender eine Aufgabe (englisch challenge), die der Beweisende lösen muss (englisch response), um nachzuweisen, dass er eine bestimmte Information kennt, ohne diese Information selbst preiszugeben.

Chinesischer Restsatz (Chinese Remainder Theorem, CRT) Satz über Lösbarkeit und Eindeutigkeit der Lösungen von Kongruenzsystemen. Es seien $n_1, \ldots, n_k \in \mathbb{N}$ paarweise teilerfremde, natürliche Zahlen, $n \coloneqq n_1 \cdot \ldots \cdot n_k$ deren Produkt und $a_1, \ldots, a_k \in \mathbb{Z}$ beliebig. Dann

besitzt das System von Kongruenzen

$$\begin{aligned} x &= a_1 \bmod n_1, \\ &\vdots \\ x &= a_k \bmod n_k \end{aligned}$$

genau eine Lösung $x \in \{0, \ldots, n-1\}$.

Chosen-Ciphertext-Attacke Kryptographischer Angriff, in dem der Angreifer Zugriff auf Klartexte zu von ihm gewählten Chiffraten erhalten kann. Das Ziel des Angreifers ist es in der Regel, ein gegebenes Chiffrat zu dechiffrieren, das zu keinem dieser Klar-Geheim-Kompromisse gehört. Abhängig davon, ob der Angreifer dieses Chiffrat vor oder nach dem Ende des Angriffes kennt, unterscheidet man zwischen adaptiven und nicht-adaptiven Chosen-Ciphertext-Attacken.

Chosen-Plaintext-Attacke Kryptographischer Angriff, in dem der Angreifer Zugriff auf Chiffrate zu von ihm gewählten Klartexten erhalten kann.

Cipher Block Chaining Mode (CBC-Modus) Betriebsart einer Blockchiffren, bei der ein Klartextblock vor dem Verschlüsseln mit dem im vorhergehenden Schritt erzeugten Geheimtextblock per XOR verknüpft wird. Für eine sichere Verwendung dürfen nur unvorhersagbare Initialisierungsvektoren wie beispielsweise eine Zufallszahl zum Einsatz kommen.

Counter Mode (CTR-Modus) Betriebsart, in der Blockchiffren betrieben werden können, um aus ihnen eine Stromchiffre zu erzeugen. Hierbei wird ein erzeugter Geheimtextblock mittels XOR mit dem Klartext kombiniert. Die Besonderheit des Counter Mode im Vergleich zu anderen Betriebsarten stellt die Tatsache dar, dass der Initialisierungsvektor aus einer für jedes Chiffrat neu zu wählenden Nonce verknüpft mit einem Zähler besteht, der mit jedem weiteren Block hochgezählt wird. Die Verknüpfung kann zum Beispiel durch Konkatenation, Addition oder XOR erfolgen.

Counter with Cipher Block Chaining Mode (Counter with CBC-MAC, CCM-Modus) Betriebsart einer Blockchiffre, die den den Counter Mode zur Verschlüsselung mit dem CBC-MAC-Modus zur Integritätssicherung kombiniert und somit aus einer Blockchiffre ein Authenticated-Encryption-Verfahren macht, das in der Lage ist, sowohl Vertraulichkeit als auch Integrität zu garantieren. Beim CCM ist darauf zu achten, dass ein Initialisierungsvektor nicht zweimal mit dem gleichen Schlüssel verwendet werden darf, da der CCM vom Counter Mode abgeleitet ist und letzterer eine Stromchiffre darstellt.

D

Datenauthentisierung Schutz der Integrität einer Nachricht durch kryptographische Verfahren.

Diffie-Hellman-Problem (DH) Problem der Berechnung von g^{ab} gegeben g, g^a, g^b in einer durch $g \in G$ erzeugten zyklischen Gruppe G. Die Schwierigkeit dieses Problems ist abhängig von der Darstellung der Gruppe. Das DH-Problem ist leicht lösbar für Angreifer, die diskrete Logarithmen in G berechnen können.

Diskreter-Logarithmus-Problem (DL) Problem der Berechnung von d gegeben g^d in einer durch $g \in G$ erzeugten zyklischen Gruppe G. Die Schwierigkeit dieses Problems ist abhängig von der Darstellung der Gruppe.

DLIES Discrete Logarithm Integrated Encryption Scheme, hybrides authentisiertes Verschlüsselungsverfahren auf DH-Basis in $\mathbb{F}_p^*$.

E

ECIES Elliptic Curve Integrated Encryption Scheme, hybrides authentisiertes Verschlüsselungsverfahren auf DH-Basis in elliptischen Kurven.

Einwegfunktion Mathematische Funktion, die „leicht“ berechenbar, aber „schwer“ umzukehren ist. An dieser Stelle sind die Begriffe leicht und schwer im komplexitätstheoretischen Sinne zu verstehen, insbesondere im Kontext der Lösung von Problemen in polynomieller Zeit. In einem erweiterten Sinn werden auch Funktionen so bezeichnet, zu denen bisher keine in angemessener Zeit praktisch ausführbare Umkehrung bekannt ist. Eine Einwegpermutation ist eine Einwegfunktion, die gleichzeitig eine Permutation ist, das heißt, eine bijektive Einwegfunktion. Trapdoorfunktionen, auch Trapdoor-Einwegfunktionen oder Falltürfunktionen genannt, sowie Falltürpermutationen stellen eine spezielle Art von Einwegfunktionen dar. Sie lassen sich nur dann effizient umkehren, wenn man eine gewisse Zusatzinformation besitzt. Falltürfunktionen kommen in asymmetrischen Verschlüsselungsverfahren wie zum Beispiel RSA zum Einsatz.

EME-OAEP Encoding Method for Encryption-Optimal Asymmetric Encryption Padding, Padding Verfahren für RSA, siehe auch OAEP.

Ephemeralschlüssel Ein kryptographischer Schlüssel wird als ephemeral (kurzlebig) bezeichnet, wenn er für jede Ausführung eines kryptographischen Protokolls (zum Beispiel Schlüsselaushandlung, Signaturerzeugung) neu generiert wird. Je nach Anwendung können weitere Anforderungen an den jeweiligen Schlüsseltyp gestellt werden, darunter Eindeutigkeit je Nachricht oder Sitzung.

F

Faktorisierungsproblem Aufgabenstellung aus der Zahlentheorie, bei der eine zusammengesetzte Zahl in das Produkt ihrer Primfaktoren zerlegt beziehungsweise allgemeiner ein nichttrivialer Teiler bestimmt werden soll.

Falltürfunktion, Falltürpermutation Siehe Einwegfunktion.

Fault-Attacke Angriff auf ein kryptographisches System, in dem der Angreifer eine fehlerhafte Ausführung einer kryptographischen Operation nutzt beziehungsweise aktiv hervorruft.

Festplattenverschlüsselung Vollständige Verschlüsselung eines Datenträgers mit dem Ziel, dass aus dem verschlüsselten System zumindest in abgeschaltetem Zustand keine vertraulichen Informationen ausgelesen werden können.

Forward Secrecy (eines deterministischen Zufallszahlengenerators) Sicherheitseigenschaft eines deterministischen Zufallsgenerators, die besagt, dass künftige Ausgabewerte des Zufallsgenerators nicht mit mehr als vernachlässigbarem Vorteil vorhergesagt werden können durch Angreifer, die nur frühere Ausgabewerte des Zufallsgenerators, aber nicht dessen inneren Zustand kennen und deren Rechenleistung sich unterhalb einer Schranke bewegt, die durch das Sicherheitsniveau des deterministischen Zufallsgenerators gegeben ist [37].

Forward Secrecy (eines kryptographischen Protokolls) Sicherheitseigenschaft eines kryptographischen Protokolls, die besagt, dass eine Preisgabe kryptographischer Langzeitgeheimnisse es einem Angreifer nicht möglich macht, vergangene Sitzungen des Protokolls zu kompromittieren [47]. Es ist zu beachten, dass für ein beliebiges Protokoll Forward Secrecy nur dann erfüllt sein kann, wenn bei der Erzeugung der Ephemeralschlüssel innerhalb des Protokolls ein Zufallsgenerator eingesetzt wurde, der mindestens Enhanced Backward Secrecy nach [37] garantiert. Sollen darüber hinaus nicht durch einen Angreifer manipulierte *künftige* Sitzungen im Fall einer Kompromittierung aller langfristigen Geheimnisse geschützt bleiben, muss bei der Erzeugung der Ephemeralschlüssel ein Zufallsgenerator eingesetzt werden, der zusätzlich Enhanced Forward Secrecy [37] bietet.

G

GCM Galois Counter Mode, ein Betriebsmodus für Blockchiffren, der aus der Blockchiffre ein authentisiertes Verschlüsselungsverfahren konstruiert und die Authentisierung nichtverschlüsselter Daten unterstützt.

Geburtstagsparadoxon Unter Geburtstagsparadoxon (auch Geburtstagsproblem) wird das Phänomen verstanden, dass bestimmte Wahrscheinlichkeiten intuitiv häufig falsch geschätzt werden. So liegt die Wahrscheinlichkeit, dass bei 23 Personen mindestens zwei von ihnen am gleichen Tag im Jahr Geburtstag haben, bei über 50%, was von den meisten Menschen um eine Zehnerpotenz falsch eingeschätzt wird. Dieser Effekt spielt im Kontext der Kryptographie unter anderem bei kryptographischen Hashfunktionen eine Rolle, die einen eindeutigen Prüfwert aus einer Eingabe berechnen sollen. Es ist dabei viel einfacher, zwei zufällige Eingaben zu finden, die denselben Prüfwert haben, als zu einer vorgegebenen Eingabe eine weitere zu finden, die denselben Prüfwert aufweist (siehe auch Kollisionsangriff).

Gleichverteilung Im Kontext dieser Technischen Richtlinie bedeutet *gleichverteilte* Erzeugung einer Zufallszahl aus einer Grundmenge M, dass der erzeugende Prozess praktisch nicht von einer ideal zufälligen (also von einer echt zufälligen, gleichverteilten, unabhängigen) Ziehung von Elementen aus M unterschieden werden kann.

GMAC Message Authentication Code, der sich aus einer Verwendung des GCM ohne zu verschlüsselnde Daten ergibt.

H

Hashfunktion Funktion $h\colon M \to N$, die effizient berechenbar ist und für die M deutlich größer ist als N. Der Ausgabewert einer Hashfunktion wird als Hashwert oder kurz als Hash bezeichnet. Ist h sowohl kollisionsresistent als auch resistent gegen Berechnung erster und zweiter Urbilder, so heißt h *kryptographische* Hashfunktion. Sofern nicht anders angegeben bezeichnet der Begriff *Hashfunktion* in der vorliegenden Technischen Richtlinie eine kryptographische Hashfunktion.

Hybride Verschlüsselung Kombination aus symmetrischer und asymmetrischer Verschlüsselung, die die Vorteile (Effizienz bzw. sicherer Schlüsselaustausch) beider Arten vereint. Dabei wählt der Sender einen zufälligen symmetrischen Schlüssel, den sogenannten Session-Key bzw. Sitzungsschlüssel, und verschlüsselt damit die zu schützenden Daten symmetrisch. Anschließend verschlüsselt er den Session-Key asymmetrisch mit dem öffentlichen Schlüssel des Empfängers und verschickt ihn zusammen mit der verschlüsselten

Nachricht. Der Empfänger entschlüsselt zunächst den Session-Key mit seinem privaten Schlüssel und damit im Anschluss die eigentliche Nachricht.

I

Informationstheoretische Sicherheit Ein kryptographisches Verfahren heißt *informationstheoretisch sicher*, wenn jeder Angreifer bei dem Versuch, das Verfahren zu brechen, an *Mangel an Information* scheitert. In diesem Fall wird das Sicherheitsziel Vertraulichkeit unabhängig von der dem Angreifer zur Verfügung stehenden Rechenleistung erreicht, solange die Annahmen über die dem Angreifer zugänglichen Informationen über das System zutreffend sind. Es existieren informationstheoretisch sichere Verfahren in vielen Bereichen der Kryptographie, zum Beispiel zur Verschlüsselung von Daten (One Time Pad), zur Authentisierung von Daten (Wegman-Carter-MAC), oder im Bereich Secret Sharing (Shamir Secret Sharing, siehe auch Kapitel 7). In der Regel gibt es in Verfahren dieser Art *keinerlei* Sicherheitsgarantien, wenn die Einsatzvoraussetzungen des Verfahrens nicht exakt erfüllt sind.

Initialisierungsvektor (IV) Ein Initialisierungsvektor ist eine Eingabe in ein kryptographisches Primitiv, die zur Herstellung eines initialen Zustandes verwendet wird. Üblicherweise müssen Initialisierungsvektoren (pseudo-) zufällig sein, in manchen Anwendungsfällen genügt es hingegen auch, wenn sie unvorhersagbar sind oder sich nicht wiederholen.

Instanzauthentisierung Nachweis des Besitzes eines Geheimnisses durch einen Benutzer oder ein informationsverarbeitendes System gegenüber einer anderen Stelle.

Integrität Sicherheitsziel der Bindung des verändernden Zugriffs auf eine Information an das Recht zur Veränderung der Information. Im kryptographischen Kontext bedeutet dies, dass eine Nachricht nur unter Verwendung eines bestimmten geheimen kryptographischen Schlüssels unbemerkt verändert werden kann.

K

Key-Wrapping-Verfahren Klasse von symmetrischen Verschlüsselungsalgorithmen zum Schutz (Vertraulichkeit und Integrität) von kryptografischem Schlüsselmaterial bei Transport und Speicherung.

Kollisionsangriff Ein Kollisionsangriff ist ein Angriff auf eine kryptographische Hashfunktion mit dem Ziel, zwei verschiedene Eingabewerte zu finden, die auf einen identischen Hashwert abgebildet werden. Im Gegensatz zu Preimage-Angriffen sind dabei beide Eingabewerte (und damit auch der Hashwert) frei wählbar.

Kollisionsresistenz Eine Funktion $h\colon M \to N$ heißt kollisionsresistent, wenn es praktisch unmöglich ist, $x, y \in M, x \neq y$ zu finden mit $h(x) = h(y)$.

Kryptoagilität Ein Kryptosystem gilt als kryptoagil, wenn seine Komponenten (zum Beispiel kryptographische Algorithmen, Schlüssellängen, Schlüsselgenerierungsverfahren, technische Umsetzung...) durch andere Komponenten ersetzt werden kann, ohne dass wesentliche Änderungen am Rest des Gesamtsystems vorgenommen werden müssen.

L

Learning with Errors (LWE) Problem Mathematisches Problem, das darin besteht, gestörte (fehlerbehaftete) lineare Gleichungen zu lösen.

M

MAC Message Authentication Code, schlüsselabhängige kryptographische Prüfsumme. Ohne Kenntnis des Schlüssels sollte es einem Angreifer praktisch nicht möglich sein, die MACs von sich nicht wiederholenden Nachrichten von Zufallsdaten zu unterscheiden. Erfolgreiche Fälschungen von Tags sind in diesem Fall für keinen Angreifer mit einer Wahrscheinlichkeit deutlich über 2^{-t} möglich, wobei t die Länge der Authentisierungs-Tags bezeichnet. Vorgaben zur Länge von t sind stark anwendungsabhängig.

Man-in-the-Middle-Angriff Angriffsart, bei der sich ein Angreifer unbemerkt entweder physisch oder – heutzutage meist – logisch zwischen zwei oder mehrere Kommunikationspartnern einklinkt, um beispielsweise Informationen mitzulesen oder zu manipulieren. Der Angreifer begibt sich also „in die Mitte“ der Kommunikation, indem er sich gegenüber dem Sender als Empfänger und gegenüber dem Empfänger als Sender ausgibt.

Min-Entropie Die Min-Entropie einer diskreten Zufallsvariablen X ist definiert als $-\log_2(p)$, wo p die Wahrscheinlichkeit des wahrscheinlichsten Wertes für X bezeichnet.

Module Learning with Errors (MLWE) Problem Verallgemeinerung des LWE-Problems auf Moduln über einem Ring.

Module Short Integer Solution (MSIS) Problem Verallgemeinerung des SIS-Problems auf Moduln über einem Ring.

N

Nonce Eine (kryptographische) Nonce (Number Used Only Once) ist eine beliebige Zeichenfolge, die nur einmal in einer kryptographischen Kommunikation verwendet werden darf. Es handelt sich häufig um eine Zufalls- oder Pseudozufallszahl, die in ein Authentifizierungsprotokoll integriert wird, um zu verhindern, dass vorherige Kommunikation für Replay-Angriffe ausgenutzt werden kann. Ferner werden Noncen häufig als Initialisierungsvektoren und in kryptographischen Hash-Funktionen genutzt.

O

OAEP Optimal Asymmetric Encryption Padding, auf Deutsch etwa Optimales asymmetrisches Verschlüsselungs-Padding; Paddingverfahren, das oft im Zusammenhang mit RSA verwendet wird. Das OAEP stellt eine spezielle Form eines Feistel Netzwerks dar, mit dem im Random-Oracle-Modell ein Verschlüsselungsverfahren aus beliebigen Falltürpermutationen konstruiert werden kann, welches semantisch sicher gegen Chosen-Plaintext-Attacken ist. Wenn OAEP zusammen mit der Falltürpermutation RSA verwendet wird, ist das resultierende Verfahren ferner sicher gegen Chosen-Ciphertext-Attacken. Im allgemeinen erfüllt ein OAEP die beiden folgenden Ziele: Es randomisiert ein ansonsten deterministisches Verschlüsselungsverfahren und beugt einer partiellen Entschlüsselung von Chiffraten (oder einer anderen Form des Informationsabflusses) vor, indem es sicherstellt, dass ein Gegner keine Teile des Klartextes rekonstruieren kann, ohne in der Lage zu sein, die Falltürpermutation zu invertieren.

P

Padding Bezeichnung für das Auffüllen von Nachrichten mit Fülldaten, bevor sie verschlüsselt werden. Padding dient überwiegend dazu, vorhandene Daten in die Gestalt einer durch einen Algorithmus oder ein Protokoll vorgegebenen Struktur zu bringen, das

Ergebnis (zum Beispiel den Chiffretext oder die digitale Signatur) einer Verschlüsselung/Signatur zu randomisieren oder Anfang und Ende des Inhalts eines versandten Chiffrats zu verschleiern.

Partitionsverschlüsselung Partitionsverschlüsselung bezeichnet die vollständige Verschlüsselung einer Partition eines Datenträgers. Die eingesetzten Verfahren ähneln denen zur Festplattenverschlüsselung.

Pepper Geheime, von einem Server gewählte Zeichenfolge, die vor Berechnung eines Hashwertes an ein Passwort angehängt wird, um Wörterbuch- und Brute-Force-Angriffe weiter zu erschweren, wird von NIST auch als *secret salt* bezeichnet. Der Pepper wird nicht in derselben Datenbank gespeichert wie der Hashwert, sondern an einem anderen, möglichst sicheren Ort hinterlegt.

Personal Identification Number (PIN) Unter einer PIN wird im Kontext dieser Technischen Richtlinie ein nur aus den Ziffern 0-9 bestehendes Passwort verstanden.

Post-Quanten-Kryptographie (Post-Quantum Cryptography, PQC) Kryptographische Verfahren, die auf klassischer Hardware ausgeführt werden und von denen angenommen wird, dass sie auch mit Hilfe eines Quantencomputers nicht zu brechen sind.

Pseudozufällige Funktion (Pseudo Random Function, PRF) Familie von deterministischen, effizient berechenbaren Funktionen, die von einem Zufallsorakel praktisch ununterscheidbar sind.

Public-Key-Infrastruktur System, das digitale Zertifikate ausstellen, verteilen, speichern, prüfen und widerrufen kann und insbesondere zur Verwaltung von öffentlichen Schlüsseln (Public Keys) im Rahmen asymmetrischer kryptographischer Verfahren zum Einsatz kommt.

Public-Key-Kryptographie Siehe Asymmetrische Kryptographie.

Q

Quantenschlüsselaustausch (Quantum Key Distribution, QKD) Protokolle, die quantenmechanische Effekte zur sicheren Schlüsseleinigung ausnutzen.

R

Random Oracle, Random-Oracle-Modell Siehe Zufallsorakel.

Regenbogentabelle (Rainbow Table) Datenstruktur, die eine schnelle, speichereffiziente Suche nach der ursprünglichen Eingabe (in der Regel ein Passwort) für einen gegebenen Hashwert ermöglicht. Die Suche über eine Table ist erheblich schneller als bei der Brute-Force-Methode, allerdings ist der Speicherbedarf deutlich höher (Time-Memory Tradeoff).

Related-Key-Attacke Angriff auf ein kryptographisches Verfahren, bei dem ein Angreifer Verschlüsselungen und gegebenenfalls Entschlüsselungen nicht nur unter dem eigentlich verwendeten Schlüssel K, sondern auch unter einer Anzahl anderer, dem Angreifer nicht bekannter Schlüssel abfragen darf, die mit K in einer dem Angreifer bekannten Beziehung stehen. Dieses Modell ist für den Angreifer sehr günstig, dennoch gibt es Situationen, in denen Related-Key-Attacken praktisch relevant sein können, zum Beispiel im Zusammenhang der Konstruktion einer kryptographischen Hashfunktion aus einer Blockchiffre.

RSA Asymmetrisches kryptographisches Verfahren (benannt nach seinen Erfindern Ronald Rivest, Adi Shamir und Leonard Adleman), das zum Verschlüsseln und zum digitalen Signieren verwendet werden kann und auf der Schwierigkeit des Faktorisierungsproblems basiert.

S

Salt Zufällig gewählte Zeichenfolge, die an einen gegebenen Klartext vor dessen weiterer Verarbeitung (zum Beispiel vor Eingabe in eine Hashfunktion) angehängt wird, um die Entropie der Eingabe zu erhöhen. Salts werden häufig für die Speicherung und Übermittlung von Passwörtern benutzt, um die Nutzung von Regenbogentabellen zu erschweren.

Schlüsselableitungsfunktion (Key Derivation Function) Kryptographische Funktion, die aus einem oder mehreren geheimen Eingabewert(en), beispielsweise einem Hauptschlüssel, einem Passwort oder einer Passphrase, einen oder mehrere andere Schlüssel erzeugt. Schlüsselabhängige kryptographische Hash-Funktionen stellen eine häufig verwendete Möglichkeit dar.

Schlüsselkapselungsverfahren (Key Encapsulation Mechanism, KEM) Ein Schlüsselkapselungsverfahren ist eine kryptographische Technik, mit der ein Sitzungsschlüssel, der meist zur Verwendung mit einem symmetrischen Verfahren gedacht ist, mit einem asymmetrischen Verschlüsselungsverfahren übermittelt wird, siehe auch hybride Verschlüsselung.

Schlüssellänge Für symmetrische kryptographische Verfahren ist die Schlüssellänge einfach die Bitlänge des verwendeten geheimen Schlüssels. Für RSA (Signatur- und Verschlüsselungsverfahren) wird die Bitlänge des RSA-Moduls n als Schlüssellänge bezeichnet. Für Verfahren, die auf dem Diffie-Hellman-Problem oder diskreten Logarithmen in $\mathbb{F}_p^*$ basieren (DLIES, DH-Schlüsseltausch, DSA), wird als Schlüssellänge die Bitlänge von p definiert. Für Verfahren, die auf dem Diffie-Hellman-Problem oder diskreten Logarithmen in einer elliptischen Kurve C über dem endlichen Körper $\mathbb{F}_n$ aufbauen (ECIES, ECDH, ECDSA und Varianten), ist die Schlüssellänge die Bitlänge von n.

Schlüsselstreckung (Key Stretching) Kryptographische Schlüsselableitungsoperation, die einen schwachen Schlüssel, üblicherweise ein Passwort, sicherer machen soll, indem sie dafür sorgt, dass zum Durchprobieren aller Möglichkeiten mehr Mittel (Zeit, Speicher) benötigt werden. Dabei darf es keine Möglichkeit geben, den verbesserten Schlüssel mit geringerem Aufwand aus dem Anfangsschlüssel berechnen zu können.

Secret Sharing Verfahren zur Verteilung geheimer Daten (zum Beispiel eines kryptographischen Schlüssels) auf mehrere Personen oder Speichermedien. Das ursprüngliche Geheimnis kann dabei nur unter Auswertung mehrerer Teilgeheimnisse rekonstruiert werden. Zum Beispiel kann ein Secret-Sharing-Schema vorsehen, dass von insgesamt n Teilgeheimnissen mindestens k bekannt sein müssen, um den zu schützenden kryptographischen Schlüssel zu rekonstruieren.

Seed Startwert, mit dem ein Zufallszahlengenerator initialisiert wird, um infolgedessen eine Folge von Zufallszahlen bzw. Pseudozufallszahlen zu generieren. Verwendet man in deterministischen Zufallszahlengeneratoren den gleichen Seed, so erhält man die gleiche Folge von Pseudozufallszahlen.

Seitenkanalangriff Angriff auf ein kryptographisches System, der die Ergebnisse von physikalischen Messungen am System (zum Beispiel Energieverbrauch, elektromagnetische Abstrahlung, Laufzeit einer Operation) ausnutzt, um Rückschlüsse auf sensible Daten zu ziehen. Seitenkanalangriffe sind für die praktische Sicherheit informationsverarbeitender Systeme von sehr hoher Relevanz.

Shannon-Entropie Die Shannon-Entropie einer diskreten Zufallsvariablen X ist definiert als $-\sum_{x \in W} p_x \log_2(p_x)$, wobei W der Wertebereich von X ist und p_x die Wahrscheinlichkeit, mit der X den Wert $x \in W$ annimmt, das heißt $p_x = \mathbb{P}(X = x)$.

Short Integer Solution (SIS) Problem) Mathematisches Problem, das darin besteht, einen kurzen, ganzzahligen Lösungsvektor zu einem linearen Gleichungssystem zu finden.

Sicherheitsniveau (kryptographischer Verfahren) Ein kryptographisches Verfahren erreicht ein Sicherheitsniveau von n Bit, wenn mit jedem Angriff gegen das Verfahren, der das Sicherheitsziel des Verfahrens mit hoher Erfolgswahrscheinlichkeit bricht, Kosten verbunden sind, die zu 2^n Berechnungen der Verschlüsselungsfunktion einer effizienten Blockchiffre (zum Beispiel AES) äquivalent sind.

Symmetrische Kryptographie Oberbegriff für kryptographische Verfahren, in denen alle beteiligten Parteien über vorverteilte gemeinsame Geheimnisse verfügen müssen, um das Verfahren insgesamt ausführen zu können.

T

Trapdoor-Einwegfunktion, Trapdoorfunktion, Trapdoorpermutation Siehe Einwegfunktion.

U

Urbild-Angriff Urbild-Angriffe (englisch preimage attack) sind Angriffe auf eine kryptographische Hashfunktion mit dem Ziel, zu einem gegebenen Hashwert eines unbekannten Eingabewerts ein Urbild zu finden (Erstes-Urbild-Angriff, englisch first-preimage attack) oder zu einem gegebenem Eingabewert ein weiteres Urbild zu finden, das den gleichen Hashwert liefert (Zweites-Urbild-Angriff, englisch second-preimage attack).

Urbildresistenz Eine Funktion $h\colon M \to N$ heißt Urbild-resistent, wenn es praktisch unmöglich ist, zu einem gegebenen $y \in N$ ein $x \in M$ zu finden mit $h(x) = y$. Sie heißt resistent gegen Berechnung *zweiter* Urbilder, falls es zu gegebenem x, y mit $h(x) = y$ praktisch unmöglich ist, ein $x' \neq x$ zu berechnen mit $h(x') = y$.

V

Vertraulichkeit Sicherheitsziel der Bindung des lesenden Zugriffs auf eine Information an das Recht auf Zugriff. Im kryptographischen Kontext bedeutet dies, dass der Zugriff auf den Inhalt einer Nachricht nur Besitzern eines geheimen kryptographischen Schlüssels möglich sein sollte.

Volume-Verschlüsselung Siehe Partitionsverschlüsselung.

W

Wörterbuch-Angriff (Dictionary Attack) Angriffsmethode, um ein unbekanntes Passwort (oder einen Benutzernamen) durch systematisches Ausprobieren einer Passwörterliste (auch

als wordlist oder dictionary bezeichnet) zu ermitteln. Der Erfolg derartiger Angriffe beruht auf der Tatsache, dass von Nutzern vergebene Passwörter in der Praxis häufig leicht zu erraten sind, beispielsweise wenn sie aus regulären oder nur leicht abgewandelten Wörterbucheinträgen bestehen oder in ähnlicher Form an verschiedenen Stellen benutzt werden, so dass Passwortlisten aus vormaligen Sicherheitsvorfällen zu einem erfolgreichen Angriff führen.

Z

Zufallsorakel Ein Zufallsorakel (englisch random oracle) ist ein theoretisches Konstrukt, das zu jeder Eingabe einen (gleichverteilt) zufälligen Wert der Ausgabemenge zurückgibt; bei einer wiederholten Anfrage der gleichen Eingabe antwortet es jedes Mal mit der gleichen Ausgabe. Zufallsorakel kommen üblicherweise zum Einsatz, wenn kryptographische Beweise nicht mit schwächeren Anforderungen an eine kryptographische Hashfunktion geführt werden können, da sie aufgrund ihrer Konstruktion die klassischen Anforderungen an eine kryptographische Hashfunktion (starke Kollisionsresistenz und Resistenz gegenüber der Berechnung von Urbildern erster und zweiter Art) perfekt erfüllen. Von Systemen, die beweisbar sicher sind, wenn jede Hashfunktion durch ein Zufallsorakel ersetzt wird, oder von Sicherheitsbeweisen, die ein Zufallsorakel verwenden, sagt man, sie seien sicher im Random-Oracle-Modell bzw. im Random-Oracle-Modell geführt worden, im Unterschied zum Standard-Modell der Kryptographie (das Standard-Modell ist das Berechnungsmodell, in dem ein Angreifer nur durch die ihm zur Verfügung stehende Zeit und Rechenkraft beschränkt wird).

1. Einleitung

In dieser Technischen Richtlinie veröffentlicht das Bundesamt für Sicherheit in der Informationstechnik (BSI) eine Bewertung der Sicherheit für ausgewählte kryptographische Verfahren, verbunden mit einer langfristigen Orientierung für ihren Einsatz. Die ausgesprochenen Empfehlungen werden jährlich überprüft und bei Bedarf angepasst. Es wird dabei jedoch ausdrücklich kein Anspruch auf Vollständigkeit erhoben, das heißt nicht aufgeführte Verfahren werden vom BSI nicht zwangsläufig als unsicher beurteilt. Umgekehrt ist allerdings auch der Schluss falsch, dass kryptographische Systeme, die als Grundkomponenten nur in der vorliegenden Technischen Richtlinie empfohlene Verfahren verwenden, automatisch sicher sind: Die Anforderungen der konkreten Anwendung und die Verkettung verschiedener kryptographischer und nicht-kryptographischer Mechanismen können im Einzelfall dazu führen, dass die hier ausgesprochenen Empfehlungen nicht direkt umsetzbar sind oder dass Sicherheitslücken entstehen. Aufgrund dieser Erwägungen ist insbesondere zu betonen, dass die in dieser Technischen Richtlinie ausgesprochenen Empfehlungen keine Entscheidungen, etwa im Rahmen staatlicher Evaluierungs- und Zulassungsprozesse, vorwegnehmen.

Die vorliegende Technische Richtlinie richtet sich vielmehr in erster Linie empfehlend an Entwickler, die ab 2025 die Einführung neuer kryptographischer Systeme planen. Aus diesem Grund wird in diesem Dokument auch bewusst auf die Angabe kryptographischer Verfahren verzichtet, die zwar zum derzeitigen Zeitpunkt noch als sicher gelten, mittelfristig aber nicht mehr empfohlen werden können, da sie wenn auch noch keine ausnutzbaren, so doch zumindest theoretische Schwächen zeigen. Bei der Entwicklung neuer kryptographischer Systeme können verschiedene andere, vom BSI herausgegebene Dokumente ebenfalls eine Rolle spielen, darunter [33, 34, 38, 41, 40, 39, 27, 30, 35]. Für bestimmte Anwendungen sind die in diesen Dokumenten enthaltenen Vorgaben – im Gegensatz zu den Empfehlungen der vorliegenden Richtlinie – sogar bindend. Eine Diskussion verschiedener Vorgaben und Empfehlungen findet sich in [61]. Die folgenden beiden Abschnitte beschreiben zunächst die Sicherheitsziele sowie die Auswahlkriterien der empfohlenen kryptographischen Verfahren. Weiter werden sehr allgemeine Hinweise zur konkreten Umsetzung der empfohlenen Verfahren gegeben.

Anschließend werden in den Kapiteln 3 bis 8 die empfohlenen kryptographischen Verfahren für folgende Anwendungen aufgelistet:

2 Asymmetrische Verschlüsselung und Schlüsseleinigung,

3 Symmetrische Verschlüsselung und Schlüsseleinigung,

4 Kryptographische Hashfunktionen,

5 Datenauthentisierung,

6 Instanzauthentisierung,

7 Secret Sharing und

8 Zufallszahlengeneratoren.

In den jeweiligen Abschnitten werden zudem geforderte (Mindest-) Schlüssellängen und andere zu beachtende Nebenbedingungen genannt.

Damit ein eingesetztes Verfahren die an es gestellten Sicherheitsanforderungen erfüllt, müssen häufig verschiedene kryptographische Algorithmen miteinander kombiniert werden. So ist es beispielsweise oft notwendig, vertrauliche Daten nicht nur zu verschlüsseln, sondern ein Empfänger muss sich auch sicher sein können, von wem die Daten versendet beziehungsweise ob diese während der Übertragung manipuliert wurden. Die zu übertragenden Daten müssen also zusätzlich mit einem geeigneten Verfahren authentisiert werden. Ein weiteres Beispiel stellen Schlüsselvereinbarungsverfahren dar. Hier ist es wichtig zu wissen, mit wem die Schlüsselvereinbarung durchgeführt wird, um sogenannte Man-in-the-Middle-Attacken und Unknown-Key-Share-Attacken [15] ausschließen zu können. Dies wird durch Verfahren ermöglicht, die Schlüsselvereinbarung und Instanzauthentisierung kombinieren. Für diese beiden Einsatzszenarien werden in Anhang A entsprechende Verfahren angegeben, die durch Kombination der in den Kapiteln 3 bis 8 aufgelisteten Verfahren konstruiert werden und das in dieser Technischen Richtlinie geforderte Sicherheitsniveau erfüllen. Zusätzlich werden in Anhang B häufig verwendete Funktionen und Algorithmen empfohlen, die beispielsweise zur Schlüsselableitung für symmetrische Verfahren oder zur Erzeugung von Primzahlen und anderen Systemparametern für asymmetrische Verfahren benötigt werden. Schließlich werden in Anhang C Empfehlungen zur Verwendung ausgewählter kryptographischer Protokolle ausgesprochen. In der aktuellen Version dieser Technischen Richtlinie betrifft dies nur die Protokolle SRTP und MLS; Empfehlungen zu TLS, IPsec und SSH wurden in die Technischen Richtlinien TR-02102-2 [23], TR-02102-3 [24] beziehungsweise TR-02102-4 [25] ausgelagert.

1.1. Sicherheitsziele und Auswahlkriterien

Die Sicherheit kryptographischer Verfahren hängt wesentlich von der Stärke der zugrunde liegenden kryptographischen Primitive ab. Aus diesem Grund werden in dieser Technischen Richtlinie nur Verfahren empfohlen, die basierend auf den heute vorliegenden Ergebnissen langjähriger Analysen und Diskussionen entsprechend eingeschätzt und als sicher bewertet werden können. Weitere Faktoren von zentraler Bedeutung für die Sicherheit stellen die konkreten Implementierungen der Algorithmen sowie die Zuverlässigkeit eventueller Hintergrundsysteme, wie zum Beispiel benötigte Public-Key-Infrastrukturen für den sicheren Austausch von Zertifikaten, dar. Die Umsetzung konkreter Implementierungen wird an dieser Stelle aber ebenso wenig betrachtet wie eventuell auftretende patentrechtliche Probleme. Auch wenn bei der Auswahl der Verfahren darauf geachtet wurde, dass die Algorithmen frei von Patenten sind, kann dies durch das BSI jedoch nicht garantiert werden. Zudem finden sich in dieser Technischen Richtlinie einzelne Hinweise auf mögliche Schwierigkeiten und Probleme bei der Implementierung kryptographischer Verfahren, diese sind allerdings nicht als erschöpfende Liste derartiger Probleme zu verstehen.

Insgesamt erreichen alle in dieser Technischen Richtlinie angegebenen kryptographischen Verfahren mit den in den einzelnen Abschnitten genannten Parametern ein Sicherheitsniveau von mindestens 120 Bits. Die in dieser Technischen Richtlinie zur Verwendung in neuen kryptographischen Systemen empfohlenen Bitlängen richten sich nach diesem Minimalniveau aber nur insoweit, als dass dieses für kein empfohlenes Verfahren unterschritten wird. Die effektive Stärke der empfohlenen Verfahren ist in vielen Fällen höher als 120 Bits. Damit wird ein gewisser Sicherheitsspielraum gegenüber möglichen künftigen Fortschritten in der Kryptoanalyse geschaffen. Wie bereits in der Einleitung erwähnt gilt im Umkehrschluss nicht, dass in dieser Technischen Richtlinie nicht angegebene Verfahren das geforderte Sicherheitsniveau nicht erreichen.

Tabelle 1.1 zeigt die Schlüssellängen ausgewählter Algorithmen und Typen von Algorithmen, für die ein Sicherheitsniveau von 120 Bits nach gegenwärtigem Kenntnisstand gerade erreicht wird.

Symmetrische Verfahren		Asymmetrische Verfahren		
Ideale Blockchiffre	Idealer MAC	RSA	DSA[1]/DLIES	ECDSA/ECIES
120	120	2800	2800	240

Tabelle 1.1: Beispiele für Schlüssellängen für ein Sicherheitsniveau von mindestens 120 Bits.

[1] Die Verwendung des Signaturverfahrens DSA (siehe Abschnitt 5.3.2) wird aufgrund der geringen Verbreitung und der Abkündigung in [103] in der vorliegenden Technischen Richtlinie nur noch bis 2029 empfohlen.

Tabelle 1.2 fasst die *empfohlenen* Schlüssellängen verschiedener Typen kryptographischer Primitive zusammen.

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

Tabelle 1.2: Empfohlene Schlüssellängen für verschiedene kryptographische Verfahren.

Schlüsselaustauschverfahren auf Diffie-Hellman-Basis sind in den Tabellen 1.1 und 1.2 entsprechend zu DSA/ECDSA einzugruppieren.

In vielen Anwendungsfällen spielen neben der Schlüssellänge weitere Sicherheitsparameter eine Rolle für die Gesamtsicherheit eines kryptographischen Systems. So stellt im Falle von Message Authentication Codes (MACs) die Länge des Digest-Outputs neben der Schlüssellänge einen wichtigen Sicherheitsparameter dar. Idealerweise sollte ein MAC von einem Angreifer praktisch nicht von einer Zufallsfunktion mit entsprechender Digest-Länge unterschieden werden können. Solange dieses Kriterium erfüllt ist, bleibt dem Angreifer nur die Möglichkeit, gefälschte Nachrichten durch Raten zu erzeugen, wobei er pro Versuch eine Erfolgswahrscheinlichkeit von 2^{-n} hat, wenn n die Tag-Länge ist. In vielen Anwendungen kann in einer solchen Situation $n = 96$ als akzeptabel angesehen werden, das heißt eine Tag-Länge von $n = 96$ Bits.

Bei Blockchiffren ist die Blockbreite ein von der Schlüssellänge unabhängiger Sicherheitsparameter. In Abwesenheit struktureller Angriffe auf eine Blockchiffre ist die wesentlichste Auswirkung einer geringen Blockbreite, dass ein häufigerer Schlüsselwechsel notwendig wird. Die genauen Auswirkungen hängen vom verwendeten Betriebsmodus ab. In der vorliegenden Technischen Richtlinie werden keine Blockchiffren empfohlen, die eine Blockbreite von unter 128 Bits aufweisen.

Ein wichtiger Typ kryptographischer Primitive, die überhaupt keine geheimen Daten verarbeiten, sind kryptographische Hashfunktionen. Hier stellt die Länge des zurückgegebenen Digest-Wertes den wichtigsten Sicherheitsparameter dar und sollte für allgemeine Anwendungen mindestens 200 Bits betragen, damit das in dieser Richtlinie minimal geforderte Sicherheitsniveau erreicht wird. Die in Kapitel 4 empfohlenen Hashfunktionen haben eine Mindesthashlänge von 256 Bits, auf Abweichungen von dieser Regel für besondere Anwendungen wird in der vorliegenden Technischen Richtlinie an gegebener Stelle eingegangen.

1.2. Allgemeine Hinweise

Zuverlässigkeit von Prognosen zur Sicherheit kryptographischer Verfahren Bei der Festlegung der Größe von Systemparametern (wie zum Beispiel Schlüssellänge, Größe des Bildraums für Hash-

funktionen u.ä.) müssen nicht nur die besten heute bekannten Algorithmen zum Brechen der entsprechenden Verfahren und die Leistung heutiger Rechner berücksichtigt werden, sondern vor allem eine Prognose der zukünftigen Entwicklung beider Aspekte zugrunde gelegt werden, siehe insbesondere auch [76, 75, 36].

Die Entwicklung der Leistungsfähigkeit klassischer Rechner kann heutzutage relativ gut eingeschätzt werden. Grundlegende wissenschaftliche Fortschritte (entweder in Bezug auf Angriffsalgorithmen oder etwa die Entwicklung eines kryptographisch relevanten Quantencomputers) sind dagegen nicht vorhersagbar. Daher ist jede Vorhersage über einen Zeitraum von sechs bis sieben Jahren hinaus schwierig, insbesondere bei asymmetrischen Verfahren, und selbst für diesen Zeitraum von sechs bis sieben Jahren können sich die Prognosen aufgrund unvorhersehbarer Entwicklungen als falsch erweisen. Die Angaben dieser Technischen Richtlinie werden daher nur beschränkt auf einen Zeitraum bis Ende 2031 ausgesprochen.

Allgemeine Leitlinien zum Umgang mit vertraulichen Daten mit längerfristigem Schutzbedarf
Da ein Angreifer Daten speichern und später entschlüsseln kann, bleibt ein grundsätzliches Risiko für den langfristigen Schutz der Vertraulichkeit. Daraus ergeben sich als unmittelbare Konsequenzen:

- Die Übertragung und Speicherung vertraulicher Daten sollte auf das notwendige Maß reduziert werden. Dies betrifft nicht nur Klartexte, sondern zum Beispiel in besonderem Maße auch die Vermeidung einer Speicherung von Sitzungsschlüsseln auf jeglicher Art von nichtflüchtigen Medien sowie ihre zügige sichere Löschung, sobald sie nicht mehr benötigt werden.
- Das Kryptosystem muss so ausgelegt sein, dass ein Übergang zu größeren Schlüssellängen und stärkeren kryptographischen Verfahren möglich ist (Kryptoagilität).
- Für Daten, deren Vertraulichkeit langfristig gesichert bleiben soll, wird empfohlen, für die Verschlüsselung der Übertragung über allgemein zugängliche Kanäle wie beispielsweise das Internet von vornherein möglichst starke der in dieser Technischen Richtlinie empfohlenen Verfahren zu wählen. In den meisten Kontexten ist zum Beispiel der AES-256 aufgrund seiner größeren Schlüssellänge als stärker zu betrachten als der AES-128. Da solche allgemeinen Einschätzungen aber schwierig sind – im konkreten Beispiel sind etwa in einigen (konstruierten) Szenarien der AES-192 und der AES-256 *schwächer* als der AES-128 gegenüber den besten bekannten Angriffen (siehe [14]) – sollte nach Möglichkeit bereits früh der Rat eines Experten eingeholt werden.
- Im Hinblick auf die Auswahl kryptographischer Komponenten für eine neue Anwendung ist grundsätzlich zu berücksichtigen, dass das Gesamtsystem im Allgemeinen nicht stärker ist als die schwächste Komponente. Wird daher ein Sicherheitsniveau von beispielsweise 256 Bits für das Gesamtsystem angestrebt, müssen alle Komponenten mindestens diesem Sicherheitsniveau genügen. Die Auswahl einzelner Komponenten, die ein höheres Sicherheitsniveau gegenüber den besten bekannten Angriffen erreichen als das Gesamtsystem, kann unter Umständen trotzdem sinnvoll sein, weil dies die Robustheit des Systems gegenüber Fortschritten in der Kryptoanalyse erhöht.
- Um die Möglichkeit von Seitenkanalattacken und Implementierungsfehlern zu minimieren, sollte im Falle von Software-Implementierungen der hier vorgestellten kryptographischen Verfahren einem Einsatz quelloffener Bibliotheken der Vorzug vor Eigenentwicklungen gegeben werden, wenn davon ausgegangen werden kann, dass die verwendeten Funktionen der Bibliothek einer breiten öffentlichen Analyse unterzogen wurden. Bei der Bewertung eines Kryptosystems ist dabei die Vertrauenswürdigkeit aller Systemfunktionen zu prüfen, insbesondere beinhaltet dies auch Abhängigkeiten der Lösung von Eigenschaften der verwendeten Hardware.

- Die meisten der heute eingesetzten klassischen, asymmetrischen Schlüsseleinigungs- und Signaturverfahren sind durch die fortschreitende Entwicklung hinreichend großer Quantencomputer nicht länger sicher. Insbesondere für Daten mit längerfristigem Schutzbedarf ergibt sich aus der möglichen Speicherung und späteren Entschlüsselung von Daten eine relevante Bedrohung („Store Now, Decrypt Later"-Szenario). Die vorliegende technische Richtlinie empfiehlt daher, quantensichere Schlüsseleinigungsverfahren hybrid mit klassischen Verfahren zum langfristigen Schutz der Vertraulichkeit einzusetzen, für Details siehe Abschnitt 2.4. Insbesondere für Anwendungsfälle mit langen Migrationszeiten wird eine rechtzeitige Migration zu quantensicheren Signaturverfahren empfohlen, siehe Abschnitt 5.3.4.

Fokus des vorliegenden Dokuments Die Sicherheitsbewertung der in dieser Technischen Richtlinie empfohlenen kryptographischen Verfahren erfolgt ohne Berücksichtigung der Einsatzbedingungen. Für konkrete Szenarien können sich andere Sicherheitsanforderungen ergeben, denen die in dieser Technischen Richtlinie empfohlenen Verfahren möglicherweise nicht gerecht werden. Beispiele hierfür stellen etwa die Verschlüsselung von Datenträgern, die verschlüsselte Speicherung und Verarbeitung von Daten auf durch externe Anbieter betriebenen Systemen („Cloud Computing" beziehungsweise „Cloud Storage") oder kryptographische Anwendungen auf Geräten mit extrem geringen rechentechnischen Ressourcen („Lightweight Cryptography") dar. Hinweise zu einigen der genannten Anwendungsszenarien finden sich in Abschnitt 1.6. Dieses Dokument kann daher die Entwicklung kryptographischer Infrastrukturen unterstützen, nicht aber die Bewertung des Gesamtsystems durch einen Kryptologen ersetzen oder die Ergebnisse einer solchen Bewertung vorwegnehmen.

Allgemeine Empfehlungen zur Entwicklung kryptographischer Systeme Die folgende Auflistung fasst stichpunktartig einige Grundsätze zusammen, deren Beachtung bei der Entwicklung kryptographischer Systeme generell empfohlen wird:

- Bei der Planung von Systemen, für die kryptographische Komponenten vorgesehen sind, sollte frühzeitig die Zusammenarbeit mit Experten auf kryptographischem Gebiet gesucht werden.

- Die kryptographischen Verfahren müssen in vertrauenswürdigen technischen Komponenten implementiert werden, um das in dieser Technischen Richtlinie aufgeführten geforderte Sicherheitsniveau zu erreichen.

- Die Implementierungen der kryptographischen Verfahren und Protokolle selbst sind in die Sicherheitsanalyse mit einzubeziehen, um zum Beispiel Seitenkanalangriffe oder Implementierungsschwächen zu verhindern.

- Wenn die Übereinstimmung eines Produktes mit den Anforderungen dieser Technischen Richtlinie nachgewiesen werden soll, muss die Sicherheit technischer Komponenten und Implementierungen dem jeweils vorgesehenen Schutzprofil entsprechend durch Common Criteria Zertifikate oder ähnliche Verfahren des BSI, wie zum Beispiel im Zuge einer Zulassung, nachgewiesen werden.

- Nach Entwicklung und vor Produktiveinsatz eines kryptographischen Systems sollte eine Evaluierung des Systems durch unabhängige Experten durchgeführt werden, die nicht an der Entwicklung beteiligt waren. Eine Einschätzung der Verfahrenssicherheit durch die Entwickler allein sollte nicht als belastbar betrachtet werden, auch wenn die Entwickler des Systems über gute kryptographische Kenntnisse verfügen.

- Die Folgen eines Versagens der eingesetzten Sicherheitsmechanismen müssen gründlich dokumentiert werden. Wo es möglich ist, sollte das System so ausgelegt werden, dass das Versagen oder die Manipulation einzelner Systemkomponenten unmittelbar detektiert wird und die Sicherheitsziele durch Übergang in einen geeigneten sicheren Zustand gewahrt bleiben.

1.3. Kryptographische Hinweise

Ein kryptographisches Verfahren kann häufig für verschiedene Anwendungen eingesetzt werden, so können zum Beispiel Signaturverfahren sowohl zur Datenauthentisierung als auch zur Instanzauthentisierung verwendet werden. Grundsätzlich sollten für unterschiedliche Anwendungen jeweils verschiedene Schlüssel eingesetzt werden. Ein weiteres Beispiel stellen symmetrische Schlüssel zur Verschlüsselung und symmetrischen Datenauthentisierung dar. Hier muss bei konkreten Implementierungen dafür gesorgt werden, dass für beide Verfahren jeweils verschiedene Schlüssel eingesetzt werden, die sich insbesondere nicht voneinander ableiten lassen, siehe auch Abschnitt A.1.

An einigen Stellen beschränkt sich diese Technische Richtlinie auf eine informative Beschreibung der kryptographischen Primitive. Da die kryptographische Sicherheit aber nur im Rahmen der jeweiligen exakten Spezifikation und des jeweils verwendeten Protokolls bewertet werden kann, müssen daher die entsprechenden hier angegebenen Standards beachtet werden. Weitere konkrete Hinweise werden, so nötig, in den entsprechenden Abschnitten angegeben.

1.4. Implementierungsaspekte

Neben der kryptanalytischen Sicherheit der verwendeten Algorithmen ist die Sicherheit der Implementierung, beispielsweise gegen Seitenkanal- und Fault-Attacken, für die Sicherheit eines Kryptosystems von entscheidender Bedeutung. Dies gilt insbesondere für symmetrische Verschlüsselungsverfahren. Eine detaillierte Behandlung dieses Themas liegt außerhalb des Rahmens der vorliegenden Technischen Richtlinie, zumal die zu treffenden Gegenmaßnahmen im Einzelfall auch in hohem Maße von der konkreten Implementierung abhängig sind. An dieser Stelle seien lediglich die folgenden, allgemeinen Maßnahmen empfohlen:

- Wann immer es mit vertretbarem Aufwand möglich ist, sollten kryptographische Operationen in sicherheitszertifizierten Hardwarekomponenten durchgeführt werden (also zum Beispiel auf einer geeigneten Smartcard) und die dabei verwendeten Schlüssel sollten diese Komponenten nicht verlassen.

- Angriffe, die durch entfernte, passive Angreifer durchgeführt werden können, sind naturgemäß schwer zu detektieren und können daher zu einem wesentlichen unbemerkten Datenabfluss über einen langen Zeitraum hinweg führen. Dazu zählen etwa Angriffe unter Ausnutzung variabler Bitraten, Dateilängen oder variabler Antwortzeiten kryptographischer Systeme. Es wird empfohlen, die Auswirkungen solcher Seitenkanäle auf die Systemsicherheit bei der Entwicklung eines neuen kryptographischen Systems gründlich zu analysieren und die Ergebnisse der Analyse im Entwicklungsprozess zu berücksichtigen.

- Sowohl bei Angriffen auf symmetrische als auch auf asymmetrische Verfahren kommen in jüngster Zeit vermehrt Angriffsmethoden zum Einsatz, die auf Verfahren aus dem Bereich des Maschinellen Lernens (ML) beziehungsweise der Künstlichen Intelligenz (KI) basieren. Insbesondere neuronale Netze erzielen dabei häufig State-of-the-Art-Resultate. Es zeichnet sich ab, dass KI-basierte Methoden den derzeit zumeist verwendeten klassischen Angriffsmethoden (zum Beispiel basierend auf Korrelationen oder Templates) in einigen Anwendungsfällen

deutlich überlegen sein könnten. Ein KI-Leitfaden, der detailliertere Empfehlungen zu diesem Thema enthält, befindet sich daher in Vorbereitung.

- Auf Protokollebene sollte der Entstehung von Fehlerorakeln vorgebeugt werden. Am wirkungsvollsten kann dies durch eine MAC-Sicherung aller Chiffrate geschehen. Die Authentizität der Chiffrate sollte dabei vor Ausführung aller anderen kryptographischen Operationen geprüft werden und es sollte keine weitere Verarbeitung nicht-authentischer Chiffrate erfolgen.

Wie auch in anderen Zusammenhängen trifft insbesondere auch hier die bereits mehrfach genannte, allgemeine Empfehlung zu, nach Möglichkeit stets Komponenten zu verwenden, die bereits einer intensiven Analyse durch eine breite Öffentlichkeit unterzogen wurden und frühzeitig entsprechende Experten in die Entwicklung neuer kryptographischer Systeme einzubinden.

1.5. Umgang mit Legacy-Algorithmen

Es gibt Algorithmen, gegen die bislang keine praktischen Angriffe bekannt sind und die in einigen Anwendungen noch eine hohe Verbreitung und damit eine gewisse Bedeutung besitzen, die aber grundsätzlich für neue Systeme als nicht mehr dem Stand der Technik entsprechend angesehen werden. Wir gehen im Folgenden kurz auf die wichtigsten Beispiele ein.

HMAC-MD5 Die mangelnde Kollisionsresistenz von MD5 stellt in der HMAC-Konstruktion mit MD5 als Hashfunktion [10] noch kein unmittelbares Problem dar, da die HMAC-Konstruktion nur eine sehr schwache Form von Kollisionsresistenz von der Hashfunktion benötigt. Allerdings erscheint es grundsätzlich nicht ratsam, in neuen Kryptosystemen Primitive zu verwenden, die in ihrer ursprünglichen Funktion vollständig gebrochen wurden. Systeme, die MD5 für kryptographische Zwecke verwenden, sind daher nicht mit der vorliegenden Technischen Richtlinie konform.

HMAC-SHA1 SHA1 ist keine kollisionsresistente Hashfunktion. Die Erzeugung von SHA1-Kollisionen ist zwar mit moderatem Aufwand verbunden, aber praktisch möglich [78, 77, 117], auch wenn gegen die Verwendung von SHA1 in Konstruktionen, die keine Kollisionsresistenz benötigen (zum Beispiel als Grundlage für einen HMAC, als Teil der Mask Generation Function in RSA-OAEP oder als Komponente eines Pseudozufallsgenerators) nach gegenwärtigem Kenntnisstand sicherheitstechnisch nichts spricht. Als grundsätzliche Sicherungsmaßnahme wird empfohlen, auch in diesen Anwendungen eine Hashfunktion der SHA2- oder der SHA3-Familie einzusetzen.

RSA mit PKCS1v1.5-Padding Grundsätzlich wird eine Verwendung dieses Formats in neuen Systemen weder für Verschlüsselung noch für Signaturerstellung empfohlen, da es mit RSA-OAEP beziehungsweise RSA-PSS Paddingverfahren mit solideren theoretischen Sicherheitseigenschaften gibt. Außerdem haben sich RSA-Implementierungen mit PKCS1v1.5-Padding als anfälliger gegenüber Angriffen erwiesen, die Seitenkanalinformationen oder Implementierungsfehler ausnutzen.

1.6. Weitere relevante Aspekte

Abschließend werden an dieser Stelle explizit noch einmal einige wichtige Themenbereiche genannt, die in der vorliegenden Technischen Richtlinie nicht oder nicht ausführlich behandelt werden. Die Auflistung erhebt ausdrücklich keinen Anspruch auf Vollständigkeit.

Schlüssellängen bei langfristig schützenswerten Informationen und in Systemen mit langer vorgesehener Einsatzdauer Unter *langfristig schützenswerten Informationen* sind für die Zwecke dieses Abschnitts solche Informationen zu verstehen, deren Vertraulichkeit deutlich länger gewahrt bleiben soll als es dem Zeitraum entspricht, für den diese Richtlinie Prognosen über die Eignung kryptographischer Verfahren ausspricht, das heißt deutlich über das Jahr 2031 hinaus. Eine zuverlässige Prognose über die Eignung von kryptographischen Verfahren über den gesamten Lebenszyklus eines Systems hinweg ist in diesem Fall nicht mehr möglich. Es wird empfohlen, unter Hinzuziehung eines Experten über die Minimalforderungen dieser Richtlinie wesentlich hinausgehende Schutzmechanismen vorzusehen. Beispielhaft seien nachfolgend verschiedene Wege zur Risikominimierung erläutert:

- Bei der Neuentwicklung von kryptographischen Systemen mit vorgesehener langer Einsatzdauer wird dazu geraten, die Möglichkeit eines künftigen Betriebs mit höheren Schlüssellängen schon bei der Entwicklung vorzusehen. Auch eine möglicherweise in der Zukunft entstehende Notwendigkeit zum Wechsel der eingesetzten Verfahren beziehungsweise die Durchführung solcher Verfahrenswechsel sollte schon während der Entwicklung des ursprünglichen Systems berücksichtigt werden (Kryptoagilität).

- Bereits bei Einführung des Systems sollten höhere asymmetrische Schlüssellängen als in dieser Richtlinie gefordert eingesetzt werden. Eine naheliegende Möglichkeit besteht darin, für alle Systemkomponenten ein einheitliches Sicherheitsniveau von $\geq$ 128 Bits anzustreben. Hinweise zu den für verschiedene Sicherheitsniveaus minimal erforderlichen asymmetrischen Schlüssellängen können aus Tabelle 2.3 entnommen werden.

- Insgesamt sollte die Menge an Informationen mit langfristigem Schutzbedarf, die über öffentliche Netzwerke übermittelt werden, auf das unbedingt notwendige Maß reduziert werden. Dies gilt insbesondere für Informationen, die mit einem hybriden oder asymmetrischen Kryptoverfahren verschlüsselt übertragen werden.

Für eine ausführlichere Diskussion über langfristig sichere Schlüssellängen in den bislang verbreiteten asymmetrischen kryptographischen Verfahren sei auf [48, 75] verwiesen.

Lightweight Cryptography In diesem Zusammenhang treten besonders restriktive Anforderungen an Rechenzeit und Speicherbedarf der eingesetzten kryptographischen Verfahren auf. Abhängig von der Anwendung können sich außerdem auch die Sicherheitsanforderungen von den sonst üblichen unterscheiden.

Reaktionszeiten eines Systems Beim Einsatz kryptographischer Verfahren in Bereichen, in denen enge Vorgaben an die Antwortzeiten des Systems eingehalten werden müssen, können besondere Situationen auftreten, die in dieser Richtlinie nicht behandelt werden. Die Empfehlungen zur Verwendung von SRTP in Appendix C decken Teile dieses Themas ab.

Festplattenverschlüsselung Im Kontext der Festplattenverschlüsselung tritt das Problem auf, dass in den meisten Anwendungsszenarien weder eine Verschlüsselung mit Datenexpansion noch eine deutliche Expansion der Menge an Daten, die vom Speichermedium gelesen beziehungsweise auf das Speichermedium geschrieben werden müssen, akzeptabel sind. Keiner der empfohlenen Verschlüsselungsmodi ist ohne Weiteres als Grundlage einer Lösung zur Festplattenverschlüsselung geeignet. Unter der Voraussetzung, dass ein Angreifer keine Abbilder des Festplattenzustandes zu mehreren verschiedenen Zeitpunkten miteinander kombinieren kann, bietet XTS-AES relativ gute Sicherheitseigenschaften und gute Effizienz [91]. Wenn der Angreifer zu einer größeren Anzahl verschiedener Zeitpunkte Kopien des verschlüsselten Speichermediums erstellen kann, ist jedoch von

einem gewissen, nicht zwingend unwesentlichen Abfluss von Information auszugehen. Der Angreifer kann zum Beispiel durch den Vergleich zweier zu verschiedenen Zeitpunkten angefertigter Abbilder einer mit XTS-AES verschlüsselten Festplatte unmittelbar erkennen, welche Klartextblöcke auf der Festplatte innerhalb dieses Zeitraumes verändert wurden und welche nicht.

Festplattenverschlüsselung von SSD-Platten Im Zusammenhang mit der Verschlüsselung eines Solid State Drives (SSD) ist der Umstand von Bedeutung, dass der SSD-Controller das Überschreiben logischer Speicheradressen physisch nicht in-place umsetzt, sondern auf verschiedene physische Speicherbereiche verteilt. Damit enthält der aktuelle Zustand einer SSD immer auch Information über gewisse frühere Zustände des Speichermediums. Ein Angreifer mit guter Kenntnis der Funktionsweise des SSD-Controllers kann dies potenziell ausnutzen, um aufeinanderfolgende Zustände einer logischen Speicheradresse nachzuverfolgen. Ein einzelnes Abbild des verschlüsselten Speichermediums ist bei Verwendung einer SSD damit für einen Angreifer unter Umständen wertvoller als ein einzelnes Abbild einer Festplatte.

Cloud-Speicherung Ähnliche Probleme wie bei der Verschlüsselung von Datenträgern stellen sich bei der verschlüsselten Speicherung ganzer logischer Laufwerke auf entfernten Systemen, die nicht unter der Kontrolle des Datenbesitzers stehen (sogenannte Cloud-Speicherung). Kann dem Anbieter des entfernten Servers oder dessen Sicherheitsmaßnahmen nicht in hohem Maße vertraut werden, muss davon ausgegangen werden, dass ein Angreifer unbemerkt Platten-Abbilder zu beliebigen Zeitpunkten anfertigen kann. Werden Dateien mit sensiblen Daten auf einem Speichersystem abgelegt, das regelmäßig unter fremder Kontrolle steht, sollte entsprechend vor der Übermittlung eine kryptographisch starke Dateiverschlüsselung angewandt werden. Dies gilt auch dann, wenn die Daten vor ihrer Übermittlung an das Speichermedium durch eine Volume-Verschlüsselung verschlüsselt werden. Die Verwendung einer Volume-Verschlüsselungslösung allein ist nur empfehlenswert, wenn diese einen wirksamen kryptographischen Schutz vor Manipulation der Daten beinhaltet und die sonstigen Voraussetzungen an den Einsatz des entsprechenden Verfahrens in allgemeinen kryptographischen Kontexten eingehalten werden (beispielsweise ist dies die Erfordernis unvorhersagbarer Initialisierungsvektoren). Insbesondere sollten Verfahren so ausgewählt werden, dass anders als im XTS-Modus bei wiederholtem Schreiben eines Datenblocks kein nennenswerter Abfluss von Information durch Häufigkeitsanalyse aufeinanderfolgender Zustände zu erwarten ist.

Physikalische Aspekte Die vorliegende Richtlinie geht im Wesentlichen nur auf solche Aspekte der Sicherheit kryptographischer Systeme ein, die sich auf die verwendeten Algorithmen reduzieren lassen. Physikalische Aspekte wie die Abstrahlsicherheit informationsverarbeitender Systeme oder kryptographische Systeme, deren Sicherheit auf physikalischen Effekten beruht (zum Beispiel quantenkryptographische Systeme), werden in dieser Technischen Richtlinie – ebenso wie Seitenkanalangriffe, Fault-Attacken und weitere physikalische Sicherheitsfragen – nicht oder nur am Rande behandelt. Etwaige Ausführungen zu Seitenkanalangriffen sind ausdrücklich als beispielhafte Hinweise auf mögliche Gefährdungen ohne Anspruch auf Vollständigkeit zu verstehen.

Verkehrsflussanalyse Keines der in dieser Technischen Richtlinie beschriebenen Verfahren und Protokolle zur Datenverschlüsselung erreicht für sich genommen das Ziel der *Sicherheit gegen Verkehrsflussanalyse* (englisch *Traffic Flow Confidentiality*). Eine Verkehrsflussanalyse – also eine Analyse eines verschlüsselten Datenstromes unter Berücksichtigung von Quelle, Ziel, Zeitpunkt des Bestehens einer Verbindung, Größe der übermittelten Datenpaketen, Datenrate und Zeitpunkt der Übermittlung der Datenpakete – kann wesentliche Rückschlüsse auf die Inhalte verschlüsselter Übertragungen erlauben, siehe zum Beispiel [6, 43, 116]. Traffic Flow Confidentiality ist ein Ziel, das im Regelfall nur mit hohem Aufwand vollständig erreicht werden kann und das deshalb auch in vielen Anwendungen, die sensible Informationen verarbeiten, nicht realisierbar ist. Es sollte allerdings

in jedem Einzelfall durch Experten überprüft werden, in welchem Umfang und welche vertrauliche Informationen in einem gegebenen Kryptosystem durch Verkehrsflussanalyse (und natürlich andere Seitenkanalangriffe) preisgegeben werden. Je nach konkreter Sachlage kann der Ausgang einer solchen Untersuchung wesentliche Änderungen am Gesamtsystem notwendig machen. Es wird daher empfohlen, die Widerstandsfähigkeit eines kryptographischen Systems gegen Preisgabe sensibler Informationen durch Verkehrsflussanalyse bei der Entwicklung neuer Systeme von Anfang an als Ziel zu berücksichtigen.

Endpunktsicherheit Die Sicherheit der Endpunkte einer kryptographisch abgesicherten Verbindung ist unabdingbar für die Sicherheit der übermittelten Daten. Bei der Entwicklung eines kryptographischen Systems muss eindeutig dokumentiert werden, welche Systemkomponenten vertrauenswürdig sein müssen, damit die angestrebten Sicherheitsziele erreicht werden, und diese Komponenten müssen in einer dem Einsatzkontext angemessenen Weise gegen Kompromittierung gehärtet werden. Entsprechende Überlegungen müssen den gesamten Lebenszyklus der zu schützenden Daten ebenso umfassen wie den gesamten Lebenszyklus der durch das System erzeugten kryptographischen Geheimnisse. Kryptographische Verfahren können die Anzahl der Komponenten eines Gesamtsystems, deren Vertrauenswürdigkeit sicherzustellen ist, um einen Datenabfluss zu vermeiden, zwar verringern, das Grundproblem der Endpunktsicherheit aber nicht lösen.

Die vorliegende Richtlinie liefert hinsichtlich der Umsetzung von Verfahren in den zuvor genannten Bereichen keine oder zumindest keine umfassenden Empfehlungen. Es wird daher geraten, bei der Entwicklung kryptographischer Systeme insgesamt – aber insbesondere in diesen Bereichen – von Beginn an Experten aus den entsprechenden Gebieten in die Entwicklungsarbeit mit einzubeziehen.

2. Asymmetrische Verschlüsselung und Schlüsseleinigung

Schlüsseleinigungsverfahren dienen der Aushandlung eines gemeinsamen geheimen Schlüssels über einen unsicheren Kanal. Asymmetrische Verschlüsselungsverfahren (auch Public-Key-Verschlüsselungsverfahren genannt) werden in der Praxis meist lediglich zur Übertragung symmetrischer Schlüssel eingesetzt, da sie wesentlich ineffizienter als symmetrische Verschlüsselungsverfahren sind. In diesem Kapitel werden sowohl asymmetrische Verfahren zur Verschlüsselung als auch zur Schlüsseleinigung behandelt.

Die Sicherheit von derzeitigen „klassischen" asymmetrischen Verfahren basiert auf der angenommenen Schwierigkeit bestimmter mathematischer Probleme, wobei „klassisch" im Sinne von „bietet Schutz vor Angriffen, die auf klassischer Hardware ausführbar sind" zu verstehen ist. Bei den in der Praxis am weitesten verbreiteten asymmetrischen Verfahren handelt es sich hierbei um das Faktorisierungsproblem beziehungsweise das Diskreter-Logarithmus-Problem (DL).

Auch wenn zurzeit keine effizienten *klassischen* Algorithmen zur Lösung dieser Probleme bekannt sind, ist seit den 1990er Jahren mit *Shors Algorithmus* [115] ein entsprechender Quantenalgorithmus bekannt. Bei Verfügbarkeit eines hinreichend großen Quantencomputers ist Shors Algorithmus in der Lage, das Faktorisierungsproblem und das Diskrete Logarithmus-Problem effizient zu lösen und somit die klassischen Verschlüsselungs- und Schlüsseleinigungsverfahren zu brechen. Deshalb ist für Systeme zum Schutz langfristig schützenswerter Informationen der Einsatz quantensicherer Schlüsseleinigungsverfahren (siehe Abschnitt 2.4) ratsam. Da die quantensicheren Verfahren vergleichsweise neu beziehungsweise insbesondere in Bezug auf Implementierungssicherheit weniger untersucht sind, wird in dieser Technischen Richtlinie derzeit ausschließlich der hybride Einsatz (siehe Abschnitt 2.2) von quantensicheren Verfahren in Kombination mit klassischen Verfahren empfohlen.

Schlüsseleinigungsverfahren müssen zwingend mit Authentisierungsverfahren kombiniert werden, um entscheiden zu können, mit welchem Kommunikationspartner die Schlüsseleinigung durchgeführt wird. In diesem Kapitel werden nur die Schlüsseleinigungsverfahren beziehungsweise Verschlüsselungsverfahren ohne den zusätzlich benötigten Authentisierungsmechanismus beschrieben. Für eine sichere Schlüsseleinigung müssen die hier beschriebenen Verfahren in entsprechende Protokolle eingebettet werden, die unter anderem auch die Authentizität der Kommunikationspartner sicherstellt. Aus diesem Grund werden im Folgenden nur allgemeine Ideen für Schlüsseleinigungsverfahren angegeben und für konkrete Verfahren, das heißt für Schlüsseleinigungsverfahren, die auch eine Instanzauthentisierung beinhalten, wird auf Abschnitt A.2 verwiesen.

Nach erfolgreicher Schlüsseleinigung befinden sich beide Kommunikationspartner im Besitz eines gemeinsamen Geheimnisses (beziehungsweise mehrerer Geheimnisse im Falle hybrider Schlüsseleinigung). Für empfohlene Verfahren zur Generierung symmetrischer Schlüssel aus diesem Geheimnis beziehungsweise diesen Geheimnissen siehe Abschnitt 2.2.

Im Wesentlichen wird für diese Aufgabe die Verwendung einer Schlüsselableitungsfunktion empfohlen. In manchen Situationen kann es sinnvoll sein, ein vorverteiltes Geheimnis in die Schlüsselableitungsfunktion eingehen zu lassen. Damit kann zum Beispiel eine Separierung verschiedener Benutzergruppen erreicht werden. Auch ein zusätzlicher Schutz gegen Angriffe auf das Schlüsseleinigungsverfahren kann auf diese Weise erreicht werden. Hinsichtlich einer Separierung verschiedener Benutzergruppen kann es zudem sinnvoll sein, weitere öffentliche Daten, die

spezifisch für beide Kommunikationspartner sind, bei der Schlüsselableitung zu berücksichtigen.

Bemerkung 2.1 Werden kryptographische Schlüssel mit einem Schlüsseleinigungsverfahren ausgehandelt oder mit einem Schlüsseltransportverfahren gesichert übermittelt, so besitzen diese Schlüssel beziehungsweise die kryptographischen Verfahren, die diese Schlüssel verwenden, maximal das gleiche Sicherheitsniveau wie das Schlüsseleinigungs- beziehungsweise Schlüsseltransportverfahren. Da die Möglichkeit besteht, dass ein Angreifer die Kommunikation bei der Schlüsseleinigung beziehungsweise beim Schlüsseltransport aufzeichnet, haben geänderte Empfehlungen für Schlüsseleinigungs- oder Schlüsseltransportverfahren auch Auswirkungen auf zuvor ausgehandelte beziehungsweise übertragene Schlüssel. Wenn beispielsweise das Schlüsseleinigungs- beziehungsweise Schlüsseltransportverfahren die Konformität zu dieser Technischen Richtlinie verliert, so sollten auch die damit ausgehandelten beziehungsweise übertragenen Schlüssel nicht mehr eingesetzt werden.

Die hier empfohlenen asymmetrischen kryptographischen Verfahren benötigen als Bestandteile weitere Unterkomponenten, wie Hashfunktionen, Message Authentication Codes, Zufallszahlenerzeugung, Schlüsselableitungsfunktionen und/oder Blockchiffren, die ihrerseits zur Erreichung des angestrebten Sicherheitsniveaus ebenfalls den Anforderungen der vorliegenden Richtlinie genügen müssen. In einschlägigen Standards [62] wird teilweise die Verwendung von Verfahren empfohlen, die in der vorliegenden Richtlinie nicht empfohlen werden. Grundsätzlich wird empfohlen, sich bei der Implementierung eines Standards an folgende Grundsätze zu halten:

- Für kryptographische Unterkomponenten sollten nur die jeweils in dieser Richtlinie empfohlenen Verfahren verwendet werden.
- Werden in dieser Richtlinie Verfahren empfohlen, die ausschließlich mit einer nicht empfohlenen Unterkomponente standardisiert sind, so ist diese Unterkomponente im Kontext des Verfahrens als empfohlen anzusehen.
- Sofern sich dennoch keine Standardkonformität erzielen lässt, ist ein Experte hinzuziehen und die letztlich getroffenen Entscheidungen hinsichtlich der gewählten kryptographischen Unterkomponenten sind ausführlich zu dokumentieren und unter Sicherheitsgesichtspunkten zu begründen.

Bei der Auswahl der empfohlenen asymmetrischen Verschlüsselungsverfahren wurde darauf geachtet, dass lediglich probabilistische Algorithmen[1] zum Einsatz kommen. Insbesondere wird also bei jeder Berechnung eines Chiffretextes ein neuer Zufallswert benötigt. Die Anforderungen an diese Zufallswerte sind teilweise nicht direkt durch die Erzeugung gleichverteilter Werte von fester Bitlänge zu erfüllen. Näheres zu diesen Zufallswerten wird in den Abschnitten zu den entsprechenden Verfahren angegeben.

Bemerkung 2.2 (Seitenkanalangriffe und Fault-Attacken) Je nach vorliegender Situation können für asymmetrische Verschlüsselungsverfahren und/oder asymmetrische digitale Signaturverfahren verschiedene Arten von Seitenkanalattacken und/oder Fault-Attacken von Bedeutung sein. Dieses Thema kann in der vorliegenden Richtlinie nicht umfassend behandelt werden. Die Sicherheit einer Implementierung gegen Seitenkanalangriffe und Fault-Attacken muss daher stets im Einzelfall überprüft werden. Weitere Informationen und detaillierte Empfehlungen zu diesem Thema finden sich in [38, 40, 41, 39].

Bemerkung 2.3 (Public-Key-Infrastrukturen) Die in der vorliegenden Richtlinie beschriebenen asymmetrischen Verschlüsselungsverfahren bieten für sich genommen noch keinerlei Schutz vor Man-in-the-Middle-Angriffen. Die Sicherheitsgarantien der beschriebenen Verfahren sind daher nur gültig, falls Man-in-the-Middle-Angriffe durch zusätzliche Mechanismen zuverlässig

[1] Der RSA-Algorithmus selbst ist nicht probabilistisch, dafür jedoch das hier empfohlene Paddingverfahren zu RSA.

verhindert werden können. Dafür muss eine authentische Verteilung der öffentlichen Schlüssel aller Teilnehmer sichergestellt werden.

Dies kann auf verschiedene Arten geschehen, in der Regel kommt eine Public-Key-Infrastruktur (PKI) zum Einsatz. In einer PKI wird das Problem der authentischen Verteilung öffentlicher Schlüssel auf die Verteilung der Wurzelzertifikate der PKI reduziert. Bei der Planung einer PKI für ein asymmetrisches Verschlüsselungs- oder Signaturverfahren wird empfohlen, die folgend aufgelisteten Aspekte zu berücksichtigen. Es handelt sich hierbei nicht um eine erschöpfende Aufzählung von Entwicklungsanforderungen an Public-Key-Infrastrukturen, sondern lediglich um eine Liste mit vergleichsweise generischen Punkten, zu deren Beachtung bei der Entwicklung einer PKI angeraten wird; weitere Informationen finden sich auch in [29]. Bei der Entwicklung und Evaluierung eines konkreten Systems ergeben sich in der Regel weitere Anforderungen, die hier nicht berücksichtigt sind. Die Entwicklung einer geeigneten PKI für eine neue kryptographische Anwendung ist keine triviale Aufgabe und sollte daher nur in enger Abstimmung mit entsprechenden Experten angegangen werden.

- Bei der Ausstellung von Zertifikaten sollte die PKI überprüfen, dass der Antragsteller im Besitz eines privaten Schlüssels zu seinem öffentlichen Schlüssel ist. Dies kann zum Beispiel durch ein Challenge-Response-Verfahren zur Instanzauthentisierung geschehen, das eine Kenntnis des privaten Schlüssels voraussetzt. Auch eine Erzeugung der Schlüsselpaare in einer aus Sicht der PKI sicheren Umgebung ist denkbar, wenn sie mit einem sicheren Transport der erzeugten Schlüsselpaare zum Endnutzer verbunden wird.
- Es sollte Möglichkeiten zur zeitnahen Deaktivierung von Zertifikaten geben und es sollte einem Angreifer nicht möglich sein, unbemerkt zu verhindern, dass einem Teilnehmer zum Zeitpunkt der Prüfung die Information über den aktuellen Status eines Zertifikats zur Verfügung steht.
- Zertifikate sollten nur zeitlich befristet ausgestellt werden.
- Alle Zertifikatsaussteller müssen vertrauenswürdig sein.
- Aus einem Zertifikat sollte hervorgehen, ob es zur Signierung weiterer Zertifikate berechtigt. Generell sollte jedes System, das mit einem Zertifikat in Berührung kommt, eindeutig ermitteln können, wozu dieses Zertifikat verwendet werden darf.
- Die Länge von Zertifikatsketten sollte (durch einen möglichst niedrigen Wert) nach oben beschränkt werden.

2.1. Einsatz quantensicherer Verfahren

Die heute eingesetzten klassischen, asymmetrischen Schlüsseleinigungs- und Verschlüsselungsverfahren sind durch die fortschreitende Entwicklung hinreichend großer Quantencomputer bedroht. Auch wenn heute verfügbare Quantencomputer noch nicht in der Lage sind, kryptographische Verfahren zu brechen, ist die Bedrohung insbesondere für Daten mit längerfristigem Schutzbedarf schon heute relevant, da verschlüsselte Daten bereits jetzt zur späteren Entschlüsselung gespeichert werden können („Store Now, Decrypt Later“). Zudem müssen potenziell lange Migrationszeiten zu neuen kryptograpischen Verfahren berücksichtigt werden. Vor diesem Hintergrund hat das BSI gemeinsam mit europäischen Partnerbehörden [42] das Ziel konkretisiert, die Migration zu quantensicheren Verfahren zum Schutz vor dem „Store Now, Decrypt Later“-Szenario für hochsensitive Anwendungen bis spätestens Ende 2030 abzuschließen. Die vorliegende technische Richtlinie empfiehlt daher, quantensichere Verfahren zum langfristigen Schutz der Vertraulichkeit einzusetzen; in künftigen Versionen werden Verfahren, die nicht quantensicher sind, nur noch für den hybriden

Einsatz empfohlen (eventuell mit einer kürzeren Frist als den üblichen 7 Jahren). Ein alleiniger Einsatz von klassischen Schlüsseleinigungsverfahren (RSA bzw. ECC) sollte nur nach einer sorgfältigen Risikoanalyse erfolgen. Ebenso sollten die in Abschnitt 2.4 empfohlenen quantensicheren Verfahren hybrid, also in geeigneter Kombination mit einem klassischen Verfahren, eingesetzt werden. Weitere Details zur Kombination von klassischen Schlüsseleinigungsverfahren und quantensicheren Verfahren finden sich in Abschnitt 2.2.

Bemerkung 2.4 Im Kontext der Post-Quanten-Kryptographie wird ein kryptographisches Verfahren, dessen geheimer Schlüssel auf der Kombination eines geheimen Schlüssels aus einem quantensicheren Verfahren mit einem geheimen Schlüssel aus einem klassischen Verfahren besteht, auch als „hybrides Verfahren“ oder „hybride Schlüsseleinigung“ bezeichnet. Dieser Begriff ist nicht zu verwechseln mit dem Begriff „Hybride Verschlüsselung“, welcher die Kombination aus symmetrischer und asymmetrischer Verschlüsselung meint. In der Regel ist es jedoch aus dem Kontext ersichtlich, um welche Lesart es sich handelt, so dass im Folgenden auf eine weitere Klarstellung verzichtet wird.

Ein grundsätzlich anderer Ansatz zur Schlüsseleinigung als die Post-Quanten-Kryptographie (englisch Post-Quantum Cryptography, PQC) bietet die Quantum Key Distribution (QKD). Im Gegensatz zu Post-Quanten-Kryptographie adressiert QKD das Problem eines sicheren Kommunikationsaufbaus, indem quantenphysikalische Effekte ausgenutzt werden. Allerdings sind die praktischen Einschränkungen von QKD, wie beispielsweise beschränkte Reichweiten und die Notwendigkeit des Einsatzes spezialisierter Hardware, im Vergleich zur Verwendung von PQC-Verfahren stark limitierend und führen dazu, dass QKD nur für spezielle Anwendungsfälle geeignet ist. Ferner ist QKD aus Sicherheitssicht nach Auffassung des BSI derzeit nicht einsatzreif. Aus diesen Gründen spricht das BSI zu diesem Zeitpunkt keine Empfehlung von QKD-Protokollen aus. Details zu diesem Thema finden sich einem Positionspapier zu QKD [53], welches das BSI gemeinsam mit europäischen Partnerbehörden veröffentlicht hat.

2.2. Schlüsselableitung und Hybridisierung

Nach einem Schlüsseleinigungsverfahren sind beide Parteien im Besitz eines gemeinsamen Geheimnisses, aus dem symmetrische Schlüssel, zum Beispiel für die Verschlüsselung und zur Datenauthentisierung (siehe auch Bemerkung A.2), mithilfe einer Schlüsselableitungsfunktion erzeugt werden können. Für empfohlene Verfahren zur Schlüsselableitung sei auf Abschnitt B.1.1 verwiesen.

Den quantensicheren Verfahren, die in dieser Technischen Richtlinie empfohlen werden, wird im Allgemeinen noch nicht das gleiche Vertrauen entgegengebracht wie den etablierten klassischen Verfahren, da sie beispielsweise im Hinblick auf Seitenkanalresistenz und Implementierungssicherheit nicht gleich gut untersucht sind. Um die langfristige Sicherheit einer Schlüsseleinigung zu gewährleisten, empfiehlt diese Technische Richtlinie daher den Einsatz eines hybriden Schlüsseleinigungsverfahrens, bei dem ein quantensicheres mit einem klassischen Verfahren kombiniert wird. Eine naheliegende Hybridisierung ist dadurch gegeben, zwei Schlüsseleinigungen parallel durchzuführen und aus dem erzeugten Schlüsselmaterial einen kombinierten Schlüssel abzuleiten. Die hybride Schlüsseleinigung sollte sicher sein, solange eines der verwendeten Verfahren sicher ist. Dabei ist zu beachten, wie genau das einzelne Schlüsselmaterial miteinander kombiniert und kontextabhängige Informationen miteinbezogen werden, damit die zuvor genannte Eigenschaft tatsächlich erfüllt wird.

Zur Kombination von Schlüsselmaterial wird in dieser Technischen Richtlinie folgendes Verfahren empfohlen:

- CatKDF, siehe [50].

Tabelle 2.1: Empfohlene Verfahren zur Kombination von Schlüsselmaterial.

Bemerkung 2.5

- In [107] wird eine vergleichbare Konstruktion definiert. Es ist beabsichtigt, in zukünftigen Versionen der vorliegenden Technischen Richtlinie Konstruktionen aus [107] aufzugreifen.
- Die obigen Empfehlungen sind allgemeiner Natur. Darüber hinaus gibt es noch speziell auf Protokolle zugeschnittene Verfahren, die nicht in diesem Teil der Technischen Richtlinie behandelt werden.

2.3. Klassische asymmetrische Verfahren

Die derzeit praktisch relevantesten asymmetrischen Verschlüsselungs- und Signaturverfahren beruhen vereinfacht ausgedrückt entweder auf der Schwierigkeit des Problems der Berechnung diskreter Logarithmen in geeigneten Repräsentationen endlicher zyklischer Gruppen (Diskreter-Logarithmus-Problem (DL)) oder auf der Schwierigkeit, große ganze Zahlen in ihre Primfaktoren zu zerlegen (Faktorisierungsproblem). Es taucht gelegentlich die Frage auf, welcher dieser beiden Ansätze als kryptographisch sicherer einzuschätzen ist. Die vorliegende Technische Richtlinie sieht die Faktorisierung großer Zahlen, das RSA-Problem, das Problem der Berechnung diskreter Logarithmen in geeigneten Körpern $\mathbb{F}_p$ (p prim), das Problem der Berechnung diskreter Logarithmen in geeigneten elliptischen Kurven, und die entsprechenden Diffie-Hellman-Probleme als gut untersuchte, schwere Probleme an und es gibt in dieser Hinsicht keinen Grund, auf Faktorisierung basierende Verfahren gegenüber Verfahren auf Grundlage diskreter Logarithmen zu bevorzugen oder umgekehrt. Für besonders hohe Sicherheitsniveaus ist die Verwendung von EC-Verfahren aus Effizienzgründen vorteilhaft, siehe hierzu auch Tabelle 2.3.

Bemerkung 2.6 Für asymmetrische Verfahren gibt es in der Regel verschiedene äquivalente, praktisch relevante Darstellungen der privaten und öffentlichen Schlüssel. Die Bitlänge der Schlüssel in einem Datenspeicher kann dabei je nach gewählter Repräsentation der Schlüssel unterschiedlich ausfallen. Für die exakte Definition der Schlüssellänge für die empfohlenen asymmetrischen kryptographischen Verfahren wird daher auf den Eintrag Schlüssellänge im Glossar verwiesen.

Die folgende Tabelle 2.2 gibt einen Überblick über die empfohlenen klassischen asymmetrischen Verschlüsselungs- und Schlüsseleinigungsverfahren sowie Schlüssellängen l in Bits.

Verfahren	RSA	DLIES	ECIES	DH	ECDH
Schlüssellänge l in Bits	3000	3000	250	3000	250
Referenz	[86]	[1]	[1, 62]	[1, 112]	[1, 112]
Näheres in	Abschnitt 2.3.2	Abschnitt 2.3.3	Abschnitt 2.3.4	Abschnitt 2.3.5	Abschnitt 2.3.6

Tabelle 2.2: Empfohlene klassische asymmetrische Verschlüsselungs- und Schlüsseleinigungsverfahren sowie Schlüssellängen und Referenzen.

Bemerkung 2.7 Aus den aktuellen Empfehlungen resultiert nur noch ein geringer Puffer zwischen dem durch die empfohlenen EC-Bitlängen mindestens erreichten Sicherheitsniveau von etwa 125

Bits und dem in dieser Richtlinie angestrebten Sicherheitsniveau von 120 Bits. In bestimmten Anwendungen, die besonders hohe Sicherheitsanforderungen haben oder deren Sicherheit deutlich über den Vorhersagezeitraum dieser Technischen Richtlinie hinaus sichergestellt werden muss, kann es zur Vergrößerung des Sicherheitspuffers daher sinnvoll sein, deutlich größere Schlüssellängen für EC-Verfahren vorzusehen. Die Vorgaben zur Schlüssellänge der Country Signer CA aus [28] lassen sich beispielsweise auf diese Art erklären. Da die Sicherheit von EC-Verfahren von der Annahme abhängt, dass ein Angreifer nichts von der mathematischen Struktur einer gegebenen elliptischen Kurve nutzen kann, um diskrete Logarithmen schneller zu berechnen als es der Pollard-Rho-Algorithmus erlaubt, ist es denkbar, dass in den kommenden Jahren die Anforderungen der vorliegenden Technischen Richtlinie in diesem Bereich als grundsätzliche Vorsichtsmaßnahme erhöht werden. Es wird ferner als grundsätzliche Sicherheitsmaßnahme empfohlen, in EC-Verfahren Kurvenparameter zu verwenden, die nachweisbar zufällig erzeugt wurden, deren Konstruktion nachvollziehbar dokumentiert ist und deren Sicherheit einer gründlichen Analyse unterzogen wurde. Ein Beispiel für solche Kurvenparameter stellen die Brainpool-Kurven [79] dar.

2.3.1. Äquivalente Schlüssellängen für symmetrische und klassische asymmetrische kryptographische Verfahren

Den Empfehlungen der vorliegenden Technischen Richtlinie zu den Schlüssellängen klassischer asymmetrischer kryptographischer Verfahren liegen Berechnungen zu Äquivalenzen symmetrischer und asymmetrischer Schlüssellängen zugrunde, die auf den folgenden Grundannahmen basieren:

- Für Verfahren basierend auf elliptische Kurven: Es wird angenommen, dass über den Vorhersagezeitraum dieser Technischen Richtlinie hinweg keine Methode existiert, das Diffie-Hellman-Problem auf der verwendeten Kurve wesentlich schneller zu lösen als die Berechnung diskreter Logarithmen auf derselben Kurve. Es wird weiterhin angenommen, dass die Berechnung diskreter Logarithmen auf der verwendeten elliptischen Kurve nicht mit wesentlich geringerer Komplexität (gemessen an der Anzahl der ausgeführten Gruppenoperationen) möglich ist als für generische Darstellungen der gleichen zyklischen Gruppe.[2] Für eine generische Gruppe G wird eine Komplexität der Berechnung diskreter Logarithmen von $\approx \sqrt{|G|}$ Gruppenoperationen angenommen.

- Für RSA und Verfahren basierend auf diskreten Logarithmen in $\mathbb{F}_p^*$: Es wird angenommen, dass über den Vorhersagezeitraum dieser Technischen Richtlinie hinweg keine Angriffe bekannt werden, die bei einer Wahl der Parameter wie in der vorliegenden Richtlinie empfohlen effizienter sind als das allgemeine Zahlkörpersieb. Für RSA und Verfahren basierend auf diskreten Logarithmen in $\mathbb{F}_p^*$ werden gleiche Schlüssellängen empfohlen. Im Fall von Verfahren basierend auf diskreten Logarithmen wird angenommen, dass kein Verfahren existiert, um das Diffie-Hellman-Problem in einer Untergruppe $U \subset \mathbb{F}_p^*$ mit $\mathrm{ord}(U)$ prim effizienter zu lösen als durch Berechnung diskreter Logarithmen in U.

Diese Annahmen sind insofern aus Angreifersicht pessimistisch, als dass sie keinen Spielraum für strukturelle Fortschritte in der Kryptoanalyse asymmetrischer Verfahren enthalten. Fortschritte, die mit den obigen Annahmen inkompatibel sind, können von sehr spezieller Natur sein und sich zum Beispiel auf neue Erkenntnisse zu *einer einzigen* elliptischen Kurve beziehen. Obwohl grundsätzlich eine Berechnung mit 2^{120} Elementaroperationen für den für diese Richtlinie relevanten Zeitraum als nicht praktisch durchführbar angesehen wird, liegen alle empfohlenen Schlüssellängen oberhalb des in dieser Richtlinie minimal angestrebten 120-Bit-Sicherheitsniveaus.

[2] Algorithmen, die auf einer generischen Darstellung einer Gruppe operieren, haben auf Elemente und Gruppenoperationen nur Black-Box-Zugriff. Intuitiv kann man sich etwa ein Orakel vorstellen, das verschlüsselte Gruppenelemente annimmt und das Ergebnis von Gruppenoperationen verschlüsselt ausgibt.

$\log_2(R)$	ECDLP	Faktorisierung/DLP in $\mathbb{F}_p^*$
60	120	700
70	140	1000
100	200	1900
128	256	3200
192	384	7900
256	512	15500

Tabelle 2.3: Ungefährer Rechenaufwand R (in Vielfachen des Rechenaufwandes für eine einfache kryptographische Operation, zum Beispiel das einmalige Auswertung einer Blockchiffre auf einem Block) für die Berechnung diskreter Logarithmen in elliptischen Kurven (ECDLP) beziehungsweise die Faktorisierung allgemeiner zusammengesetzter Zahlen mit den angegebenen Bitlängen.

Im Hinblick auf Verfahren, deren Sicherheit auf der Schwierigkeit der Berechnung diskreter Logarithmen beruht, insbesondere diskreter Logarithmen in elliptischen Kurven, können auch Angriffe relevant sein, die einen Orakel-Zugriff auf Operationen mit dem privaten Schlüssel eines Nutzers benötigen. Solche Angriffe können die Berechnung diskreter Logarithmen in einer Gruppe deutlich beschleunigen, siehe etwa Angriffe unter Nutzung eines Static-Diffie-Hellman-Orakels [22, 44].

Zur Abschätzung von Laufzeiten folgen wir [49]. Insbesondere wird wie in [49] angenommen, dass die Faktorisierung einer 512-Bit-Zahl von beliebiger Form etwa dem Rechenaufwand von 2^{50} DES-Operationen entspricht. Unter Verwendung der dort angegebenen Methoden ergeben sich – ohne jegliche Sicherheitsmargen für Fortschritte im Hinblick auf Faktorisierungstechniken beziehungsweise Techniken zur effizienten Berechnung diskreter Logarithmen in den jeweiligen Gruppen – ungefähr die in Tabelle 2.3 wiedergegebenen Äquivalenzen (vergleiche [49, Tabelle 7.2] und [48, Tabelle 4.1]).

Hinsichtlich empfohlener Schlüssellängen sei auf Tabelle 2.2 verwiesen.

2.3.2. RSA-Verschlüsselung

Das RSA-Verfahren, benannt nach seinen Erfindern R. Rivest, A. Shamir und L. Adleman, ist ein asymmetrisches kryptographisches Verfahren, das sowohl zum Verschlüsseln als auch zum digitalen Signieren verwendet werden kann. Es verwendet ein Schlüsselpaar, bestehend aus einem privaten Schlüssel, der zum Entschlüsseln oder Signieren von Daten verwendet wird, und einem öffentlichen Schlüssel, mit dem man verschlüsselt oder Signaturen prüft. Die Sicherheit des Verfahrens beruht auf der angenommenen Schwierigkeit, ganze Zahlen in das Produkt ihrer Primfaktoren zu zerlegen.

Schlüsselgenerierung

1.) Wähle zwei Primzahlen p und q zufällig und unabhängig voneinander aus. Die Zahlen p und q sollten von vergleichbarer Bitlänge sein und nicht zu nah beieinander liegen, da andernfalls, falls beispielsweise p und q unabhängig voneinander aus einem zu kleinen Intervall gewählt werden, Angriffe basierend auf Kenntnis der führenden Bits von p und q möglich sind.

Nähere Hinweise zur Vorgehensweise bei der Primzahlerzeugung finden sich in Abschnitt B.5. Bei einer Wahl von p und q entsprechend Abschnitt B.5 tritt die zuvor genannte Sicherheitslücke nicht auf.

2.) Wähle den öffentlichen Exponenten $e \in \mathbb{N}$ unter den Nebenbedingungen

$$\mathrm{ggT}(e, (p-1)\cdot(q-1)) = 1 \quad \text{und} \quad 2^{16}+1 \leq e \leq 2^{256}-1.$$

3.) Berechne den privaten Exponenten $d \in \mathbb{N}$ in Abhängigkeit von e unter der Nebenbedingung

$$e \cdot d = 1 \bmod \mathrm{kgV}(p-1, q-1).$$

Mit dem sogenannten Modulus $n = p \cdot q$ stellt dann (n, e) den öffentlichen Schlüssel und d den privaten Schlüssel dar. Zusätzlich müssen auch die beiden Primzahlen p und q geheim gehalten werden, da sonst jeder aus dem öffentlichen Schlüssel (n, e) wie unter Punkt 3. den privaten Exponenten berechnen kann. Es wird empfohlen, außer den erzeugten Schlüsseln keine Daten aus der Schlüsselgenerierung persistent abzuspeichern und alle erzeugten Daten nach der Schlüsselgenerierung im Arbeitsspeicher zu überschreiben. Es wird weiter empfohlen, den privaten Schlüssel auf einem geschützten Speichermedium und/oder verschlüsselt so abzuspeichern, dass nur berechtigte Nutzer Entschlüsselungs-Operationen durchführen können.

Bemerkung 2.8 (i) Die Reihenfolge der Wahl der Exponenten während der Schlüsselerzeugung, das heißt erst die Wahl von e und dann die von d, soll die zufällige Wahl kleiner privater Exponenten verhindern, siehe [19].

(ii) Bei der Verwendung probabilistischer Primzahltests zur Erzeugung der beiden Primzahlen p und q sollte die Wahrscheinlichkeit, dass eine der Zahlen doch zusammengesetzt ist, höchstens 2^{-120} betragen, siehe Abschnitt B.5 für geeignete Verfahren.

Ver- und Entschlüsselung Für die Ver- und Entschlüsselung sei auf den Standard [86] verwiesen. Dabei ist zu beachten, dass zusätzlich die Nachricht vor Anwendung des privaten Schlüssels d auf die Bitlänge des Modulus n formatiert werden muss. Das Formatierungsverfahren ist dabei sorgfältig zu wählen, empfohlen wird folgendes Verfahren:

EME-OAEP, siehe [86].

Tabelle 2.4: Empfohlenes Formatierungsverfahren für den RSA-Verschlüsselungsalgorithmus.

Eine Verwendung des älteren PKCS#1v1.5-Paddings wird nicht empfohlen, da sich dabei bereits mehrfach Varianten der Attacke von Bleichenbacher [16] als Problem erwiesen haben, siehe beispielsweise [17] für ein Beispiel aus der jüngeren Vergangenheit.

Schlüssellänge Die Länge des Modulus n sollte mindestens 3000 Bits betragen, siehe auch Tabelle 2.2. Eine notwendige Voraussetzung für die Sicherheit des RSA-Verfahrens ist es, dass es praktisch unmöglich ist, den Modul n ohne Kenntnis von p und q in seine Primfaktoren zu zerlegen. Bei der empfohlenen Mindestbitlänge von 3000 Bits ist das nach derzeitigem Kenntnisstand der Fall.

Bemerkung 2.9 Da der Modulus n sehr groß ist, sind auch die im Rechner verwendeten Bitdarstellungen der Zahlen sehr lang. Der Chinesische Restsatz erlaubt es, die Berechnungen beim Ver- oder Entschlüsseln oder dem Signieren von Nachrichten statt in einer Gruppe der Größe n in den zwei kleineren Gruppen der Größen p und q durchzuführen und das Ergebnis danach zusammenzusetzen. Da $p, q \ll n$, ist diese Berechnung insgesamt effizienter. Diese Variante wird nach der englischen Bezeichnung des Chinesischen Restsatzes CRT (Chinese remainder theorem) auch CRT-RSA genannt.

Der private Schlüssel besteht in diesem Fall aus den Komponenten $(n, d, p, q, d_p, d_q, q_{\text{inv}})$, wobei

$$d_p = d \bmod p - 1, \qquad d_q = d \bmod q - 1, \qquad q_{\text{inv}} = q^{-1} \bmod p.$$

Bemerkung 2.10 Das RSA-Verfahren kann auch zur Schlüsselvereinbarung genutzt werden, siehe Anhang A.2.

2.3.3. DLIES-Verschlüsselung

Ein *Discrete Logarithm Integrated Encryption Scheme* ist ein hybrides Verschlüsselungsverfahren, bei dem die Sicherheit der asymmetrischen Komponente auf der Schwierigkeit des Diffie-Hellman-Problems in einer geeigneten Untergruppe von $\mathbb{F}_p^*$ basiert. Im Folgenden wird eine Version von DLIES beschrieben, die mit den übrigen Empfehlungen der vorliegenden Technischen Richtlinie vereinbar ist, wobei sich in der Verfahrensbeschreibung eng an [1] angelehnt wird.

Ein DLIES benötigt folgende Komponenten:

- Symmetrisches Verschlüsselungsverfahren E_K: Alle in der vorliegenden Richtlinie empfohlenen Kombinationen aus Blockchiffre und Betriebsmodus sind hierfür geeignet.
- Message Authentication Code MAC_{KM}: Es können die in Abschnitt 5.2 empfohlenen Verfahren verwendet werden.
- Schlüsselableitungsfunktion H: H kann beispielsweise eine Hashfunktion sein, falls deren Ausgabe mindestens die Länge des gesamten abzuleitenden symmetrischen Schlüsselmaterials besitzt. Alternativ kann auch die in Abschnitt B.1 empfohlene oder eine der in [62] vorgeschlagenen Schlüsselableitungsfunktionen verwendet werden, um aus den gegebenen Daten abgeleitetes Schlüsselmaterial der gewünschten Länge zu erzeugen.

Darüber hinaus benötigt ein DLIES Schlüsselmaterial, wie im folgenden Abschnitt zur Schlüsselgenerierung beschrieben.

Schlüsselgenerierung

1.) Wähle zufällig eine Primzahl q von geeigneter Bitlänge (siehe Unterabschnitt zu Schlüssellängen).

2.) Wähle k zufällig von einer Bitlänge, die sicherstellt, dass kq von der Länge des zu erzeugenden Schlüssels ist. Wiederhole diesen Schritt, bis $p := kq + 1$ prim ist.

3.) Wähle nun ein $x \in \mathbb{Z}_p^*$ so, dass $x^k \neq 1 \bmod p$ und setze $g := x^k$. Dann ist g ein Element der Ordnung q in $\mathbb{Z}_p^*$.

4.) Wähle zufällig eine Zahl $a \in \{2, \ldots, q-1\}$ und setze $A := g^a$.

Dann bilden (p, g, A, q) den öffentlichen Schlüssel und a den privaten Schlüssel.

Verschlüsselung Gegeben seien eine Nachricht $M \in \{0,1\}^*$ und ein öffentlicher Schlüssel (p, g, A, q), der auf zuverlässige Weise dem berechtigten Empfänger E der Nachricht zugeordnet werden kann. Zur Verschlüsselung wählt der Sender S eine zufällige Zahl $b \in \{1, \ldots, q-1\}$ und berechnet $B := g^b$, $X := A^b$ und daraus $h := H(X)$. Aus h werden genügend Bits entnommen, um einen Schlüssel K für das symmetrische Verschlüsselungsverfahren sowie einen Schlüssel KM für den MAC zu bilden. Aus der Nachricht M berechnet S das Chiffrat $C := E_K(M)$ sowie einen MAC $T := \text{MAC}_{KM}(C)$ und sendet das Tripel (B, C, T) an den Empfänger E.

Entschlüsselung Der Empfänger E erhält (B, C, T) und berechnet $X := B^a$ sowie damit weiter $h := H(X)$, K und KM. Er berechnet $T' := \mathrm{MAC}_{KM}(C)$ und prüft, ob $T = T'$ ist. Ist dies nicht der Fall, bricht der Entschlüsselungsvorgang ab. Ist dagegen $T = T'$, dann erhält E durch $M = E_K^{-1}(C)$ die Nachricht zurück.

Schlüssellänge Die Länge der Primzahl p sollte mindestens 3000 Bits betragen, die Länge der Primzahl q sollte mindestens 250 Bits betragen.

Bemerkung 2.11 Das DLIES-Verfahren stellt einen probabilistischen Algorithmus dar, da während der Schlüsselgenerierung mehrere Zufallszahlen benötigt werden, unter anderem eine Zufallszahl $b \in \{1, \ldots, q-1\}$, die zufällig bezüglich der Gleichverteilung auf $\{1, \ldots, q-1\}$ gewählt werden muss. Für empfohlene Algorithmen zur Berechnung der Zufallszahl b sei auf Abschnitt B.4 verwiesen.

Bemerkung 2.12 Die Effizienz des am Anfang des Abschnitts beschriebenen Verfahrens zur Schlüsselerzeugung kann erhöht werden, indem mehrere Nutzer die Werte (p, q, g) verwenden, so dass sie einmalig vorberechnet werden können. Alternativ ist es auch möglich, veröffentlichte Parameter zu verwenden. Die vorliegende Technische Richtlinie empfiehlt in diesem Fall eine Verwendung der MODP-Gruppen aus [73] oder der ffdhe-Gruppen aus [55], jeweils verbunden mit der Wahl geeigneter Schlüssellängen (MODP-1536 ist also zum Beispiel unabhängig vom vorgesehenen Einsatzzeitraum *nicht* geeignet). In den zuvor genannten Gruppen ist jeweils $q = (p-1)/2$ und $g = 2$. Die Verwendung eines gemeinsamen p durch mehrere Nutzer wird nur dann empfohlen, wenn $\log_2(p) \geq 3000$, da die Berechnung diskreter Logarithmen durch Vorberechnungsattacken vereinfacht werden kann, die nur von dem Parameter p abhängen.

2.3.4. ECIES-Verschlüsselung

Ein *Elliptic Curve Integrated Encryption Scheme* (ECIES) ist ein hybrides Verschlüsselungsverfahren, bei dem die Sicherheit der asymmetrischen Komponente auf dem Diffie-Hellman-Problem in der jeweils verwendeten elliptischen Kurve basiert. Im Folgenden wird eine Version von ECIES beschrieben, die mit den übrigen Empfehlungen der vorliegenden Technischen Richtlinie vereinbar ist, wobei sich in der Verfahrensbeschreibung eng an [1] angelehnt wird.

Die hier wiedergegebene Beschreibung von ECIES ist nahezu vollkommen identisch zur Beschreibung des eng verwandten Verfahrens DLIES in Abschnitt 2.3.3. Der Hauptgrund für eine separate Behandlung beider Verfahren sind potenzielle Schwierigkeiten, die sich aus verschiedenen Notationen ergeben könnten, sowie die für beide Verfahren unterschiedlichen Empfehlungen hinsichtlich sicherer Schlüssellängen. Als normative Referenz wird ECIES-HC in [62] empfohlen. Für einen Überblick zur Standardisierung von ECIES und DLIES sei auf [81] verwiesen.

Ein ECIES benötigt folgende Komponenten:

- Symmetrisches Verschlüsselungsverfahren E_K: Alle in der vorliegenden Richtlinie empfohlenen Kombinationen aus Blockchiffre und Betriebsmodus sind hierfür geeignet.
- Message Authentication Code MAC_{KM}: Es können die in Abschnitt 5.2 empfohlenen Verfahren verwendet werden.
- Schlüsselableitungsfunktion H: H kann beispielsweise eine Hashfunktion sein, falls deren Ausgabe mindestens die Länge des gesamten abzuleitenden symmetrischen Schlüsselmaterials besitzt. Alternativ kann auch die in Abschnitt B.1 empfohlene oder eine der in [62] vorgeschlagenen Schlüsselableitungsfunktionen verwendet werden, um aus den gegebenen Daten abgeleitetes Schlüsselmaterial der gewünschten Länge zu erzeugen.

Darüber hinaus benötigt ein ECIES Schlüsselmaterial, wie im folgenden Abschnitt zur Schlüsselgenerierung beschrieben.

Schlüsselgenerierung

1.) Erzeuge kryptographisch starke EC-Systemparameter (p, a, b, P, q, i), siehe Abschnitt B.3.

2.) Wähle d zufällig und gleichverteilt in $\{1, \ldots, q-1\}$.

3.) Setze $G := d \cdot P$.

Die EC-Systemparameter (p, a, b, P, q, i) bilden zusammen mit G den öffentlichen Schlüssel und d den privaten Schlüssel. Es wird empfohlen, die in Tabelle B.3 angegebenen Kurvenparameter zu verwenden.

Verschlüsselung Gegeben seien eine Nachricht $M \in \{0,1\}^*$ und ein öffentlicher Schlüssel (p, a, b, P, q, i, G), der auf zuverlässige Weise dem berechtigten Empfänger E der Nachricht zugeordnet werden kann. Zur Verschlüsselung wählt der Sender S eine zufällige Zahl $k \in \{1, \ldots, q-1\}$ und berechnet $B := k \cdot P$, $X := k \cdot G$ und daraus $h := H(X)$. Aus h werden genügend Bits entnommen, um einen Schlüssel K für das symmetrische Verschlüsselungsverfahren sowie einen Schlüssel KM für den MAC zu bilden. Aus der Nachricht M berechnet S das Chiffrat $C := E_K(M)$ sowie einen MAC $T := \mathrm{MAC}_{KM}(C)$ und sendet das Tripel (B, C, T) an den Empfänger E.

Entschlüsselung Der Empfänger E erhält (B, C, T) und berechnet $X := d \cdot B$ sowie damit $h := H(X)$, K und KM. Er bestimmt $T' := \mathrm{MAC}_{KM}(C)$ und überprüft, ob $T = T'$ gilt. Falls dies nicht der Fall ist, bricht er den Entschlüsselungsvorgang ab. Ist $T = T'$, dann erhält E durch $M = E_K^{-1}(C)$ die Nachricht zurück.

Schlüssellänge Für die Ordnung q des Basispunktes P sollte mindestens $q \geq 250$ gelten.

Eine notwendige Voraussetzung für die Sicherheit des ECIES-Verfahrens ist es, dass es praktisch unmöglich ist, das Diffie-Hellman-Problem in der von P erzeugten Untergruppe zu lösen. Dies ist bei den in Tabelle B.3 empfohlenen Kurvenparametern nach derzeitigem Kenntnisstand der Fall.

Bemerkung 2.13 Das vorgestellte ECIES-Verfahren stellt einen probabilistischen Algorithmus dar, da im zweiten Schritt der Schlüsselgenerierung eine Zufallszahl $k \in \{1, \ldots, q-1\}$ zufällig bezüglich der Gleichverteilung auf $\{1, \ldots, q-1\}$ gewählt werden muss. Für empfohlene Algorithmen zur Berechnung der Zufallszahl k sei auf Abschnitt B.4 verwiesen.

2.3.5. Diffie-Hellman Schlüsseleinigung

Die Sicherheit dieses Verfahrens beruht auf der angenommenen Schwierigkeit des Diffie-Hellman-Problems in Gruppen (beziehungsweise Untergruppen von) $\mathbb{F}_p^*$, wobei p eine Primzahl ist.

Systemparameter

1.) Wähle zufällig eine Primzahl p.

2.) Wähle ein Element $g \in \mathbb{F}_p^*$ mit $\mathrm{ord}(g)$ prim und $q := \mathrm{ord}(g) \geq 2^{250}$.

Das Tripel (p, g, q) muss vorab authentisch zwischen den Kommunikationspartnern A und B ausgetauscht werden, wobei gleiche Systemparameter prinzipiell durch viele Nutzer verwendet werden. Zur Erzeugung geeigneter Systemparameter, siehe Bemerkung 2.12.

Schlüsselvereinbarung

1.) A wählt gleichverteilt einen Zufallswert $x \in \{1, \ldots, q-1\}$ und sendet $Q_A := g^x$ an B.

2.) B wählt gleichverteilt einen Zufallswert $y \in \{1, \ldots, q-1\}$ und sendet $Q_B := g^y$ an A.

3.) A berechnet $(g^y)^x = g^{xy}$.

4.) B berechnet $(g^x)^y = g^{xy}$.

Auch die Schlüsselvereinbarung muss durch starke Authentisierung abgesichert werden, um Man-in-the-Middle-Angriffe zu verhindern; für weitere Informationen siehe Kapitel 6. Das ausgehandelte gemeinsame Geheimnis ist g^{xy}. Ein Mechanismus für eine nachfolgende Schlüsselableitung aus diesem Geheimnis wird in Abschnitt B.1 empfohlen.

Schlüssellänge Die Länge von p sollte mindestens 3000 Bits betragen.

Bemerkungen zur Implementierung Bei der Implementierung des Diffie-Hellman-Protokolls ist eine Reihe von Implementierungsfehlern weit verbreitet. Auf einige dieser Implementierungsprobleme wird in [112] eingegangen. Es wird empfohlen, insbesondere Abschnitt 7 von [112] zu beachten.

2.3.6. EC Diffie-Hellman Schlüsseleinigung

Die Sicherheit dieses Verfahrens beruht auf der angenommenen Schwierigkeit des Diffie-Hellman-Problems in elliptischen Kurven.

Systemparameter Wähle kryptographisch starke EC-Systemparameter (p, a, b, P, q, i) gemäß Abschnitt B.3. Die damit definierte elliptische Kurve sei mit C und die durch P erzeugte zyklische Untergruppe mit $\mathcal{G}$ bezeichnet. Die Systemparameter (p, a, b, P, q, i) müssen vorab authentisch zwischen den Kommunikationspartnern ausgetauscht werden.

Schlüsselvereinbarung

1.) A wählt gleichverteilt einen Zufallswert $x \in \{1, \ldots, q-1\}$ und sendet $Q_A := x \cdot P$ an B.

2.) B wählt gleichverteilt einen Zufallswert $y \in \{1, \ldots, q-1\}$ und sendet $Q_B := y \cdot P$ an A.

3.) A berechnet $x \cdot Q_B = xy \cdot P$.

4.) B berechnet $y \cdot Q_A = xy \cdot P$.

Auch die Schlüsselvereinbarung muss durch starke Authentisierung abgesichert werden, um Man-in-the-Middle-Angriffe zu verhindern; für weitere Informationen siehe Kapitel 6. Das ausgehandelte Geheimnis ist $xy \cdot P$. Ein Mechanismus für eine nachfolgende Schlüsselableitung aus diesem Geheimnis wird in Abschnitt B.1 empfohlen.

Soweit möglich, wird empfohlen, auf beiden Seiten der Schlüsselvereinbarung zu überpüfen, ob die Punkte Q_A und Q_B protokollgerecht gewählt wurden und das Protokoll andernfalls abzubrechen. Bei korrekter Ausführung des obigen Protokolls sollten $Q_A \in \mathcal{G}$, $Q_B \in \mathcal{G}$, $Q_A \neq \mathcal{O}$ und $Q_B \neq \mathcal{O}$ gelten. Im Rahmen der Prüfung $Q_A, Q_B \in \mathcal{G}$ sollte explizit auch überprüft werden, ob $Q_A, Q_B \in C$. Weitere Hinweise finden sich in Abschnitt 4.3.2.1 von [27].

Schlüssellänge Die Länge von q sollte mindestens 250 Bits betragen.

Bemerkungen zur Implementierung Bei der Implementierung des Diffie-Hellman-Schlüsselaustausches ist eine Reihe von Implementierungsfehlern weit verbreitet. Auf einige dieser Implementierungsprobleme wird in [112] eingegangen. Es wird empfohlen, insbesondere Abschnitt 7 von [112] zu beachten, ebenso sind die Hinweise in Abschnitt 4.3 von [27] und die AIS46 [40] zu berücksichtigen.

2.4. Quantensichere asymmetrische Verfahren

Die sich derzeit in der Standardisierung befindlichen quantensicheren asymmetrischen Verfahren zur Schlüsseleinigung basieren entweder auf Gittern oder auf binären Codes. Die Verfahren wurden wegen des üblichen Einsatzzwecks, also der Verteilung oder Aushandlung von Schlüsselmaterial, direkt als „Key Encapsulation Mechanism“ (kurz KEM, siehe Schlüsselkapselungsverfahren (Key Encapsulation Mechanism, KEM)) entworfen.

2.4.1. FrodoKEM Schlüsseleinigung

Bei FrodoKEM handelt es sich um ein gitterbasiertes Verfahren, das auf dem Learning with Errors-Problem (kurz LWE, siehe Learning with Errors (LWE) Problem) beruht. Das NIST hat entschieden, FrodoKEM nicht zu standardisieren, da das ebenfalls auf Learning with Errors (LWE) Problem beruhende ML-KEM (siehe Abschnitt 2.4.3) effizienter ist. Da FrodoKEM im Gegensatz zu ML-KEM auf unstrukturierten Gittern beruht, wird es jedoch als die konservativere Wahl angesehen. FrodoKEM wird aktuell bei ISO standardisiert; zukünftige Versionen dieser Technischen Richtlinie werden den entsprechenden Standard referenzieren.

FrodoKEM mit den folgenden Parametern wird als kryptographisch geeignet eingeschätzt, um vertrauliche Informationen auf dem in dieser Technischen Richtlinie angestrebten Sicherheitsniveau langfristig zu schützen:

FrodoKEM-976 und FrodoKEM-1344, siehe [3, Abschnitt 2.5].

Tabelle 2.5: Empfohlene Parameter für FrodoKEM.

2.4.2. Classic McEliece Schlüsseleinigung

Bei Classic McEliece handelt es sich um einen codebasierten KEM. Das zugrundeliegende Verfahren, das mit binären Goppa-Codes instanziiert wird, ist in etwa so alt wie das RSA-Verfahren und gilt daher als konservativ und sehr gründlich untersucht. Ein Nachteil sind die vergleichsweise großen öffentlichen Schlüssel, andererseits hat Classic McEliece sehr kleine Chiffrate.

Classic McEliece wird aktuell bei ISO standardisiert und möglicherweise auch durch NIST nach der 4. Runde des Standardisierungsprozesses. Zukünftige Versionen dieser Technischen Richtlinie werden entsprechende Standards referenzieren.

Classic McEliece mit den folgenden Parametern wird als kryptographisch geeignet eingeschätzt, um vertrauliche Informationen auf dem in dieser Technischen Richtlinie angestrebten Sicherheitsniveau langfristig zu schützen:

- mceliece460896, mceliece6688128 und mceliece8192128, siehe [2, Abschnitt 7],
- mceliece460896f, mceliece6688128f und mceliece8192128f (schnellere Varianten), siehe [2, Abschnitt 7].

Tabelle 2.6: Empfohlene Parameter für ClassicMcEliece-KEM.

2.4.3. ML-KEM Schlüsseleinigung

Bei ML-KEM handelt es sich um ein gitterbasiertes Schlüsselkapselungsverfahren (Key Encapsulation Mechanism, KEM), dessen Sicherheit auf dem „Module Learning with Errors (MLWE)“-Problem beruht.

Schlüsselgenerierung, Enkapsulierung und Dekapsulierung sind in [104] beschrieben. ML-KEM mit den Parametersätzen entsprechend NIST Security Strength Categories 3 und 5 (siehe Tabelle 2.7) aus [104] wird als kryptografisch geeignet eingeschätzt, um vertrauliche Informationen auf dem in dieser Technischen Richtlinie angestrebten Sicherheitsniveau langfristig zu schützen.

- ML-KEM-768, siehe [104],
- ML-KEM-1024, siehe [104].

Tabelle 2.7: Empfohlene Parameter für ML-KEM.

3. Symmetrische Verschlüsselung und Schlüsseleinigung

In diesem Kapitel werden symmetrische Verschlüsselungsverfahren behandelt, das heißt Verfahren, bei denen Ver- und Entschlüsselungsschlüssel gleich sind – im Gegensatz zu asymmetrischen Verfahren, bei denen der private aus dem öffentlichen Schlüssel ohne zusätzliche Informationen praktisch nicht berechnet werden kann. Für asymmetrische Verschlüsselungsverfahren, die in der Praxis in der Regel lediglich als Schlüsseltransportverfahren eingesetzt werden, sei auf Kapitel 2 verwiesen.

Symmetrische Verschlüsselungsverfahren dienen der Gewährleistung der Vertraulichkeit von Daten, die zum Beispiel über einen öffentlichen Kanal wie Telefon oder Internet ausgetauscht werden. Authentizität beziehungsweise Integrität der Daten wird dadurch in der Regel nicht automatisch gewährleistet, für einen Integritätsschutz siehe Kapitel 5 und Abschnitt A.1. Verfahren, die zusätzlich zur Verschlüsselung eine kryptographisch sichere Datenauthentisierung liefern, werden in Bemerkung 3.1 addressiert. Auch in Fällen, in denen auf den ersten Blick der Schutz der Vertraulichkeit übermittelter Daten das dominierende oder sogar das einzige Sicherheitsziel zu sein scheint, kann eine Vernachlässigung integritätssichernder Mechanismen leicht zu Schwächen im kryptographischen Gesamtsystem führen, die das System dann auch für Angriffe auf die Vertraulichkeit anfällig machen. Insbesondere können auf solche Weise Schwächen durch bestimmte Arten aktiver Seitenkanalangriffe entstehen, für ein Beispiel siehe etwa [118].

Die Entwicklung fehlertoleranter Quantencomputer hat wesentlich weniger gravierende Auswirkungen auf die Sicherheit symmetrischer Verfahren als auf die Sicherheit von asymmetrischen Verfahren. Der Grover-Algorithmus [56] könnte theoretisch die Durchsuchung des Schlüsselraums symmetrischer Verfahren quadratisch beschleunigen. Ob eine Beschleunigung im Vergleich zur klassischen Durchsuchung des Schlüsselraums auch praktisch erreicht werden kann, ist Gegenstand aktueller Forschung und nicht abschließend geklärt (siehe z.B. [71]). Insbesondere bei Anwendungen mit hohem oder langfristigem Schutzbedarf oder langlebigen Systemen ist es dennoch ratsam, bei den im Folgenden empfohlenen symmetrischen Verschlüsselungsverfahren eine Schlüssellänge von 256 Bits zu nutzen.

Ferner werden in diesem Kapitel auch symmetrische Schlüsseleinigungs- und Transportverfahren sowie Key-Update- und Key-Wrapping-Mechanismen vorgestellt.

3.1. Blockchiffren

Allgemeine Empfehlungen Eine Blockchiffre ist ein Algorithmus, der einen Klartext fester Bitlänge (zum Beispiel 128 Bits) mittels eines Schlüssels zu einem Chiffretext gleicher Bitlänge verschlüsselt. Diese Bitlänge wird auch als *Blockgröße* der Chiffre bezeichnet. Für die Verschlüsselung von Klartexten anderer Länge werden Blockchiffren in verschiedenen Betriebsarten angewendet, siehe Abschnitt 3.1.1. Für neue kryptographische Anwendungen sollten nur noch Blockchiffren eingesetzt werden, deren Blockgröße mindestens 128 Bits beträgt.

Folgende Blockchiffren werden zur Verwendung in neuen kryptographischen Systemen empfohlen:

AES-128, AES-192, AES-256, siehe [88].

Tabelle 3.1: Empfohlene Blockchiffren.

In Version 1.0 der vorliegenden Technischen Richtlinie wurden neben dem AES noch weitere Blockchiffren empfohlen, deren Sicherheit seit dem Ende des AES-Wettbewerbs jedoch deutlich weniger intensiv untersucht wurde als die des Rijndael-Algorithmus, welcher als Sieger und somit künftiger AES aus dem Wettbewerb hervorging. Dies gilt sowohl für klassische kryptoanalytische Angriffe als auch für andere Sicherheitsaspekte, zum Beispiel die Seitenkanalresistenz konkreter Implementierungen. Aus diesem Grund wird in der vorliegenden Version dieser Technischen Richtlinie auf eine Empfehlung weiterer Blockchiffren neben dem AES verzichtet.

Related-Key-Attacken und AES Bei Related-Key-Attacken wird davon ausgegangen, dass der Angreifer Zugriff auf Verschlüsselungen oder Entschlüsselungen bekannter oder gewählter Klartexte oder Chiffrate unter verschiedenen Schlüsseln hat, die zueinander in einer dem Angreifer bekannten Beziehung stehen (also zum Beispiel sich genau in einer Bitposition des Schlüssels unterscheiden). Bestimmte Angriffe dieser Art gegen rundenreduzierte Versionen des AES-256 [13] und gegen unmodifizierte Versionen des AES-192 sowie AES-256 [14] stellen bislang die einzigen bekannten kryptoanalytischen Techniken dar, denen gegenüber AES ein wesentlich schlechteres Verhalten zeigt als eine ideale Chiffre mit entsprechender Schlüssellänge und Blockgröße.

Zum gegenwärtigen Zeitpunkt haben diese Erkenntnisse zur Sicherheit von AES unter spezifischen Typen von Related-Key-Attacken keine Auswirkungen auf die in dieser Technischen Richtlinie ausgesprochenen Empfehlungen. Insbesondere ein Related-Key Boomerang-Attacken auf AES-256 aus [14] mit Rechenzeit- und Datenkomplexität von $2^{99.5}$ ist aufgrund der technischen Voraussetzungen von Related-Key Boomerang-Attacken nicht als Verletzung des in dieser Technischen Richtlinie mittelfristig angestrebten Sicherheitsniveaus von 120 Bits zu betrachten. Die besten bekannten Angriffe gegen AES, die keine Related-Keys benötigen, erzielen nur einen geringen Vorteil gegenüber generischen Angriffen [18].

3.1.1. Betriebsarten

Wie bereits in Abschnitt 3.1 erwähnt, liefert eine Blockchiffre lediglich einen Mechanismus zur Verschlüsselung von Klartexten einer einzigen festen Länge. Um Klartexte beliebiger Länge zu verschlüsseln, muss aus der Blockchiffre mittels einer geeigneten *Betriebsart* ein Verschlüsselungsverfahren für Klartexte (annähernd) beliebiger Länge konstruiert werden. Ein weiterer Effekt einer kryptographisch starken Betriebsart ist, dass das resultierende Verschlüsselungsverfahren in mancher Hinsicht stärker sein wird als die zugrundeliegende Blockchiffre, zum Beispiel wenn die Betriebsart den Verschlüsselungsvorgang randomisiert und dadurch das Wiedererkennen einer mehrfachen Verschlüsselung gleicher Klartexte erschwert.

Verschiedene Betriebsarten für Blockchiffren können dabei zunächst nur mit Klartexten umgehen, deren Länge ein Vielfaches der Blockgröße ist. In diesem Fall ist der letzte Block eines gegebenen Klartextes möglicherweise zu kurz und muss entsprechend aufgefüllt werden. Eine Formatierung durch das Auffüllen dieses letzten Blocks zu der geforderten Größe wird auch als *Padding* bezeichnet. In Abschnitt 3.1.3 werden geeignete Padding-Verfahren vorgestellt. Unter den empfohlenen Betriebsarten für Blockchiffren benötigt aber nur der CBC-Modus einen Padding-Schritt.

Die einfachste Möglichkeit, einen Klartext zu verschlüsseln, dessen Länge bereits ein Vielfaches der Blockgröße ist, besteht darin, jeden Klartextblock mit dem gleichen Schlüssel zu verschlüsseln; diese Betriebsart heißt auch Electronic Code Book (ECB). Die Verwendung des ECB-Modus führt

jedoch dazu, dass gleiche Klartextblöcke zu gleichen Chiffretextblöcken verschlüsselt werden. Der Chiffretext liefert damit zumindest Informationen über die Struktur des Klartextes und ermöglicht bei niedriger Entropie pro Block des Klartextes gegebenenfalls eine Rekonstruktion von Teilen des Klartextes durch Häufigkeitsanalysen. Aus diesem Grund sollte der n-te Chiffreblock nicht nur vom n-ten Klartextblock und dem eingesetzten Schlüssel abhängen, sondern von einem weiteren Wert, wie beispielsweise dem $(n-1)$-ten Chiffretextblock oder einem Zähler (auch Counter genannt).

Dies ist bei den in Tabelle 3.2 empfohlenen Betriebsarten, die für die in Tabelle 3.1 aufgeführten Blockchiffren geeignet sind, der Fall.

- Counter with Cipher Block Chaining Message Authentication (CCM), siehe [89],
- Galois/Counter Mode (GCM), siehe [90],
- Galois/Counter Mode with Synthetic Initialization Vector für AES (AES-GCM-SIV), siehe [57],
- Cipher Block Chaining (CBC), siehe [87],
- Counter Mode (CTR), siehe [87].

Tabelle 3.2: Empfohlene Betriebsarten für Blockchiffren.

Bemerkung 3.1 Sowohl der GCM und der AES-GCM-SIV-Modus als auch der CCM-Modus liefern bei ausreichender Tag-Länge zusätzlich zur Verschlüsselung eine kryptographisch sichere Datenauthentisierung. Derartige Verfahren werden auch als Authenticated Encryption with Associated Data (AEAD)) bezeichnet.

Für die beiden anderen Betriebsmodi wird generell empfohlen, separate Mechanismen zur Datenauthentisierung im Gesamtsystem vorzusehen. Idealerweise sollte für nicht authentisierte verschlüsselte Daten keine Entschlüsselung oder sonstige weitere Verarbeitung erfolgen. Wenn nicht authentisierte verschlüsselte Daten entschlüsselt und weiter verarbeitet werden, dann ergeben sich erhöhte Restrisiken im Hinblick auf die Ausnutzung von Fehlerorakeln, siehe zum Beispiel [118].

3.1.2. Betriebsbedingungen

Für die in Abschnitt 3.1.1 aufgeführten Betriebsarten werden Initialisierungsvektoren benötigt, zudem müssen für einen sicheren Betrieb bestimmte weitere Randbedingungen eingehalten werden, die im Folgenden zusammengefasst sind:

Für CCM:

- Die Länge der Authentisierungstags muss geeignet gewählt werden. Für allgemeine kryptographische Anwendungen wird eine Tag-Länge von ≥ 96 Bits empfohlen. Allgemein können Angreifer Chiffrate oder authentisierte Daten bei Verwendung der Tag-Länge t im CCM-Modus mit einer Erfolgswahrscheinlichkeit von $\approx 2^{-t}$ pro Versuch unbemerkt verändern. Bei Verwendung geringerer Tag-Längen als der hier empfohlenen müssen die damit einhergehenden Restrisiken sorgfältig durch einen Experten untersucht werden.

Für GCM:

- Für die GCM-Initialisierungsvektoren wird in [90] eine Bitlänge von 96 Bits empfohlen. Dieser Empfehlung schließt sich die vorliegende Technische Richtlinie an, insbesondere mit Verweis

auf die Resultate aus [70].[1] In [90] wird gefordert, dass die Wahrscheinlichkeit einer Wiederholung von Initialisierungsvektoren unter einem gegebenen Schlüssel $\leq 2^{-32}$ sein soll. Daraus ergibt sich ein Schlüsselwechselintervall von höchstens 2^{32} Aufrufen der authentisierten Verschlüsselungsfunktion. Bei einer deterministischen Erzeugung der Initialisierungsvektoren muss nachgewiesen werden, dass eine Wiederholung von Initialisierungsvektoren über die gesamte Lebensdauer eines Schlüssels hinweg ausgeschlossen ist.

- Für allgemeine kryptographische Anwendungen sollte GCM mit einer Länge der GCM-Prüfsummen von mindestens 96 Bits verwendet werden. Für spezielle Anwendungen können nach Rücksprache mit Experten auch kürzere Prüfsummen genutzt werden. In diesem Fall müssen die Richtlinien zur Anzahl der erlaubten Aufrufe der Authentisierungsfunktion mit einem gemeinsamen Schlüssel aus [90] strikt eingehalten werden.

Für AES-GCM-SIV:

- Für die AES-GCM-SIV-Initialisierungsvektoren wird in [57] eine Bitlänge von 96 Bits empfohlen, zudem sollten sie zufällig erzeugt werden. Diesen Empfehlungen schließt sich die vorliegende Technische Richtlinie an.
- AES-GCM-SIV ist für AES-128 und AES-256 definiert, eine Schlüssellänge von 192 Bit sollte daher nicht verwendet werden.

Für CCM, GCM und CTR-Modus:

- Initialisierungsvektoren dürfen sich innerhalb einer Schlüsselwechselperiode nicht wiederholen. Genauer dürfen in dem gesamten Mechanismus keine zwei AES-Verschlüsselungen (das heißt Anwendungen der zugrundeliegenden AES-Blockchiffre) mit gleichen Eingabewerten (Schlüssel, Nachricht) durchgeführt werden. Eine Nichtbeachtung dieser Bedingung führt zu einem potenziell vollständigen Verlust der Vertraulichkeit für die betroffenen Klartextblöcke, im Falle des GCM zusätzlich auch der Integrität. Falls die Wiederholung eines Initialisierungsvektors nicht ausgeschlossen werden kann, bietet sich die Verwendung des AES-GCM-SIV als AEAD-Modus an, da in diesem Fall Vertraulichkeit und Integrität einer Nachricht auch bei einer Wiederholung des Initialisierungsvektors sichergestellt sind.

Für CBC:

- Es dürfen nur unvorhersagbare Initialisierungsvektoren verwendet werden. Diesbezüglich werden in Abschnitt B.2 verschiedene Verfahren empfohlen.

Bei Anwendungen, bei denen die hier angegebenen Anforderungen an die Initialisierungsvektoren nicht erfüllt werden können, wird dringend zur Hinzuziehung eines Experten geraten.

3.1.3. Paddingverfahren

Wie bereits in Abschnitt 3.1.1 erläutert, verlangt der CBC-Modus einen zusätzlichen Padding-Schritt: Es kann bei der Partitionierung eines zu verschlüsselnden Klartextes geschehen, dass der letzte Klartextblock kleiner als die Blockgröße der eingesetzten Chiffre ist.

In dieser Technischen Richtlinie werden folgende Paddingverfahren empfohlen:

[1] In [70] wird auf Fehler in bis dahin akzeptierten Sicherheitsbeweisen zum Galois/Counter Modus hingewiesen und es wird eine korrigierte Analyse der Sicherheit von GCM vorgestellt. In dieser korrigierten Analyse erweist sich eine IV-Länge von exakt 96 Bits als vorteilhaft.

- ISO-Padding, siehe [64], padding method 2 und [87, Appendix A],
- Padding gemäß [59, Abschnitt 6.3],
- ESP-Padding, siehe [72, Abschnitt 2.4].

Tabelle 3.3: Empfohlene Paddingverfahren für Blockchiffren.

Bemerkung 3.2 Beim CBC-Mode ist darauf zu achten, dass ein Angreifer nicht anhand von Fehlermeldungen oder anderen Seitenkanälen erfahren kann, ob das Padding eines eingespielten Datenpakets korrekt war [118]. Allgemeiner gilt folgendes: Wenn ein Angreifer bei Verschlüsselungsverfahren Änderungen am Chiffrat durchführen kann, die zu kontrollierten Änderungen am Klartext führen, darf dem Angreifer keine Seitenkanalinformation zur Verfügung stehen, die Aufschluss darüber liefert, ob ein gegebenes Chiffrat zu einem gültigen Klartext korrespondiert oder ob es von ungültigem Format ist.

3.2. Stromchiffren

Bei Stromchiffren wird aus einem Schlüssel und einem Initialisierungsvektor zunächst ein Schlüsselstrom generiert, das heißt eine pseudozufällige Folge von Bits, die dann auf die zu verschlüsselnde Nachricht bitweise XOR-addiert wird. Zurzeit werden keine dedizierten Stromchiffren empfohlen, allerdings kann AES im Counter Modus (AES-CTR) als Stromchiffre aufgefasst werden. Wird eine Stromchiffre eingesetzt, wird dringend empfohlen, die Integrität der übertragenen Information durch separate kryptographische Mechanismen zu schützen. Ein Angreifer kann in Abwesenheit solcher Mechanismen bitgenaue Änderungen am Klartext vornehmen.

3.3. Symmetrische Schlüsseleinigungsverfahren, Key-Wrapping und Key-Update

Neben Schlüsseleinigungsverfahren sind auch kryptographische Verfahren zum Schutz von Schlüsseln hinsichtlich Vertraulichkeit und Integrität bei Transport und Speicherung von praktischer Relevanz. Solche Verfahren werden auch als Key-Wrapping-Verfahren bezeichnet. Schlüsselmaterial kann von einer vertrauenswürdigen Drittpartei oder einem der Kommunitkationspartner erzeugt werden. Im letzteren Fall wird empfohlen, dass alle Teilnehmer nur jeweils selbst erzeugte Schlüssel zur Übertragung eigener sensibler Daten nutzen. Die Empfänger haben an dieser Stelle keine Kontrolle über die verteilten Sitzungsschlüssel. Auch hier muss die Authentizität der Empfänger sowie die Sicherheit des Übertragungskanals zusätzlich zum sicheren Schlüsseleinigungs- beziehungsweise Key-Wrapping-Verfahren durch externe Mittel sichergestellt werden.

Schließlich werden in diesem Abschnitt auch Key-Update-Verfahren behandelt. Hier teilen zwei Parteien bereits ein gemeinsames Geheimnis und leiten daraus am Ende einer Schlüsselperiode einen neuen Schlüssel ab. Dies kann entweder durch Ableitung neuer Sitzungsschlüssel aus einem dauerhaften Masterschlüssel oder auch durch eine Update-Prozedur, die aus dem aktuellen Schlüssel und gegebenenfalls weiteren Daten einen neuen Schlüssel generiert, erreicht werden. Bei der Festlegung der Lebensdauer von Schlüsselmaterial müssen verschiedene Faktoren berücksichtigt werden, darunter die Art des Schlüssels, die Einsatzumgebung oder die Sensitivität der zu schützenden Daten. Weitere Informationen zu diesem Thema finden sich beispielsweise in [101].

Bemerkung 3.3 (Asymmetrische versus symmetrische Schlüsseleinigungsverfahren) Mit asymmetrischen Schlüsseleinigungsverfahren können Sicherheitseigenschaften erfüllt werden, die allein unter Verwendung symmetrischer Kryptographie nicht realisierbar sind. Zum Beispiel erfüllen die beiden empfohlenen klassischen asymmetrischen Schlüsseleinigungsverfahren Diffie-Hellman und EC Diffie-Hellmann (siehe Abschnitt 2.3.5 beziehungsweise 2.3.6) die Eigenschaft, dass ein Angreifer, der alle gegebenenfalls vorhandenen Langzeitgeheimnisse beider Kommunikationsteilnehmer kennt[2], dennoch nicht den während einer unkompromittierten Protokollausführung ausgehandelten Schlüssel ermitteln kann, falls er das dem verwendeten asymmetrischen Verfahren zugrundeliegende mathematische Problem nicht effizient lösen kann. Im Vergleich dazu kann in symmetrischen Schlüsseleinigungsverfahren höchstens das Sicherheitsziel *Post-Compromise Security* erreicht werden, das heißt, dass ein Angreifer, der alle Langzeitgeheimnisse beider Teilnehmer kennt, die Ergebnisse *vergangener* korrekt durchgeführter Schlüsseleinigungen nicht ermitteln kann.[3]

Schlüsseleinigung Wenn die Existenz eines gemeinsamen langfristigen Geheimnisses vorausgesetzt werden kann, lassen sich auch Schlüsseleinigungsverfahren auf Basis ausschließich symmetrischer Verfahren realisieren. Key Establishment Mechanism 5 aus [65] stellt ein geeignetes Verfahren dar. Falls eine implizite Schlüsselbestätigung durch Besitz gleicher Sitzungsschlüssel für die jeweils gegebene kryptographische Anwendung nicht ausreichend ist, wird empfohlen, dieses Protokoll noch um einen Schritt zur Schlüsselbestätigung zu erweitern. Als Key Derivation Function sollte dabei der in Abschnitt B.1 empfohlene Mechanismus verwendet werden.

Schutz von Schlüsselmaterial Schlüsselmaterial wie beispielsweise Langzeit- oder Sitzungsschlüssel muss beim Transport oder der Speicherung hinsichtlich Vertraulichkeit und Integrität geschützt werden. Symmetrische Verfahren, die das leisten, werden auch als Key-Wrapping-Verfahren bezeichnet. Der Key Encryption Key (KEK), der zum Schutz des Schlüsselmaterials verwendet wird, sollte dabei mindestens genauso hohen Sicherheitsanforderungen genügen wie das Schlüsselmaterial selbst. Ferner darf ein Schlüssel nicht mehrfach mit demselben KEK gewrapped werden und die Anzahl der erfolglosen Verifizierungsversuche für einen Schlüssel sollte limitiert sein.

Grundsätzlich können alle der zuvor empfohlenen symmetrischen Verschlüsselungsverfahren in einem empfohlenen AEAD-Mode als Key-Wrapping-Verfahren verwendet werden. Auch eignet sich die Kombination eines der in Abschnitt 3.1 empfohlenen Verschlüsselungsverfahren mit einem MAC aus Abschnitt 5.2 (im Encrypt-then-MAC-Modus) zum Key-Wrapping.

Bei besonders hohen Sicherheitsanforderungen und wenn die Länge des Schlüsselmaterials die Blockbreite der zugrunde liegenden Blockchiffre überschreitet, wird die Verwendung der speziellen Key-Wrapping-Verfahren KW und KWP empfohlen, siehe [92].

Key Update In manchen Situationen kann es erforderlich sein, die in einem kryptographischen System genutzten Schlüssel synchron bei allen Beteiligten auszutauschen, ohne dass ein erneuter Schlüsselaustausch oder weitere Kommunikation stattfindet. In diesem Fall können Key-Update-Mechanismen zum Einsatz kommen. Unter der Annahme, dass der Masterschlüssel K_t eines Kryptosystems zum Zeitpunkt t über ein solches Verfahren ersetzt werden soll, empfehlen wir

$$K_{t+1} \coloneqq \mathrm{KDF}(s, \text{Label}, \text{Context}, L, K_t)$$

[2] Gemeint sind hier in erster Linie die langfristigen Geheimnisse, die zur Absicherung der Verbindung gegen Man-in-the-Middle-Attacken verwendet werden müssen.

[3] *Merkle Puzzles* stellen in diesem Zusammenhang insofern eine Ausnahme dar, als dass es sich dabei um ein Schlüsseleinigungsverfahren mit öffentlichen Schlüsseln unter ausschließlicher Nutzung symmetrischer Primitive handelt [85]. Dieses Verfahren ist aber nur von akademischer Bedeutung.

zu setzen. Hierbei bezeichnet KDF eine zweischrittige kryptographische Schlüsselableitungsfunktion nach [100, Abschnitt 5], und s ist der dabei im Extraktionsschritt genutzte Salt-Wert. Die Parameter Label und Context gehen in dem in [100] vorgesehenen Expansionsschritt gemäß [102] ein. Dabei ist Label ein String, der die Funktion des abzuleitenden Schlüssels kenntlich macht und Context enthält Informationen zum weiteren Protokollkontext. L bezeichnet die Länge des abzuleitenden Schlüssels K_{t+1} und fließt ebenfalls in den Expansionsschritt ein.

Bei diesem Verfahren ist unbedingt darauf zu achten, dass bei einer eventuellen Ableitung weiteren Schlüsselmaterials aus K_t andere Ableitungsparameter verwendet werden als bei der Ableitung von K_{t+1}. Es wird empfohlen, dies durch Verwendung geeigneter Label-Werte zu erzwingen und ferner, in Label oder Context mindestens auch die Kryptoperiode t zu codieren. Als zusätzliche Maßnahme kann es zudem sinnvoll sein, für jede Schlüsselableitung einen neuen Salt-Wert zu verwenden. Es wird empfohlen, K_t unmittelbar nach Berechnung von K_{t+1} ebenso wie alle Zwischenergebnisse der Berechnung sicher zu löschen. Für weitere Empfehlungen zur Implementierung dieser Verfahren wird auf [100, 102] verwiesen.

4. Hashfunktionen

Hashfunktionen $H\colon \{0,1\}^* \to \{0,1\}^n$ spielen in vielen kryptographischen Verfahren eine große Rolle, beispielsweise bei der Ableitung kryptographischer Schlüssel oder bei der Datenauthentisierung. Sie bilden einen Bitstring $m \in \{0,1\}^*$ beliebiger Länge[1] auf einen Bitstring $h \in \{0,1\}^n$ fester Länge $n \in \mathbb{N}$ ab.

Hashfunktionen, die in kryptographischen Verfahren eingesetzt werden, müssen je nach Anwendung die folgenden drei Bedingungen erfüllen:

Einweg-Eigenschaft: Es ist praktisch nicht möglich, für gegebenes $h \in \{0,1\}^n$ einen Wert $m \in \{0,1\}^*$ mit $H(m) = h$ zu finden.

2nd-Preimage-Eigenschaft: Es ist praktisch nicht möglich, für gegebenes $m \in \{0,1\}^*$ einen Wert $m' \in \{0,1\}^* \backslash \{m\}$ mit $H(m) = H(m')$ zu finden.

Kollisionsresistenz: Es ist praktisch nicht möglich, zwei Werte $m, m' \in \{0,1\}^*$ mit $m \neq m'$ und $H(m) = H(m')$ zu finden.

Eine Hashfunktion H, die sämtliche der obigen Bedingungen erfüllt, heißt *kryptographisch stark*.

Mathematisch präziser lassen sich diese drei Begriffe jeweils durch einen Vergleich der besten bekannten Angriffe auf diese Eigenschaften mit optimalen generischen Angriffen fassen. Die Länge des Hash-Outputs ist ein Sicherheitsparameter von zentraler Bedeutung, da er den Aufwand generischer Angriffe bestimmt. Für das in dieser Technischen Richtlinie minimal geforderte Sicherheitsniveau von 120 Bits muss wegen des Geburtstagsparadoxons für eine Hashfunktion $H\colon \{0,1\}^* \to \{0,1\}^n$ mindestens $n \geq 240$ gelten. Eine Fallunterscheidung je nach Einsatzzeitraum des Verfahrens ist an dieser Stelle nicht nötig, da die in dieser Technischen Richtlinie empfohlenen Hashverfahren alle bereits eine Digest-Länge von ≥ 256 Bits aufweisen.

Bemerkung 4.1 Es gibt kryptographische Anwendungen von Hashfunktionen, in denen nicht alle drei angegebenen Eigenschaften einer starken Hashfunktion benötigt werden. Umgekehrt gibt es weitere relevante kryptographische Anforderungen an Hashfunktionen, die sich nicht aus den drei angegebenen Eigenschaften ergeben. Ein Beispiel ist die Eigenschaft der *Zero Finder Resistance* (Resistenz gegen Suche nach Urbildern des Hashwertes Null, [21]), die im Zusammenhang mit ECDSA-Signaturen von Bedeutung ist. Sämtliche der in der vorliegenden Richtlinie empfohlenen Hashverfahren haben im Hinblick auf die in dieser Richtlinie empfohlenen kryptographischen Verfahren, in denen sie eingesetzt werden, keine bekannten kryptographischen Schwächen.

Die Entwicklung fehlertoleranter Quantencomputer hat wesentlich weniger gravierende Auswirkungen auf die Sicherheit von Hashfunktionen als auf die Sicherheit von asymmetrischen Verfahren. Der Grover-Algorithmus könnte theoretisch die Suche nach Urbildern quadratisch beschleunigen. Außerdem sind Quantenalgorithmen theoretisch in der Lage, Kollisionen einer Hashfunktion mit einer Ausgabelänge von n Bits mit $2^{\frac{n}{3}}$ Aufrufen der Hashfunktion zu finden (siehe [20, 120]). Ob eine Beschleunigung im Vergleich zur klassischen Urbild- und Kollisionssuche auch praktisch erreicht werden kann, ist Gegenstand aktueller Forschung und nicht abschließend geklärt [11]. Insbesondere bei Anwendungen mit hohem oder langfristigem Schutzbedarf oder langlebigen Systemen

[1] Spezifikationen realer Hashfunktionen beinhalten in der Regel eine Längenbegrenzung, die aber so hoch liegt, dass sie von realen Eingabestrings nicht überschritten wird.

ist es dennoch ratsam, die im Folgenden empfohlenen Hashfunktionen mit einer Ausgabelänge von mindestens 384 Bits zu nutzen.

Nach heutigem Kenntnisstand gelten die folgenden Hashfunktionen als kryptographisch stark und sind damit für alle in dieser Technischen Richtlinie genannten Verfahren einsetzbar:

- SHA-256, SHA-512/256, SHA-384 und SHA-512, siehe [94].
- SHA3-256, SHA3-384, SHA3-512, siehe [95].

Tabelle 4.1: Empfohlene Hashfunktionen.

Bemerkung 4.2 (i) Für die Hashfunktion SHA1 wurden erstmals in [117] Beispiele von Hashkollisionen angegeben. Aufgrund erheblicher kryptoanalytischer Fortschritte [77, 78] konnten die Kosten für die Berechnung einer solchen Kollision inzwischen auf größenordnungsmäßig mehrere tausend Euro reduziert werden, und sogar Angriffe, in denen Nachrichten mit zwei willkürlichen Anfangsstücken zur Kollision gebracht werden sollen, sind in der Reichweite akademischer Angreifer. SHA1 sollte daher niemals als sichere kryptographische Hashfunktion verwendet werden. Eine Verwendung in anderen kryptographischen Anwendungen, etwa im Rahmen einer HMAC-Konstruktion, wird dadurch nicht ausgeschlossen, sollte aber ebenfalls vermieden werden.

(ii) Bereits eine einzige Kollision einer Hashfunktion kann zu einer Unsicherheit bei Signaturverfahren führen, siehe zum Beispiel [80] und [54].

(iii) Sowohl die Hashfunktionen der SHA2-Familie als auch die der SHA3-Familie werden als kryptographisch stark eingeschätzt. Im Hinblick auf klassische Angriffe auf Kollisionsresistenz und Einwegeigenschaften ist nach derzeitigem Kenntnisstand kein praktisch relevanter Unterschied zwischen den beiden Funktionenfamilien bekannt. In anderen Anwendungsszenarien gibt es hingegen Unterschiede, zum Beispiel sind die Funktionen der SHA3-Familie resistent gegen Length-Extension-Attacken, siehe auch [9].

5. Datenauthentisierung

Im Kontext dieser technischen Richtlinie werden unter dem Begriff Datenauthentisierung kryptographische Verfahren verstanden, die sicherstellen, dass übersandte oder gespeicherte Daten nicht durch unbefugte Personen und/oder Anwendungen verändert wurden. Dazu benutzt ein Beweisender (üblicherweise der Sender der Daten) einen kryptographischen Schlüssel zur Berechnung der Prüfsumme der zu authentisierenden Daten. Ein Prüfer (üblicherweise der Empfänger der Daten) überprüft dann, ob die empfangene Prüfsumme der zu authentisierenden Daten mit der übereinstimmt, die er bei Unverfälschtheit der Daten und Verwendung des richtigen Schlüssels erwarten würde.

Bei der Datenauthentisierung werden symmetrische und asymmetrische Verfahren unterschieden. Bei symmetrischen Verfahren verwenden Beweisender und Prüfer den gleichen kryptographischen Schlüssel. Ein Dritter kann in diesem Fall nicht überprüfen, wer die Prüfsumme berechnet hat oder ob sie richtig berechnet wurde. Bei asymmetrischen Verfahren wird der private Schlüssel für die Berechnung der Prüfsumme benutzt und mit dem assoziierten öffentlichen Schlüssel überprüft. Dies wird in der Regel durch digitale Signaturen (siehe Abschnitt 5.3) umgesetzt.

In symmetrischen Verfahren zur Datenauthentisierung kann der Prüfer einer Nachricht somit grundsätzlich auch gefälschte Nachrichten erzeugen. Damit eignen sich solche Verfahren nur dann, wenn das zusätzliche Kompromittierungsrisiko tragbar ist, welches aus der Verteilung des symmetrischen Schlüssels und seiner Verfügbarkeit für (mindestens) zwei Parteien entsteht. Zudem muss es unkritisch sein, wenn die prüfende Partei eine Nachricht fälscht. Ist eine dieser Bedingungen nicht erfüllt, sind symmetrische Datenauthentisierungsverfahren ungeeignet und es müssen digitale Signaturen zum Einsatz kommen. In Szenarien, in denen diese Eigenschaften irrelevant sind, ist der Einsatz symmetrischer Verfahren effizienter. Ein Standardszenario, in dem sich die Verwendung symmetrischer Verfahren zur Datenauthentisierung anbietet, stellt der integritätsgesicherte Transport von verschlüsselten Daten über ein Netzwerk nach Aushandlung ephemerer Schlüssel dar.

5.1. Sicherheitsziele

Beim Einsatz kryptographischer Verfahren zur Datenauthentisierung ist eine Klärung der Sicherheitsziele, die im jeweiligen Szenario erreicht werden sollen, für die Auswahl der Mechanismen von entscheidender Bedeutung. Vereinfacht lassen sich die folgenden Szenarien unterscheiden, die in vielen Anwendungen wichtig sind:

- Sicherung der Integrität von Daten, die über ein Netzwerk übermittelt werden, auf dem Weg vom Sender zum Empfänger: In diesem Anwendungsfall besitzen Sender und Empfänger in der Regel ein gemeinsames Geheimnis, und der Empfänger hat kein Interesse daran, gefälschte Übertragungen zu erzeugen. Es bietet sich somit die Nutzung eines symmetrischen Verfahrens zur Datenauthentisierung an.
- Sicherung der Nichtabstreitbarkeit einer Nachricht: Hier soll sichergestellt werden, dass der Besitzer eines bestimmten Schlüssels zuverlässig als Urheber einer Nachricht identifiziert werden kann und dass der Urheber selbst eine signierte Nachricht nicht so erstellen kann, dass über die Validität der Signatur nachträglich Zweifel entstehen können. In solchen Situationen dürfen die Prüfer einer Nachricht nicht über den entsprechenden Signaturschlüssel verfügen, daher kommt in diesem Fall nur die Verwendung digitaler Signaturen

in Frage. Zudem muss der private Signaturschlüssel je nach konkretem Szenario und angestrebtem Schutzniveau gegebenenfalls auch vor Einsichtnahme durch den Signaturgeber selbst geschützt werden. Dies ist zum Beispiel der Fall, wenn die Gefahr besteht, dass der Signaturersteller vergangene Signaturen durch absichtliche Verbreitung des eigenen privaten Schlüssels nachträglich ungültig machen könnte. Außerdem muss sichergestellt werden, dass dem Empfänger die Nachricht genauso angezeigt wird wie dem Ersteller, und dass etwaige nichtsignierte Anteile (zum Beispiel die nichtsignierte Betreffzeile im Fall einer signierten E-Mail) für den Empfänger und auch für den Ersteller eindeutig als solche identifizierbar sind.

- Absicherung eines asymmetrischen Schlüsselaustausches gegen Man-in-the-Middle-Angriffe: In diesem Anwendungsfall steht kein gemeinsames Geheimnis zur Verfügung, so dass eine integritätsgeschützte Übermittlung der Key-Exchange-Nachrichten mittels digitaler Signaturen sichergestellt werden muss.

Bemerkung 5.1 In einigen Anwendungsszenarien kann es zu speziellen Anforderungen an die beteiligten Sicherheitsfunktionen kommen. Zum Beispiel verfolgt eine Codesignatur die Sicherheitsziele der Integrität der übermittelten Anwendung sowie der Nichtabstreitbarkeit möglicherweise enthaltener Schadfunktionen in der ausgelieferten Software, obwohl die signierten Daten in der Regel weder beim Empfänger noch beim Ersteller sinnvoll angezeigt und mit vertretbarem Aufwand inhaltlich geprüft werden können. Die Sicherheitsfunktion der sicheren Anzeige auf Erstellerseite verlagert sich damit vollständig auf die Prozesse zur Qualitätssicherung beim Ersteller sowie auf die Sicherheit der von ihm eingesetzten technischen Komponenten.

Bemerkung 5.2 Bei der Verarbeitung authentisierter Daten dürfen nur die Datenbestandteile, die wirklich signiert wurden, als integer angesehen werden. Die Durchsetzung dieses Grundsatzes ist nicht immer einfach, auch weil für eine Anwendung kritische Fälle unter Umständen in legitim signierten Daten niemals auftauchen. Besonders bei der Verwendung komplexerer Signaturformate (zum Beispiel XML-Signaturen) oder in Kontexten, in denen Sicherheitsziele durch digitale Signaturen durchgesetzt werden sollen, die bei der Entwicklung der eingesetzten Komponenten nicht vorhergesehen wurden, sollte daher immer durch einen Experten gründlich überprüft werden, ob gegebenenfalls zusätzliche Schutzmaßnahmen erforderlich sind.

Bemerkung 5.3 Die Authentizität signierter Daten kann durch eine Signatur unter Umständen noch nicht in ausreichendem Maße garantiert werden, etwa wenn Replay-Attacken möglich sind. Derartige Angriffe müssen durch zusätzliche Maßnahmen unterbunden werden. Im Allgemeinen kann dies durch eine geeignete Kombination von Verfahren zur Datenauthentisierung mit Verfahren zur Durchführung einer Challenge-Response-basierten Instanzauthentisierung erreicht werden. In einigen Anwendungsfällen (zum Beispiel Softwareupdates, Schlüsselupdates) kann auch die Überprüfung mitsignierter Versionszähler oder Zeitstempel ausreichend sein.

5.2. Message Authentication Code (MAC)

Message Authentication Codes sind symmetrische Verfahren zur Datenauthentisierung, die üblicherweise auf Blockchiffren oder Hashfunktionen basieren und zum Einsatz kommen, wenn große Datenmengen authentisiert werden sollen oder wenn Prüfung oder Erstellung von Prüfsummen aus anderen Gründen besonders effizient sein muss. Voraussetzung in diesem Fall ist, dass Beweisender und Prüfer vorab einen gemeinsamen symmetrischen Schlüssel vereinbart haben. Häufig müssen sowohl die Vertraulichkeit als auch die Authentizität der Daten gewährleistet werden, derartige Verfahren werden in Abschnitt A.1 behandelt. In Kapitel 2 werden Verfahren vorgestellt, die den Schlüsselaustausch über unsichere Kanäle ermöglichen.

Die Entwicklung fehlertoleranter Quantencomputer hat wesentlich weniger gravierende Auswirkungen auf die Sicherheit symmetrischer Verfahren als auf die Sicherheit von asymmetrischen Verfahren. Der Grover-Algorithmus [56] könnte theoretisch die Durchsuchung des Schlüsselraums symmetrischer Verfahren quadratisch beschleunigen. Ob eine Beschleunigung im Vergleich zur klassischen Durchsuchung des Schlüsselraums auch praktisch erreicht werden kann, ist Gegenstand aktueller Forschung und nicht abschließend geklärt (siehe z.B. [71]). Insbesondere bei Anwendungen mit hohem oder langfristigem Schutzbedarf oder langlebigen Systemen ist es dennoch ratsam, bei den im Folgenden empfohlenen Message Authentication Codes eine Schlüssellänge von 256 Bits zu nutzen.

Grundsätzlich gelten die folgenden Verfahren als sicher, wenn im CMAC-Verfahren und im GMAC-Verfahren eine der in Tabelle 3.1 aufgeführten Blockchiffren beziehungsweise im HMAC-Verfahren eine der in Tabelle 4.1 aufgeführten Hashfunktionen eingesetzt wird und die Länge des Schlüssels für sämtliche Verfahren mindestens 128 Bits beträgt.

- CMAC, siehe [97],
- HMAC, siehe [10],
- KMAC128, KMAC256, siehe [96],
- GMAC, siehe [90].

Tabelle 5.1: Empfohlene MAC-Verfahren.

Bei der Verwendung dieser Verfahren sind folgende Aspekte zu beachten:

- Bei CMAC, HMAC und KMAC handelt es sich um pseudozufällige Funktionen, bei GMAC hingegen nicht. Ferner benötigt GMAC im Vergleich zu den anderen beiden Verfahren eine 96 Bit-Nonce als Initialisierungsvektor.
- Als Tag-Länge wird für allgemeine kryptographische Anwendungen in allen oben genannten Verfahren eine Länge von ≥ 96 Bits empfohlen. Kürzere Tag-Längen dürfen nur nach Abwägung aller die jeweilige Anwendung betreffenden Umstände durch Experten verwendet werden. Für GMAC-Tags existieren Angriffe, bei denen Fälschungen von Tags der Länge t für Nachrichten, deren Länge n Blöcke beträgt, mit einer Wahrscheinlichkeit von $2^{-t+\log_2(n)}$ pro Versuch möglich sind und sich diese Wahrscheinlichkeit bei Detektion erfolgreicher Fälschungen weiter steigert [51]. Dies bedeutet, dass GMAC (und damit auch der authentisierte Verschlüsselungsmodus GCM) bei gleicher Tag-Länge einen schwächeren Integritätsschutz liefert als es für CMAC oder HMAC mit den jeweils in dieser Technischen Richtlinie empfohlenen Blockchiffren beziehungsweise Hashfunktionen erwartet wird. Die praktische Relevanz dieser Angriffe wächst erheblich, wenn kurze Authentisierungs-Tags (< 96 Bits) eingesetzt werden. Von einer Verwendung kurzer Tags mit GMAC/GCM wird daher dringend abgeraten.
- Bei der Verwendung des HMAC-Verfahrens sollte die Tag-Länge höchstens auf die Hälfte der Ausgabelänge der Hashfunktion gekürzt werden.
- Hinsichtlich des GMAC-Verfahrens gelten die sonstigen Bemerkungen zu den Betriebsbedingungen für GCM aus Abschnitt 3.1.2 entsprechend, soweit sie die Authentisierungsfunktion betreffen.
- Die verwendeten Authentisierungsschlüssel sind ebenso gut zu schützen wie sonstige kryptographische Geheimnisse im gleichen Kontext.

- Allgemein müssen alle Auflagen aus [10, 97, 96, 90] bei dem jeweils verwendeten Verfahren eingehalten und ihre Einhaltung dokumentiert werden.

Die Tabelle 5.2 fasst die Empfehlungen zu Schlüssel- und Prüfsummenlänge bei Verwendung von MAC-Verfahren zusammen.

Verfahren	CMAC	HMAC	KMAC128	KMAC256	GMAC
Schlüssellänge	≥ 128	≥ 128	≥ 128	≥ 256	≥ 128
Tag-Länge empfohlen	≥ 96	≥ 128	≥ 96	≥ 96	≥ 96

Tabelle 5.2: Parameter für empfohlene MAC-Verfahren.

5.3. Signaturverfahren

In Signaturverfahren wird mit einem privaten Schlüssel eine Signatur zu einer gegebenen Nachricht (bzw. bei einigen Verfahren zu einem Hash dieser Nachricht) berechnet. Der Prüfer verifiziert anschließend die Signatur mit dem entsprechenden öffentlichen Schlüssel. Wie schon bei asymmetrischen Verschlüsselungsverfahren darf es dabei praktisch nicht möglich sein, die Signatur ohne Kenntnis des privaten Schlüssels zu berechnen. Dies impliziert insbesondere, dass es praktisch nicht möglich sein darf, den privaten Schlüssel aus dem öffentlichen Schlüssel zu konstruieren.

Zur Verteilung der öffentlichen Schlüssel an die Verifizierer wird üblicherweise eine Public-Key-Infrastruktur genutzt. In jedem Fall ist ein zuverlässiger (vor Manipulationen sicherer) Weg zur Verteilung der öffentlichen Schlüssel wie bei allen Public-Key-Verfahren unerlässlich. Eine tiefgehende Diskussion der technischen und organisatorischen Möglichkeiten zur Lösung dieses Problems geht allerdings deutlich über den Rahmen der vorliegenden Technischen Richtlinie hinaus, das Thema wird daher nur am Rande betrachtet.

Für die Spezifizierung von Signaturverfahren sind folgende Algorithmen festzulegen:

- Ein Algorithmus zur Generierung von Schlüsselpaaren.
- Ein Algorithmus zum Signieren der (gegebenenfalls gehashten) Daten und ein Algorithmus zum Verifizieren der Signatur.
- Falls erforderlich: eine Hashfunktion, die die zu signierenden Daten zunächst auf einen Datenblock fester Bitlänge abbildet.

Für die gegebenenfalls erforderliche Berechnung des Hashwertes sind grundsätzlich alle der in Tabelle 4.1 aufgelisteten Hashfunktionen geeignet, es verbleibt somit die Angabe der unter dem ersten beziehungsweise dritten Punkt aufgeführten Algorithmen und Schlüssellängen. Zusätzlich werden Empfehlungen für minimale Schlüssellängen angegeben.

Tabelle 5.3 liefert einen Überblick über die im Folgenden empfohlenen Signaturverfahren. Alle empfohlenen Verfahren können sowohl zur Signierung von Daten als auch zum Ausstellen von Zertifikaten genutzt werden.

Nach heutigem Kenntnisstand erreichen bei geeigneter Wahl der Sicherheitsparameter alle hier empfohlenen Signaturverfahren ein vergleichbares Sicherheitsniveau, wenn die privaten Schlüssel zuverlässig geheim gehalten werden und insbesondere nicht aufgrund von Implementierungsschwächen, wie beispielsweise durch Seitenkanäle, Fault-Attacken oder gegen eine bestimmte Art der Schlüsselgenerierung ausgerichtete mathematische Angriffe, ermittelt werden können.

Bemerkung 5.4 Mit Ausnahme des DS 3 (vergleiche Tabelle 5.4) sind die empfohlenen asymmetrischen Signaturverfahren probabilistische Algorithmen[1]. Es wird also bei jeder Berechnung einer

[1] Der RSA-Algorithmus selbst ist deterministisch, nicht aber die hier empfohlenen Paddingverfahren zu RSA (außer DS 3).

- Klassische Signaturverfahren (nicht quantensicher)
  - RSA, siehe [63],
  - DSA[1], siehe [66],
  - DSA-Varianten auf elliptischen Kurven:
    - ECDSA, siehe [27],
    - ECKDSA/ECKCDSA, ECGDSA, siehe [27, 66], und
- Quantensichere Signaturverfahren
  - ML-DSA in der „hedged"-Variante, siehe [105],
  - SLH-DSA in der „hedged"-Variante, siehe [106], und
  - Zustandsbehaftete hashbasierte Signaturverfahren[2](LMS/HSS und XMSS/XMSS^{MT}), siehe [99].

Tabelle 5.3: Empfohlene Signaturverfahren.

[1] Die Verwendung des Signaturverfahrens DSA (siehe Abschnitt 5.3.2) wird aufgrund der geringen Verbreitung und der Abkündigung in [103] in der vorliegenden Technischen Richtlinie nur noch bis 2029 empfohlen.

[2] Zustandsbehaftete hashbasierte Signaturverfahren unterscheiden sich in wesentlichen Aspekten von den anderen an dieser Stelle empfohlenen Signaturverfahren. Für eine genauere Beschreibung der wichtigsten Punkte wird auf Abschnitt 5.3.4.3 verwiesen.

Signatur ein neuer Zufallswert benötigt, die Anforderungen an diese Zufallswerte werden in den entsprechenden Abschnitten angegeben.

5.3.1. RSA

Die Sicherheit des RSA-Verfahrens beruht auf der angenommenen Schwierigkeit der Berechnung e-ter Wurzeln in $\mathbb{Z}_n$, wobei n eine ganze Zahl von unbekannter Faktorisierung in zwei Primfaktoren p, q ist und e ein Exponent, der zu $\varphi(n) = (p-1)(q-1)$ teilerfremd ist.

Schlüsselgenerierung Die Schlüsselgenerierung verläuft analog wie beim RSA-Verschlüsselungsverfahren, für Details siehe Abschnitt 2.3.2. Der Signaturprüfschlüssel lautet (n, e), wobei $n = p \cdot q$ zusammengesetzt, e invertierbar $\bmod \varphi(n)$ und $2^{16} < e < 2^{256}$ ist, der Signaturschlüssel lautet $d := e^{-1} \bmod \varphi(n)$.

Signaturerzeugung und Signaturverifikation Für die Signaturerzeugung beziehungsweise -verifikation sei auf [63] verwiesen. Dabei muss vor Anwendung des privaten Schlüssels d auf die Bitlänge des Moduls n zusätzlich der Hashwert der Nachricht formatiert werden. Das Formatierungsverfahren ist sorgfältig zu wählen (siehe zum Beispiel [45]), die folgenden Verfahren werden empfohlen:

- EMSA-PSS, siehe [86],
- Digital Signature Scheme (DS) 2 und 3, siehe [68].

Tabelle 5.4: Empfohlene Formatierungsverfahren für den RSA-Signaturalgorithmus.

Schlüssellänge Die Länge des Modulus n sollte mindestens 3000 Bits betragen, siehe auch Tabelle 2.2.

5.3.2. Digital Signature Algorithm (DSA)

Die Sicherheit des DSA-Verfahrens beruht auf der angenommenen Schwierigkeit der Berechnung diskreter Logarithmen in $\mathbb{F}_p^*$.

Schlüsselgenerierung

1.) Wähle zwei Primzahlen p und q, so dass $q \mid (p-1)$.

2.) Wähle $x \in \mathbb{F}_p^*$ und berechne $g := x^{(p-1)/q} \bmod p$.

3.) Falls $g = 1$, gehe zu 2.).

4.) Wähle zufällig eine Zahl $a \in \{2, \ldots, q-1\}$ und setze $A := g^a$.

Dann ist (p, q, g, A) der öffentliche Schlüssel und a der private Schlüssel.

Signaturerzeugung und Signaturverifikation Für die Signaturerzeugung beziehungsweise -verifikation sei auf [66] und [103] verwiesen. Sowohl Signaturerzeugung als auch Signaturverifikation benötigen eine kryptographische Hashfunktion. Dabei sollte eine der in der vorliegenden Richtlinie empfohlenen Hashfunktionen verwendet werden und die Länge der Hashwerte der Bitlänge von q entsprechen. Falls keine der in Tabelle 4.1 empfohlenen Hashfunktionen eine geeignete Hashlänge aufweisen, sollten die k führenden Bits der Hash-Ausgabe verwendet werden, wobei k die Bitlänge von q bezeichnet. Ist die Länge L_H des Hashwertes *geringer* als die Bitlänge von q, resultiert daraus ein Signaturverfahren mit einem Sicherheitsniveau von (höchstens) $L_H/2$ Bits.

Schlüssellänge Die Länge der Primzahl p sollte mindestens 3000 Bits betragen, , die Länge der Primzahl q sollte mindestens 250 Bits betragen.

Bemerkung 5.5 Die Verwendung des Signaturverfahrens DSA wird aufgrund der geringen Verbreitung und der Abkündigung in [103] in der vorliegenden Technischen Richtlinie nur noch bis 2029 empfohlen.

Bemerkung 5.6 Das DSA-Verfahren ist ein sogenannter probabilistischer Algorithmus, da zur Berechnung der Signatur eine Zufallszahl $k \in \{1, \ldots, q-1\}$ benötigt wird. Diese sollte bezüglich der Gleichverteilung auf $\{1, \ldots, q-1\}$ gewählt werden, da andernfalls Angriffe existieren, vergleiche [110]. Zwei Algorithmen zur Berechnung von k werden in Abschnitt B.4 vorgestellt.

Bemerkung 5.7 Zur Erzeugung der Systemparameter siehe Bemerkung 2.12.

5.3.3. DSA-Varianten basierend auf elliptischen Kurven

Die Sicherheit dieser Verfahren beruht auf der angenommenen Schwierigkeit der Berechnung diskreter Logarithmen in elliptischen Kurven.

Schlüsselgenerierung

1.) Erzeuge kryptographisch starke EC-Systemparameter (p, a, b, P, q, i), siehe Abschnitt B.3.

2.) Wähle d zufällig und gleichverteilt in $\{1, \ldots, q-1\}$.

3.) Setze $G := d \cdot P$.

Dann bilden die EC-Systemparameter (p, a, b, P, q, i) zusammen mit G den öffentlichen Schlüssel und d den privaten Schlüssel.

Signaturerzeugung und Signaturverifikation Die folgenden Algorithmen werden in der vorliegenden Technischen Richtlinie empfohlen:

- ECDSA, siehe [27],
- ECKDSA/ECKCDSA, ECGDSA, siehe [27, 66].

Tabelle 5.5: Empfohlene Signaturverfahren basierend auf elliptischen Kurven.

Bei Signaturerzeugung und Signaturverifikation wird eine kryptographische Hashfunktion benötigt. Dabei sind grundsätzlich alle in dieser Technischen Richtlinie empfohlenen Hashfunktionen geeignet. Die Länge der Hashwerte sollte der Bitlänge von q entsprechen. Die sonstigen Hinweise zur Wahl der Hashfunktion aus Abschnitt 5.3.2 gelten entsprechend.

Schlüssellänge Alle in Tabelle 5.5 aufgeführten Signaturverfahren garantieren ein Sicherheitsniveau von n Bits, wenn für die Ordnung q des Basispunktes P die Relation $q \geq 2^{2n}$ gilt und angenommen wird, dass die Berechnung diskreter Logarithmen auf den verwendeten Kurven nicht effizienter möglich ist als durch generische Verfahren. Es wird empfohlen, $q \geq 2^{250}$ zu wählen.

Bemerkung 5.8 Wie das DSA-Verfahren sind alle in diesem Abschnitt empfohlenen Signaturverfahren probabilistische Algorithmen. Auch hier muss ein Zufallswert $k \in \{1, \ldots, q-1\}$ gemäß der Gleichverteilung auf $\{1, \ldots, q-1\}$ gewählt werden, da andernfalls Angriffe existieren, vergleiche [110]. Zwei Verfahren zur Berechnung von k werden in Abschnitt B.4 vorgestellt.

5.3.4. Quantensichere Signaturverfahren

Wegen der Bedrohung der klassischen Signaturverfahren durch die fortschreitende Entwicklung hinreichend großer Quantencomputer sollte eine Migration zu quantensicheren Verfahren erfolgen. Dies betrifft insbesondere sensitive Anwendungen mit langen Migrationszeiten, beispielsweise im Zusammenhang mit Public-Key-Infrastrukturen.

Diese Technische Richtlinie empfiehlt den Einsatz eines quantensicheren Signaturverfahrens grundsätzlich nur in Kombination mit einem klassischen Signaturverfahren. Die Hybridisierung sollte so erfolgen, dass das hybride Signaturverfahren sicher ist, solange mindestens eines der Verfahren sicher ist. Eine natürliche und robuste Hybridisierung ist die Konkatenation von einer

quantensicheren mit einer klassischen Signatur, so dass die konkatenierte Signatur genau dann als gültig anerkannt wird, wenn sämtliche einzelnen Signaturen gültig sind. Hierbei sollte darauf geachtet werden, Schlüsselmaterial für hybride Signaturen dediziert nur für diesen Zweck zu erzeugen und nicht etwa auch für nicht-hybride Signaturen zu verwenden.

Der Sicherheit von hashbasierten Signaturverfahren liegen nur komplexitätstheoretische Annahmen über kryptografische Hashfunktionen zugrunde. Daher können hashbasierte Signaturen unter der Voraussetzung, dass die Implementierungssicherheit von zustandsbehafteten und zustandslosen hashbasierten Verfahren sorgfältig berücksichtigt wird, grundsätzlich auch alleine (das heißt nicht hybrid) zum Einsatz kommen und als eine gute Methode für die Erstellung langfristig sicherer Signaturen dienen.

5.3.4.1. SLH-DSA

Die Sicherheit des zustandslosen hashbasierten Signaturverfahrens SLH-DSA [106] beruht auf den Sicherheitseigenschaften der verwendeten Hashfunktion. Zu diesen Eigenschaften zählt insbesondere die Urbildresistenz, jedoch nicht die Kollisionsresistenz.

Schlüsselgenerierung, Signaturerzeugung und Signaturverifikation sind in [106] beschrieben. Die vorliegende Technische Richtlinie empfiehlt die „hedged“-Variante von SLH-DSA und alle Parametersätze entsprechend NIST Security Strength Categories 3 und 5 aus [106], das heißt

- SLH-DSA-SHA2-192s in der „hedged“-Variante, siehe [106],
- SLH-DSA-SHAKE-192s in der „hedged“-Variante, siehe [106],
- SLH-DSA-SHA2-192f in der „hedged“-Variante, siehe [106],
- SLH-DSA-SHAKE-192f in der „hedged“-Variante, siehe [106],
- SLH-DSA-SHA2-256s in der „hedged“-Variante, siehe [106],
- SLH-DSA-SHAKE-256s in der „hedged“-Variante, siehe [106],
- SLH-DSA-SHA2-256f in der „hedged“-Variante, siehe [?],
- SLH-DSA-SHAKE-256f in der „hedged“-Variante, siehe [106].

Tabelle 5.6: Empfohlene Parametersätze für SLH-DSA.

Bemerkung 5.9 Bevorzugt wird die „pure“-Version von SLH-DSA; für spezielle Anwendungsfälle kann auch die „pre-hash“-Version von SLH-DSA gemäß den Hinweisen in [106] zum Einsatz kommen.

5.3.4.2. ML-DSA

Die Sicherheit des Signaturverfahrens ML-DSA [105] beruht auf dem Module Learning with Errors (MLWE) Problem-Problem und einer Variante des Module Short Integer Solution (MSIS) Problem.

Schlüsselgenerierung, Signaturerzeugung und Signaturverifikation sind in [105] beschrieben. Die vorliegende Technische Richtlinie empfiehlt die „hedged“-Variante von ML-DSA und alle Parametersätze entsprechend NIST Security Strength Categories 3 und 5 aus [105], das heißt

- ML-DSA-65 in der „hedged“-Variante, siehe [105],
- ML-DSA-87 in der „hedged“-Variante, siehe [105].

Tabelle 5.7: Empfohlene Parametersätze für ML-DSA.

Bemerkung 5.10 Bevorzugt wird die „pure“-Version von ML-DSA; für spezielle Anwendungsfälle kann auch die „pre-hash“-Version von ML-DSA gemäß den Hinweisen in [105] zum Einsatz kommen.

5.3.4.3. Zustandsbehaftete Hashbasierte Signaturverfahren

Die Sicherheit der zustandsbehafteten hashbasierten Signaturverfahren XMSS/XMSS^{MT} und LMS/HSS beruht auf den Sicherheitseigenschaften der jeweils verwendeten Hashfunktion. Zu diesen Eigenschaften zählt insbesondere die Urbildresistenz, jedoch nicht die Kollisionsresistenz. Allerdings stellt bei diesen Verfahren die Zustandsbehaftung eine Herausforderung für die Schlüsselverwaltung dar: Durch die Zustandsbehaftung ist die Anzahl der erstellbaren Signaturen pro Schlüsselpaar beschränkt und bei jeder Signaturerzeugung muss ein streng monotoner Zähler fehlerfrei hochgezählt werden. Daher werden zustandsbehaftete Signaturverfahren nur in den speziellen Szenarien empfohlen, in denen eine sichere und fehlerfreie Zustandsverwaltung gewährleistet werden kann. Ein solches spezielles Szenario ist beispielsweise bei Signaturen von Software- und Firmware-Updates gegeben; siehe auch [108].

Schlüsselgenerierung, Signaturerzeugung und Signaturverifikation sind in [99] beschrieben, wobei zum Teil auf die entsprechenden RFCs [60] und [82] verwiesen wird. Die vorliegende Technische Richtlinie empfiehlt alle Parametersätze aus [99], das heißt

- XMSS/XMSS^{MT}, alle Parametersätze gemäß [99],
- LMS/HSS, alle Parametersätze gemäß [99].

Tabelle 5.8: Empfohlene zustandsbehaftete hashbasierte Signaturverfahren.

Bemerkung 5.11 In [111] wird ein Angriff gegen SLH-DSA (vormals SPHINCS+) mit SHA-256 als zugrundeliegender Hashfunktion beschrieben. Dieser Angriff reduziert die Sicherheit der angegriffenen SLH-DSA-Instanz um ca. 40 Bit, das heißt statt der behaupteten Sicherheit von 256 Bit bietet diese SLH-DSA-Instanz nur noch eine Sicherheit von ca. 216 Bit. Theoretisch lässt sich die Idee dieses Angriffs auf LMS/HSS in Kombination mit SHA-256 übertragen, so dass dieser Angriff bei der Nutzung derartiger LMS/HSS-Instanzen berücksichtigt werden sollte. Auch unter Berücksichtigung des Angriffs liegt die Sicherheit der in [99] standardisierten Parametersätzen deutlich über dem in der vorliegenden Technischen Richtlinie angestrebten Sicherheitsniveau von 120 Bit.[2]

5.3.5. Langfristige Beweiswerterhaltung für digitale Signaturen

Unabhängig von den vorliegenden Empfehlungen zu Verfahren und Schlüssellängen für digitale Signaturen wird dazu geraten, die Möglichkeit künftiger Umstellungen der Systeme auf neue Signa-

[2] Die in [106] standardisierten Parametersätze für SLH-DSA sind von diesem Angriff nicht betroffen.

turverfahren oder längere Signaturschlüssel schon bei der Entwicklung zu berücksichtigen, wenn die vorgesehene Zeitdauer, über die hinweg die Authentizität und Integrität der durch ein System zur Datenauthentisierung zu schützenden Daten gesichert bleiben soll, den Vorhersagezeitraum der vorliegenden Richtlinie deutlich übersteigt. Dies sollte Mechanismen zur Übersignierung alter signierter Dokumente unter Verwendung der aktualisierten Verfahren mit einschließen. Nähere Informationen zu diesem Thema finden sich in der Technischen Richtlinie TR-03125 (TR-ESOR) [31].

6. Instanzauthentisierung

Unter Instanzauthentisierung werden in dieser Technischen Richtlinie kryptographische Protokolle verstanden, in denen ein Beweisender einem Prüfenden den Besitz eines Geheimnisses nachweist. Bei symmetrischen Verfahren ist dies ein symmetrischer Schlüssel, der vorab ausgetauscht werden muss. Bei asymmetrischen Verfahren zeigt der Beweisende, dass er im Besitz eines privaten Schlüssels ist. Hierfür wird in der Regel eine PKI benötigt, damit der Prüfende dem Beweisenden den zugehörigen öffentlichen Schlüssel zuordnen kann. Passwortbasierte Verfahren dienen in erster Linie der Freischaltung von Chipkarten oder anderen kryptographischen Komponenten. Hier beweist der Inhaber der Komponente, dass er im Besitz eines Passwortes oder einer PIN ist.

Die Authentisierung sollte – wo sinnvoll und möglich – gegenseitig erfolgen und kann mit einer Schlüsseleinigung einhergehen, um die Vertraulichkeit und Integrität einer anschließenden Kommunikation zu gewährleisten, siehe Kapitel 2 für empfohlene Schlüsselaustausch- und Schlüsseleinigungsverfahren und Abschnitt A.2 für empfohlene Protokolle, die beide Verfahren kombinieren.

In diesem Kapitel werden für die ersten beiden Verfahren (Abschnitte 6.1 und 6.2) nur allgemeine Ideen zur Instanzauthentisierung angegeben und lediglich die entsprechenden kryptographischen Primitive empfohlen. Für die benötigten kryptographischen Protokolle sei auf Abschnitt A.2 verwiesen. Insbesondere werden dort auch Empfehlungen für Schlüssellängen etc. angegeben.

6.1. Symmetrische Verfahren

Für den Nachweis eines Beweisenden (B) gegenüber einem Prüfenden (P), dass B im Besitz des geheimen symmetrischen Schlüssels ist, sendet P einen Zufallswert r an B. Damit das Verfahren das in der vorliegenden Technischen Richtlinie minimal angestrebte Sicherheitsniveau erreicht, sollte r mindestens 120 Bits Min-Entropie besitzen. Wird eine große Anzahl von Authentisierungsverfahren mit dem gleichen privaten Schlüssel durchgeführt, dann sollte die Wahrscheinlichkeit einer Kollision zweier dieser Challenge-Werte auf $\leq 2^{-32}$ beschränkt werden. B berechnet anschließend mittels des gemeinsamen privaten Schlüssels K eine Prüfsumme von r und schickt diese zurück an P, der diese dann prüft. Derartige Verfahren heißen auch *Challenge-Response-Verfahren*, siehe Tabelle 6.1 für eine schematische Darstellung.

Beweisender (B)		Prüfender (P)
		Wähle Zufallswert r
	$\xleftarrow{r}$ (Challenge)	
Berechne Prüfsumme c		
	$\xrightarrow{c}$ (Response)	
		Verifiziere Prüfsumme

Tabelle 6.1: Schematische Darstellung eines Challenge-Response-Verfahren zur Instanzauthentisierung.

Die Berechnung und Verifikation der Prüfsumme hängt vom gewählten Verfahren ab. Grundsätzlich können alle in Kapitel 3 empfohlenen Verschlüsselungsverfahren und alle in Abschnitt 5.2

empfohlenen MAC-Verfahren eingesetzt werden. Für empfohlene Bitlängen und Nebenbedingungen an die benutzten Zufallswerte sei auf Abschnitt A.2 verwiesen.

6.2. Asymmetrische Verfahren

Auch für asymmetrische Verfahren werden Challenge-Response-Protokolle zur Instanzauthentisierung eingesetzt. Hierfür berechnet der Beweisende mit seinem privaten Schlüssel eine Prüfsumme zu einem vom Prüfer gesendeten Zufallswert r. Der Prüfende verifiziert dann die Prüfsumme mithilfe des zugehörigen öffentlichen Schlüssels. Grundsätzlich können hierfür alle in Abschnitt 5.3 empfohlenen Verfahren eingesetzt werden. Für empfohlene Bitlängen und Nebenbedingungen an die benutzten Zufallswerte siehe ebenfalls Abschnitt A.2.

Bemerkung 6.1 Wenngleich die in Abschnitt 5.3 empfohlenen Signaturverfahren zur Datenauthentisierung auch zur Instanzauthentisierung genutzt werden können, sollte darauf geachtet werden, dass die eingesetzten Schlüssel verschieden sind, das heißt, dass ein Schlüssel zur Erzeugung von Signaturen nicht zur Instanzauthentisierung eingesetzt wird. Dies muss in den entsprechenden Zertifikaten für die öffentlichen Schlüssel kenntlich gemacht werden.

6.3. Passwortbasierte Verfahren

Passwörter zum Freischalten der auf kryptographischen Komponenten (beispielsweise Signaturkarten) zur Verfügung gestellten kryptographischen Schlüssel sind meist kurz, damit sich der Inhaber der Komponente das Passwort merken kann. In vielen Situationen ist zudem der erlaubte Zeichensatz begrenzt auf die Ziffern $0, \ldots, 9$; in diesem Fall spricht man auch von PIN statt Passwort. Um trotzdem ein ausreichendes Sicherheitsniveau zu erreichen, wird die Anzahl der Zugriffsversuche üblicherweise begrenzt.

6.3.1. Empfohlene Passwortlängen für den Zugriff auf kryptographische Hardwarekomponenten

Folgende Passwortlängen und Anzahl der Zugriffsversuche für den Zugriff auf kryptographische Hardwarekomponenten werden empfohlen:

- Es wird grundsätzlich empfohlen, Passwörter mit einer Entropie von mindestens $\log_2(10^6)$ Bits zu verwenden. Dies kann zum Beispiel durch eine ideal zufällige Vergabe sechsstelliger PINs erreicht werden.
- Die Anzahl der aufeinanderfolgenden erfolglosen Zugriffsversuche muss stark eingeschränkt werden. Bei einer Passwortentropie von $\log_2(10^6)$ Bits wird eine Beschränkung auf drei Versuche empfohlen.

Tabelle 6.2: Empfohlene Passwortlängen und Anzahl der Zugriffsversuche für den Zugriffsschutz kryptographischer Komponenten.

Bemerkung 6.2 Werden Zugriffs-Passwörter für kryptographische Komponenten nicht (zumindest annähernd) ideal zufällig durch einen technischen Prozess erzeugt, sondern durch den Nutzer gewählt, wird dringend eine Sensibilisierung des Nutzers bezüglich der Auswahl sicherer Passwörter

empfohlen. Ferner wird in diesem Fall empfohlen, von der Verwendung rein numerischer Passwörter (PINs) abzusehen. Für Passwörter, die über einem Alphabet gebildet werden, das mindestens die Buchstaben A-Z, a-z, 0-9 und gegebenenfalls Sonderzeichen enthält, wird eine Länge von mindestens acht Zeichen empfohlen. Zudem wird empfohlen, Maßnahmen zu treffen, die die Wahl naheliegender Passwörter (zum Beispiel beliebige einzelne Wörter der Landessprache oder einer wichtigen Fremdsprache sowie Datumsangaben in naheliegenden Formaten) ausschließen.

Bemerkung 6.3 In manchen Anwendungen kann nach Abwägung sämtlicher Umstände durch Experten auch eine Nutzung von Passwörtern mit geringerer Entropie als oben empfohlen mit der vorliegenden Richtlinie kompatibel sein. Ein einzelner unautorisierter Zugriffsversuch sollte dabei aber wenigstens niemals mit einer Erfolgswahrscheinlichkeit größer als $\approx 10^{-4}$ erfolgreich sein. Die Anzahl der aufeinanderfolgenden erfolglosen Zugriffsversuche muss stark eingeschränkt werden, die genauen Beschränkungen sind dabei abhängig von der Anwendung. Die Restrisiken sollten gründlich dokumentiert werden und es wird empfohlen, den berechtigten Nutzer falls möglich über erfolgte unberechtigte Zugriffsversuche zu informieren, auch wenn die Komponente in deren Folge nicht gesperrt wurde.

Bemerkung 6.4 (i) Um Denial-of-Service Attacken oder eine versehentliche Sperrung der Komponente zu verhindern, muss ein Mechanismus zum Aufheben einer Sperrung vorgesehen sein. Die Entropie des Schlüssels zur Entsperrung (englisch *Personal Unblocking Key*, kurz PUK) sollte mindestens 120 Bits betragen, wenn Offline-Attacken denkbar sind.

(ii) Wenn keine Offline-Angriffe auf den PUK möglich sind, wird empfohlen, einen PUK mit mindestens 32 Bits Entropie zu verwenden (zum Beispiel 10 Ziffern) und nach einer relativ geringen Anzahl von Zugriffsversuchen (zum Beispiel 20) die in der Komponente enthaltenen kryptographischen Geheimnisse unwiderruflich zu löschen.

(iii) Die allgemeine Empfehlung von mindestens etwa 20 Bits Entropie für das in einem passwortbasierten Authentisierungsverfahren verwendete Passwort gilt nur für die Authentisierung einer Sicherheitskomponente gegenüber, die keine Offline-Angriffe erlaubt und die die angegebenen Beschränkungen hinsichtlich der Anzahl zulässiger Zugriffsversuche zuverlässig durchsetzen kann. In anderen Situationen, in denen diese Bedingungen nicht erfüllt sind (zum Beispiel wenn aus dem Passwort direkt ein kryptographisches Geheimnis abgeleitet wird, das Zugriff zu sensitiven Informationen verschafft), wird empfohlen, Passwörter über ein Verfahren auszuwählen, das mindestens 120 Bits Entropie liefert. Für den Zugang zu Daten oder für die Authentisierung von Transaktionen mit hohem Schutzbedarf wird grundsätzlich von einer Ein-Faktor-Authentisierung abgeraten. Stattdessen wird in dieser Situation eine Zwei-Faktor-Authentisierung durch Wissen (Kenntnis eines Passwortes) und Besitz (einer sicheren Hardwarekomponente) empfohlen.

6.3.2. Empfohlene Verfahren zur Passwort-basierten Authentisierung gegenüber kryptographischen Hardwarekomponenten

Für kontaktbehaftete Chipkarten kann derzeit noch auf eine kryptographische Absicherung der Übertragung der PIN an die Chipkarte abgesehen werden, wenn das Kartenterminal selbst als vertrauenswürdig eingestuft werden kann und ein physikalischer Abgriff beziehungsweise eine Manipulation der zwischen Leser und Karte übertragenen Information durch geeignete Maßnahmen in der Einsatzumgebung verhindert wird. Allerdings wird auch hier die kryptographische Absicherung (im Hinblick auf Integrität und Vertraulichkeit der übertragenen Identifizierungsdaten) empfohlen. Grundsätzlich eignen sich dafür die gleichen Verfahren wie für kontaktlose Karten.

Bei kontaktlosen Chipkarten kann die Kommunikation zwischen Kartenleser und Chipkarte auch noch aus einiger Entfernung mitgelesen werden. In diesem Fall kann das Passwort zur Freischaltung der Chipkarte also nicht einfach vom Kartenleser zur Chipkarte gesendet werden.

Folgendes passwortbasiertes Verfahren wird für den Zugriffsschutz auf kontaktlose Chipkarten empfohlen:

PACE: Password Authenticated Connection Establishment, siehe [26].

Tabelle 6.3: Empfohlenes passwortbasiertes Verfahren für den Zugriffsschutz auf kontaktlose Chipkarten.

Das in Tabelle 6.3 empfohlene Verfahren beweist der kontaktlosen Chipkarte nicht nur, dass der Benutzer im Besitz des korrekten Passwortes ist, sondern führt gleichzeitig ein Schlüsseleinigungsverfahren durch, so dass im Anschluss eine vertrauliche und authentisierte Kommunikation durchgeführt werden kann.

Bemerkung 6.5 Auch bei dem in Tabelle 6.3 empfohlenen Verfahren muss die Anzahl der Versuche beschränkt sein. Es wird empfohlen, die Chipkarte nach drei erfolglosen Versuchen zu sperren. Die weiteren Bemerkungen aus Abschnitt 6.3.1 gelten entsprechend.

7. Secret Sharing

Häufig müssen kryptographische Schlüssel über einen langen Zeitraum gespeichert werden. Dies erfordert insbesondere, dass Kopien dieser Schlüssel angelegt werden müssen, um einem Verlust der Schlüssel entgegegenzuwirken. Mit wachsender Anzahl der Kopien wächst jedoch auch die Wahrscheinlichkeit, dass das zu schützende Geheimnis kompromittiert wird. Daher wird in diesem Kapitel ein Verfahren angegeben, welches erlaubt, ein Geheimnis wie beispielsweise einen kryptographischen Schlüssel K so in n Teilgeheimnisse $K_1, \ldots, K_n$ aufzuteilen, dass beliebige $t \leq n$ dieser Teilgeheimnisse genügen, um das Geheimnis zu rekonstruieren, $t-1$ Teilgeheimnisse aber keine Information über K liefern. Eine weitere Anwendung dieses Verfahrens ist, ein Vieraugenprinzip oder allgemeiner ein t-aus-n-Augenprinzip zu gewährleisten, um zum Beispiel das Passwort für eine kryptographische Komponente so auf n verschiedene Anwender zu verteilen, dass mindestens t Anwender benötigt werden, um das Passwort zu rekonstruieren.

Das hier vorgestellte Secret-Sharing-Verfahren wurde von A. Shamir entwickelt und wird daher auch als Shamir Secret-Sharing-Verfahren bezeichnet, siehe [114]. Wir nehmen im Folgenden an, dass das zu verteilende Geheimnis ein Schlüssel K der Bitlänge r ist, das heißt $K = (k_0, \ldots, k_{r-1}) \in \{0,1\}^r$. Zur Berechnung der verteilten Geheimnisse auf n Benutzer, so dass t Benutzer das Geheimnis K rekonstruieren können, wird wie folgt vorgegangen:

1.) Wähle eine Primzahl $p \geq \max(2^r, n+1)$ und setze $a_0 := \sum_{i=0}^{r-1} k_i \cdot 2^i \bmod p$.

2.) Wähle unabhängig voneinander $t-1$ zufällige Werte $a_1, \ldots, a_{t-1} \in \{0, 1, \ldots, p-1\}$ gemäß der Gleichverteilung aus $\{0, 1, \ldots, p-1\}$. Die Werte $a_0, a_1, \ldots, a_{t-1}$ definieren ein zufälliges Polynom

$$f(x) = \sum_{j=0}^{t-1} a_j x^j \bmod p$$

über $\mathbb{F}_p$, für das $f(0) = a_0 = \sum_{i=0}^{r-1} k_i \cdot 2^i \bmod p$ gilt.

3.) Berechne die Werte $K_i := f(i)$ für alle $i \in \{1, \ldots, n\}$.

Tabelle 7.1: Berechnung der Teilgeheimnisse im Shamir Secret-Sharing-Verfahren.

Die Teilgeheimnisse K_i werden dann, zusammen mit i, dem i-ten Benutzer übergeben.

Bemerkung 7.1 Die Grundlage für das in Tabelle 7.1 genannte Verfahren stellt die sogenannte *Lagrange-Interpolations-Formel* dar, welche es ermöglicht, die Koeffizienten $a_0, \ldots, a_{t-1}$ eines unbekannten Polynoms f vom Grad $t-1$ aus t Punkten $(x_i, f(x_i))$ wie folgt zu bestimmen:

$$f(x) = \sum_{i=1}^{t} \left[f(x_i) \prod_{\substack{1 \leq j \leq t \\ j \neq i}} \frac{x - x_j}{x_i - x_j} \right] \bmod p.$$

Insbesondere lässt sich auf diese Weise $a_0 = f(0)$ (und damit K) aus t gegebenen Punkten berechnen.

Um aus t Teilgeheimnissen $K_{j_1}, \ldots, K_{j_t}$ (mit paarweise verschiedenen j_l) das Geheimnis K zu rekonstruieren, berechnet man $a_0 = \sum_{i=0}^{r-1} k_i \cdot 2^i \bmod p$ wie folgt:

1.) Berechne für alle $j \in \{j_1, \ldots, j_t\}$ den Wert $c_j = \prod_{\substack{1 \le l \le t \\ j_l \neq j}} \frac{j_l}{j_l - j} \bmod p$.

2.) Berechne $K = \sum_{l=1}^{t} c_{j_l} K_{j_l} \bmod p$.

Tabelle 7.2: Zusammensetzen der Teilgeheimnisse im Shamir Secret-Sharing-Verfahren.

Sowohl bei der Aufteilung in Teilgeheimnisse in Tabelle 7.1 als auch beim Zusammensetzen in Tabelle 7.2 ist zu beachten, dass jeweils in $\mathbb{F}_p$, das heißt modulo p, gerechnet wird.

Bemerkung 7.2 Die Bedingung $p \geq \max(2^r, n+1)$ garantiert, dass einerseits das Geheimnis als Element von $\mathbb{F}_p$ dargestellt werden kann und andererseits mindestens n unabhängige Teilgeheimnisse erzeugt werden können. Das Verfahren erreicht informationstheoretische Sicherheit, es ist also auch einem Angreifer mit unbeschränkten Ressourcen nicht möglich, das verteilte Geheimnis zu rekonstruieren, ohne t Teilgeheimnisse oder einen aus der Kenntnis von t Teilgeheimnissen auf geeignete Weise abgeleiteten Wert in Erfahrung zu bringen.

Die Sicherheit des Verfahrens hängt daher abgesehen von der angegebenen Bedingung nicht von weiteren Sicherheitsparametern ab. Allerdings muss durch organisatorische und technische Maßnahmen sichergestellt werden, dass ein Angreifer keine Kenntnis von t Teilgeheimnissen erlangen kann. Jegliche Kommunikation über die Teilgeheimnisse muss daher verschlüsselt und authentisiert stattfinden, soweit es einem Angreifer physikalisch möglich ist, diese Kommunikation aufzuzeichnen oder zu manipulieren.

Zudem ist die informationstheoretische Sicherheit nur gegeben, wenn die a_i für $i > 0$ echt zufällig und entsprechend der Gleichverteilung aus $\mathbb{F}_p$ gewählt werden. Um mindestens eine komplexitätstheoretische Sicherheit zu garantieren, sollte daher zur Erzeugung der a_i ein physikalischer Zufallsgenerator der Funktionalitätsklasse PTG.3 oder ein deterministischer Zufallsgenerator der Funktionalitätsklasse DRG.3 oder DRG.4 genutzt werden. Die Ausgabewerte dieses Zufallsgenerators müssen so nachbearbeitet werden, dass sie der Gleichverteilung auf $\mathbb{F}_p$ entsprechen; geeignete Verfahren hierzu finden sich in Abschnitt B.4.

8. Zufallszahlengeneratoren

Die Mehrzahl kryptographischer Anwendungen benötigt Zufallszahlen, etwa zur Erzeugung kryptographischer Langzeit- oder Ephemeralschlüssel, Systemparameter oder zur Instanzauthentisierung. Dies trifft für symmetrische und asymmetrische Verschlüsselungsverfahren ebenso zu wie für Signatur-, Authentisierungs- und Paddingverfahren. Ungeeignete Zufallszahlengeneratoren können grundsätzlich starke kryptographische Mechanismen entscheidend schwächen, daher ist bei kryptographischen Anwendungen besonders darauf zu achten, dass geeignete Zufallszahlengeneratoren eingesetzt werden. So sind – im Gegensatz zu beispielsweise numerischen Simulationen oder Experimenten, bei denen Reproduzierbarkeit eine wichtige Rolle spielen kann – im kryptographischen Kontext für die meisten Anwendungen Unvorhersagbarkeit und Geheimhaltung der Zufallszahlen beziehungsweise der aus ihnen abgeleiteten Werte unverzichtbare Eigenschaften. Selbst wenn ein Angreifer lange Teilfolgen von Zufallszahlen kennt, darf ihm dies nicht ermöglichen, deren Vorgänger oder Nachfolger zu bestimmen.

In der Regel wird bei der Erzeugung von Zufallszahlen das Ziel verfolgt, Ausgabewerte gleichverteilt auf $\{0,1\}^n$ zu erzeugen. In einigen Fällen werden jedoch Zufallszahlen mit bestimmten anderen, vorgegebenen Verteilungen benötigt. Aus diesem Grund enthält Anhang B Algorithmen, mit denen man aus den Ausgabewerten eines Zufallszahlengenerators Zufallswerte mit gewünschten Eigenschaften (zum Beispiel gleichverteilt auf $\{0,\ldots,q-1\}$) berechnen kann.

Im deutschen Zertifizierungschema sind die AIS 20 [33] (für deterministische Zufallszahlengeneratoren) und AIS 31 [34] (für physikalische Zufallszahlengeneratoren) verbindlich. Die beiden Dokumente verweisen auf eine gemeinsame mathematisch-technische Anlage. Die mathematisch-technische Anlage aus dem Jahr 2011 [32] wurde 2024 durch eine revidierte Fassung [37] ersetzt. Übergangsweise kann allerdings auch noch auf die alte Fassung [32] Bezug genommen werden.

Die mathematisch-technische Anlage [37] definiert Funktionalitätsklassen für physikalische Zufallszahlengeneratoren (PTG.2, PTG.3), für deterministische Zufallszahlengeneratoren (DRG.2, DRG.3, DRG.4 und DRT.1) und für nicht-physikalische nicht-deterministische Zufallszahlengeneratoren (NTG.1). Darüber hinaus erläutert [37] den mathematischen-technischen Hintergrund und die Anforderungen der Funktionalitätsklassen und illustriert die Konzepte anhand zahlreicher Beispiele. Im Vergleich zu [32] wird in [37] eine weitere Funktionalitätsklasse eingeführt (DRT.1), und zwei Funktionalitätsklassen (PTG.1, DRG.1) wurden gestrichen.

In den folgenden Abschnitten wird näher auf die verschiedenen Arten von Zufallszahlengeneratoren eingegangen. Die wichtigsten Empfehlungen zum Einsatz von Zufallszahlengeneratoren in allgemeinen kryptographischen Anwendungen lassen sich wie folgt zusammenfassen:

- Bei Verwendung eines physikalischen Zufallszahlengenerators wird grundsätzlich empfohlen, einen PTG.3-Generator einzusetzen. Dies gilt besonders für die Erzeugung von Ephemeralschlüsseln bei der Berechnung digitaler Signaturen und bei Diffie-Hellman-basierten Schlüsselaushandlungen. In Fällen, in denen für die Zufallszahlenerzeugung der Einsatz einer zertifizierten Kryptokomponente erforderlich ist, gilt diese Empfehlung nur bei Verfügbarkeit entsprechend zertifizierter Komponenten. Andernfalls kann ein PTG.3-Generator in der Regel durch eine zu den Anforderungen der Funktionalitätsklasse PTG.3 kompatible, in Software implementierte kryptographische Nachbearbeitung der Ausgabe eines PTG.2-Generators konstruiert werden.

- Für einige bestimmte kryptographische Anwendungen sind auch PTG.2-Generatoren ausreichend, beispielsweise bei der Erzeugung symmetrischer Sitzungsschlüssel oder bei der Ge-

nerierung eines Seeds für einen starken deterministischen Zufallszahlengenerator. Zufallszahlen aus PTG.2-konformen Zufallszahlengeneratoren besitzen hohe Entropie, können aber gewisse Schiefen und/oder Abhängigkeiten aufweisen. Sie können unter Umständen eingesetzt werden, wenn der daraus für einen Angreifer resultierende Vorteil nachweislich gering ist. Generell wird jedoch von der direkten Verwendung von PTG.2-Zufallszahlengeneratoren abgeraten. Dies gilt insbesondere für Anwendungen, in denen die Existenz selbst relativ geringfügiger Schiefen in der Verteilung der erzeugten Zufallszahlen zu ausnutzbaren Schwächen führen kann, etwa bei der Erzeugung von Noncen in DSA-ähnlichen Signaturverfahren.

- Beim Einsatz eines deterministischen Zufallszahlengenerators wird empfohlen, einen DRG.3- oder einen DRG.4-Generator einzusetzen, dessen Seed aus einer physikalischen Zufallsquelle der Klasse PTG.2 oder PTG.3 generiert wird. Falls keine derartige Zufallsquelle verfügbar ist, kann unter Umständen auch die Verwendung eines nicht-physikalischen, nicht-deterministischen Zufallszahlengenerators in Erwägung gezogen werden. So kann ein DRG.3-Generator beispielsweise auch mit einem NTG.1-Generator geseedet werden, für weitere Details sei auf die Abschnitte 8.3 und 8.5 verwiesen. DRG.3-Generatoren können unter geeigneten Randbedingungen auch mit einem DRG.4-Generator geseedet werden.

- Bei Softwareimplementierungen tritt häufig die Situation auf, dass nicht alle deterministische RNGs zur Seederzeugung direkten Zugriff auf einen echten RNG haben. In diesen Fällen kann die neu geschaffene Funktionalitätsklasse DRT.1 geeignet sein, die Bäume von deterministischen RNGs definiert.

- Grundsätzlich haben PTG.3-Generatoren und DRG.4-Generatoren verglichen mit PTG.2-Generatoren und DRG.3-Generatoren den Vorteil einer stärkeren Resistenz gegen Seitenkanalattacken und Fault-Angriffe. Im Falle eines PTG.3-Generators bedeutet der stetige Zufluss großer Mengen an Entropie in den inneren Zustand, dass mögliche Seitenkanalangriffe gegen die kryptographische Nachbearbeitung deutlich erschwert werden, da ein Angreifer Informationen über den inneren Zustand zu zwei aufeinanderfolgenden Zeitpunkten t und $t + 1$ nur sehr schwer zusammenführen kann.

- Neben dem Risiko einer langfristigen Kompromittierung durch Seitenkanal- und Fault Attacken kommt bei DRG.3-Zufallszahlengeneratoren ein gegenüber DRG.4- und PTG.3-Generatoren erhöhtes Restrisiko einer langfristig denkbaren kryptoanalytischen Kompromittierung hinzu, wenn der Zufallszahlengenerator große Mengen an langfristig schützenswertem Schlüsselmaterial aus einem einzelnen Seed-Wert erzeugt. Bei der Verwendung von PTG.3- oder DRG.4-Zufallszahlengeneratoren sind Seitenkanal- und Fault-Angriffe zwar ebenfalls relevant, führen aber im Vergleich nur zu der Kompromittierung relativ weniger erzeugter Zufallswerte.

- Im Allgemeinen wird für eine Systemsicherheit von n Bits eine Min-Entropie des DRNG-Seeds von n Bits benötigt.

- Sowohl für physikalische als auch für deterministische Zufallszahlengeneratoren sollte im jeweiligen Anwendungskontext eine Widerstandsfähigkeit gegen hohes Angriffspotential gezeigt werden.

8.1. Physikalische Zufallszahlengeneratoren

Physikalische Zufallszahlengeneratoren nutzen dedizierte Hardwaredesigns (üblicherweise eine elektronische Schaltung) oder physikalische Experimente, um „echten“ Zufall, das heißt unvorhersagbare Zufallszahlen, zu erzeugen. In der Regel wird dabei das unvorhersehbare Verhalten einfacher elektrischer Schaltungen genutzt, wie es durch verschiedene Formen von Rauschen

in den Schaltungen verursacht werden kann. Letzten Endes beruht die Entropie des Signals üblicherweise physikalisch auf Quanteneffekten oder auf der Verstärkung nicht kontrollierbarer beziehungsweise separat messbarer Umgebungseinflüsse in einem chaotischen System. Ein Angreifer sollte auch bei Kenntnis von Teilfolgen von Zufallszahlen sowie genauer Kenntnis des Zufallszahlengenerators einschließlich der physikalischen Umgebungsbedingungen zum Zeitpunkt der Erzeugung vorheriger oder nachfolgender Zufallszahlen nur einen vernachlässigbaren (im Idealfall gar keinen) Vorteil gegenüber blindem Raten der Zufallszahlen besitzen. Häufig ist eine deterministische Nachbearbeitung der „Rohzufallszahlen“ (üblicherweise digitalisierte Rauschsignale) notwendig, um etwaig vorhandene Schiefen oder Abhängigkeiten zu beseitigen.

Bei der Nutzung eines physikalischen Zufallszahlengenerators wird grundsätzlich empfohlen, einen PTG.3-Generator im Sinne der AIS 31 einzusetzen (vergleiche [37, Kapitel 3]). Dies trifft insbesondere auf Anwendungen zu, bei denen ein Angreifer zumindest prinzipiell Informationen über verschiedene Zufallszahlen zusammenführen kann.

Es ist möglich, einen PTG.3-Generator aus einem PTG.2-Generator zu konstruieren, indem die Ausgabe des PTG.2-Generators auf geeignete Weise kryptographisch nachbearbeitet wird. Diese Nachbearbeitung kann in der Regel in Software implementiert werden. Die genauen Anforderungen an die Nachbearbeitung finden sich in [37]. Vereinfacht dargestellt muss die Nachbearbeitung einen DRG.3-kompatiblen deterministischen Zufallszahlengenerator implementieren und es muss dem internen Zustand dieses Zufallszahlengenerators jederzeit mindestens soviel neue Entropie durch einen Zufallszahlengenerator der Klasse PTG.2 zugeführt werden, wie durch die kryptographische Anwendung verlangt wird.

Kurz zusammengefasst müssen PTG.2- beziehungsweise PTG.3-konforme Zufallszahlengeneratoren folgende Eigenschaften erfüllen:

- Die stochastischen Eigenschaften der Rohzufallszahlen lassen sich hinreichend genau durch ein stochastisches Modell beschreiben. Auf der Basis dieses stochastischen Modells kann die Entropie der ausgegebenen Zufallszahlen zuverlässig abgeschätzt werden.
- Der durchschnittliche Entropiezuwachs pro Ausgabezufallsbit liegt oberhalb einer gegebenen Mindestschranke (nahe bei 1). Der Hersteller kann entscheiden, ob diese Mindestschranke in Shannon-Entropie, in Min-Entropie oder in beidem definiert wird.
- Rohzufallszahlen werden im laufenden Betrieb einem Onlinetest unterzogen, um nicht akzeptable statistische Schwächen oder eine nicht tolerierbare Verringerung der Entropie in angemessener Zeit zu erkennen.
- Ein Totalausfall der Rauschquelle wird de facto sofort erkannt. Es dürfen keine Zufallszahlen ausgegeben werden, die nach einem Totalausfall der Rauschquelle erzeugt wurden.
- Die Feststellung eines Totalausfalls der Rauschquelle oder nicht akzeptabler statistischer Defekte der Zufallszahlen führt zu einem Rauschalarm. Auf einen Rauschalarm folgt eine definierte, geeignete Reaktion (zum Beispiel Stilllegen der Rauschquelle).
- (Nur PTG.3-konforme Zufallszahlengeneratoren): Eine (gegebenenfalls zusätzliche) starke kryptographische Nachbearbeitung stellt sicher, dass selbst bei einem unbemerkten Totalausfall der Rauschquelle noch das Sicherheitsniveau eines DRG.3-konformen deterministischen Zufallszahlengenerators vorliegt.

Hybride Zufallszahlengeneratoren vereinen Sicherheitseigenschaften von deterministischen und physikalischen Zufallszahlengeneratoren. Physikalische Zufallszahlengeneratoren der Funktionalitätsklasse PTG.3 besitzen neben einer starken Rauschquelle eine starke kryptographische Nachbearbeitung mit Gedächtnis. Eine typische Realisierung eines hybriden Zufallszahlengenerators besteht darin, dass Zufallszahlen eines PTG.3-konformen Zufallszahlengenerators in geeigneter Weise kryptographisch nachbearbeitet werden.

Entwicklung und sicherheitskritische Bewertung von physikalischen Zufallszahlengeneratoren setzen eine umfassende Erfahrung auf diesem Gebiet voraus. Es wird empfohlen, frühzeitig Experten auf diesem Gebiet zu Rate zu ziehen.

8.2. Deterministische Zufallszahlengeneratoren

Deterministische Zufallszahlengeneratoren (auch Pseudozufallszahlengeneratoren) können aus einem Zufallswert fester Länge, dem sogenannten *Seed*, eine pseudozufällige Bitfolge praktisch beliebiger Länge berechnen. In die Berechnung können auch öffentlich bekannte Parameter eingehen. Dazu wird zunächst der innere Zustand des Pseudozufallszahlengenerators mit dem Seed initialisiert. In jedem Schritt wird dann der innere Zustand erneuert, und es wird eine Zufallszahl (normalerweise eine Bitfolge fester Länge) aus dem inneren Zustand abgeleitet und ausgegeben. Hybride deterministische Zufallszahlengeneratoren erneuern den inneren Zustand von Zeit zu Zeit mit „echten“ Zufallswerten. Dies kann auf unterschiedliche Weise veranlasst werden, zum Beispiel regelmäßig oder auf Anfrage der Applikation. Der innere Zustand eines deterministischen Zufallszahlengenerators muss zuverlässig gegen Auslesen und Manipulation geschützt werden. Kommt ein deterministischer Zufallszahlengenerator zum Einsatz, dann wird empfohlen, einen DRT.1-, DRG.3- oder DRG.4-konformen Zufallszahlengenerator im Sinne der AIS 20 zu verwenden (vergleiche [37]).

Bei Verwendung von Zufallszahlengeneratoren der Klasse DRG.3 ist ein regelmäßiger Zufluss von frischer Entropie in den inneren Zustand wünschenswert, auch wenn dieser nicht ausreichend regelmäßig oder qualitativ hochwertig genug ist, um für die Gesamtkonstruktion DRG.4-Konformität zu erreichen. Vereinfacht gesagt, bedeutet DRG.3-Konformität:

- Es ist einem Angreifer praktisch nicht möglich, zu einer bekannten Teilfolge von Zufallszahlen Vorgänger oder Nachfolger von Zufallszahlen zu berechnen oder mit signifikant höherer Wahrscheinlichkeit zu erraten, als dies ohne Kenntnis dieser Teilfolge möglich wäre.
- Eine starke kryptographische Nachbearbeitung stellt sicher, dass selbst bei einem unbemerkten Totalausfall der Rauschquelle noch das Sicherheitsniveau eines DRG.3-konformen deterministischen Zufallszahlengenerators vorliegt.

Für eine DRG.4-Konformität kommt hinzu, dass selbst wenn ein Angreifer den aktuellen inneren Zustand kennt, es ihm praktisch nicht möglich ist, Zufallszahlen, die nach dem nächsten Reseeden oder dem nächsten Additional Input mit hoher Entropie erzeugt werden, zu berechnen oder mit signifikant höherer Wahrscheinlichkeit zu erraten, als dies ohne Kenntnis des inneren Zustands möglich wäre.[1] Auch im Hinblick auf Implementierungsangriffe weisen DRG.4-Generatoren gegenüber DRT.1- und DRG.3-konformen Zufallszahlengeneratoren gewisse Vorteile auf. Im Vergleich zu DRG.3-konformen Zufallszahlengeneratoren weisen DRT.1-konforme Zufallszahlengeneratoren aufgrund ihrer Baumstruktur weitere Sicherheitsrisiken auf (vergleiche [37, §170], die beachtet werden müssen.

8.3. Nicht-physikalische nicht-deterministische Zufallszahlengeneratoren

Für viele kryptographische Anwendungen, etwa aus dem Bereich des E-Businesses oder des E-Governments, steht kein physikalischer Zufallszahlengenerator zur Verfügung, da diese Anwendungen im Allgemeinen auf Computern ohne zertifizierte kryptographische Hardware ausgeführt

[1] Unter einer signifikant höheren Wahrscheinlichkeit wird hier eine Wahrscheinlichkeit verstanden, die mindestens über der Wahrscheinlichkeit liegt, die für das Seed-Update erzeugten echten Zufallswerte zu erraten. Für jedes Seed-Update müssen mindestens 120 Bits Min-Entropie erzeugt werden.

werden. Stattdessen werden in aller Regel nicht-physikalische nicht-deterministische Zufallszahlengeneratoren (NPTRNG) verwendet, entweder direkt oder zum Seeden/Reseeden eines starken deterministischen Zufallszahlengenerators.

Wie physikalische Zufallszahlengeneratoren erzeugen auch nicht-physikalische nicht-deterministische Zufallszahlengeneratoren „echte“ Zufallszahlen und setzen auf Sicherheit im informationstheoretischen Sinn durch hinreichend viel Entropie. Allerdings nutzen sie hierfür keine dedizierte Hardware, sondern Systemressourcen (Systemzeit, RAM-Inhalte usw.) und/oder Nutzerinteraktion (zum Beispiel Tastatureingaben oder Mausbewegungen). Nicht-physikalische nicht-deterministische Zufallszahlengeneratoren werden üblicherweise auf Systemen eingesetzt, die nicht speziell für kryptographische Anwendungen entwickelt wurden, zum Beispiel handelsübliche PCs, Laptops oder Smartphones.

Eine typische Vorgehensweise zur Erzeugung von Zufallszahlen mithilfe nicht-physikalischer nicht-deterministischer Zufallszahlengeneratoren ist die Folgende: Zunächst wird ein langer Bitstring von „zufälligen Daten“ (genauer: von nicht-deterministischen Daten) erzeugt, wobei die Entropie pro Bit normalerweise eher gering ist. Dieser Bitstring wird mit einem inneren Zustand vermischt und aus dem inneren Zustand werden anschließend Zufallszahlen errechnet und ausgegeben.

In der mathematisch-technischen Anlage [37] wird eine Funktionalitätsklasse für derartige Zufallszahlengeneratoren (NTG.1) definiert. Für NTG.1-Zufallszahlengeneratoren wird grob gesagt verlangt, dass die Menge der gesammelten Entropie im laufenden Betrieb zuverlässig geschätzt werden kann und dass die Ausgabedaten eine Min-Entropie von ≥ 0.98 Bit pro Ausgabebit aufweisen.

Dieses bedeutet unter anderem:

- Die Entropie des inneren Zustands wird nach heuristischen Regeln geschätzt. Wird eine Zufallszahl ausgegeben, wird der Entropiezähler entsprechend reduziert.
- Zufallszahlen dürfen nur ausgegeben werden, wenn der Wert des Entropiezählers groß genug ist.
- Eine starke kryptographische Nachbearbeitung stellt sicher, dass selbst bei einem unbemerkten Totalausfall der Rauschquelle noch das Sicherheitsniveau eines DRG.3-konformen deterministischen Zufallszahlengenerators vorliegt.

Es ist für NPTRNG von entscheidender Bedeutung, dass die durch den Zufallszahlengenerator verwendeten Entropiequellen nicht durch einen Angreifer im Sinne einer Entropiereduktion manipuliert werden können oder vorhersagbar werden, wenn der Angreifer über präzise Informationen zur Ausführungsumgebung verfügt. Diese Voraussetzung ist auch bei Verwendung eines an sich guten NPTRNG nicht selbstverständlich. Ein Beispiel für eine in dieser Hinsicht kritische Situation stellt die Verwendung von Virtualisierungslösungen dar [113]. In diesem Fall kann der Output eines NPTRNG im Extremfall vollständig vorhergesagt werden, wenn das System zweimal aus dem gleichen Systemabbild heraus gestartet wird und alle Entropiequellen des virtuellen Systems vom Wirtsrechner kontrolliert werden.

Falls ein NPTRNG als alleinige oder als wesentliche Zufallsquelle für ein System verwendet werden soll, das für die Verarbeitung sensibler Daten bestimmt ist, wird dringend empfohlen, einen Experten hinzuzuziehen.

8.4. Verschiedene Aspekte

Hybride Zufallszahlengeneratoren vereinen Sicherheitseigenschaften von deterministischen und physikalischen Zufallszahlengeneratoren. Die Sicherheit eines hybriden deterministischen Zufallszahlengenerators der Klasse DRG.4 beruht in erster Linie auf der Komplexität des deterministi-

schen Anteils, welcher der Klasse DRG.3 angehört. Während der Nutzung des Zufallszahlengenerators wird zudem immer wieder neuer Zufall hinzugefügt. Dies kann beispielsweise in regelmäßigen Abständen oder auf die Anforderung einer Applikation hin erfolgen.

Hybride physikalische Zufallszahlengeneratoren der Klasse PTG.3 besitzen neben einer starken Rauschquelle eine starke kryptographische Nachbearbeitung mit Gedächtnis. Im Vergleich zu PTG.2-konformen Zufallszahlengeneratoren bietet die Funktionalitätsklasse PTG.3 zudem den Vorteil, dass die Zufallszahlen weder Schiefen noch ausnutzbare Abhängigkeiten aufweisen. Insbesondere für Anwendungen, bei denen ein potenzieller Angreifer zumindest prinzipiell Informationen über viele Zufallszahlen kombinieren kann (zum Beispiel Ephemeralschlüssel), sollte ein physikalischer Zufallszahlengenerator der Funktionalitätsklasse PTG.3 verwendet werden.

Die Ableitung von Signaturschlüsseln, Ephemeralschlüsseln und Primzahlen (für RSA) oder ähnlichem aus den erzeugten Zufallszahlen hat mit geeigneten Algorithmen zu erfolgen (zu elliptischen Kurven vergleiche [40]). Vereinfacht gesagt, sollte einem potenziellen Angreifer so wenig Information über die abgeleiteten (geheim zu haltenden) Werte zur Verfügung stehen wie möglich. Im Idealfall treten alle Werte innerhalb des jeweilig zulässigen Wertebereichs mit der gleichen Wahrscheinlichkeit auf, und verschiedene Zufallszahlen sollten zumindest keine praktisch ausnutzbaren Zusammenhänge aufweisen. Ebenso wie Signaturalgorithmen kann auch die Erzeugung geheim zu haltender Signaturschlüssel, Ephemeralschlüssel und Primzahlen Ziel von Seitenkanalangriffen werden (siehe zum Beispiel [54, 41, 40]).

8.5. Seedgenerierung für deterministische Zufallszahlengeneratoren

Für die Initialisierung eines deterministischen Zufallszahlengenerators wird ein Seed mit hinreichend hoher Entropie benötigt. Dieser Seed sollte mit einem physikalischen Zufallszahlengenerator der Funktionalitätsklassen PTG.2 oder PTG.3 erzeugt werden. Auf PCs steht normalerweise kein physikalischer Zufallszahlengenerator zur Verfügung oder ein derartiger Zufallszahlengenerator wurde zumindest keiner gründlichen herstellerunabhängigen Zertifizierung unterzogen. In solchen Fällen bietet sich die Verwendung eines nicht-physikalischen nicht-deterministischen Zufallzahlengenerators an; empfohlen werden in dieser Technischen Richtlinie NTG.1-konforme Zufallszahlengeneratoren. Zurzeit gibt es keine NTG.1-zertifizierten Zufallszahlengeneratoren. Daher werden unten für die zwei wichtigsten PC-Betriebssysteme geeignete Verfahren zur Seedgenerierung angegeben.

Der Einsatz der in den beiden folgenden Unterabschnitten empfohlenen Verfahren zur Seedgenerierung kann aber nur dann als sicher eingestuft werden, wenn der Rechner unter vollständiger Kontrolle des Benutzers steht und keine Drittkomponenten direkten Zugriff auf den gesamten inneren Zustand des Rechners haben, wie es zum Beispiel der Fall sein kann, wenn das gesamte Betriebssystem in einer virtuellen Umgebung abläuft. Dies schließt insbesondere die Existenz von beispielsweise Viren oder Trojanern auf diesem Rechner aus. Benutzer sollten über diese Risiken entsprechend aufgeklärt werden.

8.5.1. GNU/Linux

Folgendes Verfahren wird zur Seedgenerierung unter dem Betriebssystem GNU/Linux empfohlen:

Auslesen von Daten aus der Gerätedatei `/dev/random`.

Tabelle 8.1: Empfohlenes Verfahren zur Seedgenerierung unter GNU/Linux.

Bemerkung 8.1 Der durch die Gerätedatei `/dev/random` gelieferte Zufall wird regelmäßig für aktuelle Kernelversionen vom BSI untersucht und bei einer Anwendung in PC-ähnlichen Systemen als geeignet bewertet [5]. Der Linux-RNG wird dabei durch die vorliegenden Technische Richtlinie als für allgemeine kryptographische Anwendungen in der Regel geeignet eingeschätzt, wenn die Anforderungen der Funktionalitätsklasse DRG.3, DRT.1 oder NTG.1 nach [37] erfüllt sind. Eine kryptographische Bewertung von `/dev/random` in verschiedenen Linux-Kernelversionen findet sich in der BSI-Studie [5]. Ein Abgleich der Sicherheitseigenschaften des Linux-Zufallszahlengenerators in verschiedenen Linux-Kernelversionen mit den Funktionalitätsklassen der AIS 20 beziehungsweise AIS 31 wird in den dort enthaltenen Kernelübersichten gegeben.

Bemerkung 8.2 Die Nutzung von `/dev/urandom` kann problematisch sein [58], da hierbei nicht geprüft wird, ob bei der Initialisierung des Zufallszahlengenerators eine für kryptographische Zwecke ausreichende Menge an Systemdaten gesammelt wurde. Hingegen blockiert `/dev/random` während des initialen Seedvorgangs, bis der interne Entropieschätzer eine festgelegte Schranke übersteigt.

8.5.2. Windows

Im Gegensatz zu GNU/Linux-Betriebssystemen gibt es für die Betriebssysteme der Windows-Familie derzeit keine vom BSI untersuchte Funktion, die hinreichend große Entropie gewährleistet. Zur Erzeugung sicherer Seeds sollten daher mehrere Entropiequellen in geeigneter Weise kombiniert werden. Beispielhaft wäre in Windows 10 oder 11 zur Erzeugung eines Seedwertes von mindestens 120 Bits Entropie die folgende Methode denkbar:

1.) Einlesen von 128 Bits Zufallsdaten in einen 128-Bit-Puffer S_1 aus der Windows-API-Funktion `BCryptGenRandom()`.

2.) Beziehen eines Bitstrings S_2 mit mindestens 128 Bits Entropie aus einer *anderen Quelle*. Hierbei kommen beispielsweise in Frage:

- Auswertung der Zeitabstände zwischen aufeinanderfolgenden Tastaturanschlägen des Benutzers: Wenn diese nachweislich mit einer Genauigkeit von einer Millisekunde erfasst werden können, können hierfür konservativ etwa drei Bits an Entropie pro Tastenanschlag angesetzt werden. Dabei ist im Hinblick auf eine Einschätzung der zeitlichen Auflösung der gemessenen Zeitabstände die gesamte Verarbeitungskette auf die gesammelte Entropie limitierende Faktoren zu untersuchen. Zum Beispiel ist es denkbar, dass die Genauigkeit der internen Uhr eine Auflösungsgrenze angibt, die Abtastfrequenz der Tastatur eine andere, und der Zeitabstand, innerhalb dessen der genutzte Systemtimer aktualisiert wird, eine weitere. Es wird empfohlen, zuvor in praktischen Tests die Verteilung der Tastaturanschlagszeiten zu messen und auf Auffälligkeiten zu untersuchen. Die Sequenz der zeitlichen Abstände einer hinreichend großen Anzahl von Tastaturereignissen kann dann in einen Binärstring B kodiert werden. Anschließend wird $S_2 := \mathrm{SHA256}(B)$ und die aufgenommenen Daten zu den Tastaturanschlagszeiten (und andere Daten, die in dem Prozess erhoben wurden) werden durch Überschreiben mit Nullen aus dem Arbeitsspeicher gelöscht.
- Durch den Benutzer ausgelöste Ereignisse: Die Zeitpunkte $T_1, T_2, T_3, T_4, T_5, T_6$ von sechs durch den Benutzer ausgelösten Ereignissen werden mittels der Windows-API-Funktion `QueryPerformanceCounter()` aufgenommen. Diese hat in der Regel mindestens eine Genauigkeit von der Größenordnung einer Mikrosekunde. Man kann unter den Voraussetzungen,

  (i) dass jedes T_i selbst dann nicht genauer als auf eine Sekunde geschätzt werden kann, wenn T_j dem Angreifer für alle $j \neq i$ bekannt ist,

(ii) dass, auch wenn der Angreifer T_j für alle $j \neq i$ kennt, der Wert von T_i nicht durch andere Erwägungen (zum Beispiel zur Polling-Frequenz der Tastatur) auf weniger als 2^{20} Möglichkeiten beschränkt werden kann, wenn irgendein Intervall von einer Sekunde Länge angegeben wird, das T_i enthält,

annehmen, dass der Bitstring $T := T_1||T_2||T_3||T_4||T_5||T_6$ aus Angreifersicht etwa 120 Bits Entropie enthält. Wie im vorherigen Beispiel setzt man $S_2 :=$ SHA256(T) und löscht T aus dem Arbeitsspeicher.

Es ist nicht immer ganz einfach, die Voraussetzungen an die Unabhängigkeit und Unvorhersagbarkeit der durch den Benutzer ausgelösten Ereignisse zu erfüllen. Das Problem hierbei ist, dass der Zeitpunkt, zu dem die Software vom Benutzer die Auslösung eines Ereignisses anfordert, möglicherweise eng vorhersagbar ist, wenn ein früherer Zeitpunkt bekannt ist. Der Zeitraum, der zwischen der Aufforderung zu einer Eingabe und der Benutzereingabe selbst vergeht, könnte ebenfalls genauer als im Sekundenbereich vorhersagbar sein. Die Erfüllung der Voraussetzungen und die Plausibilität derartiger Entropieeinschätzungen muss stets im konkret vorliegenden Einzelfall untersucht werden.

- Auf ähnliche Weise können auch Mausbewegungen des Benutzers zur Gewinnung von Entropie genutzt werden. Die in Mausbewegungen enthaltene Entropie kann nicht ohne Weiteres genau geschätzt werden. Es bedarf daher stets einer Einzelfallanalyse, bei der Art und Anzahl der aufgenommenen Ereignisse (Zeigerpositionen, gegebenenfalls zusätzlich Zeitmessungen) berücksichtigt werden, so dass sichergestellt werden kann, dass die erhobenen Messungen sich nicht verlustlos auf einen Datensatz von unter 120 Bits Größe komprimieren lassen. Man definiert dann S_2 wieder durch einen SHA2-Hash über die Mausereignisse.

3.) In allen Fällen kann anschließend ein Seed-Wert S für einen geeigneten Pseudozufallsgenerator abgeleitet werden, indem $S :=$ SHA256$(S_1||S_2)$ gesetzt wird. Idealerweise werden dabei so viele unabhängige Entropiequellen $S_1, \ldots, S_n$ wie möglich verwendet, um ein gewünschtes Sicherheitsniveau zu erreichen.

Bemerkung 8.3 Dem BSI liegen keine Erkenntnisse vor, die anzeigen, dass in dem obigen Beispiel ein 128-Bit-Wert bezogen aus `BCryptGenRandom()` nicht bereits näherungsweise 128 Bits Entropie enthält. Allerdings ist die exakte Funktionsweise von `BCryptGenRandom()` weder ausführlich in öffentlich zugänglichen Herstellerdokumenten beschrieben noch wurde die Funktion intensiv durch vom Hersteller unabhängige Parteien untersucht, wie es zum Beispiel für den Zufallszahlengenerator des Linux-Kernels der Fall ist. Daher ist die Kombination von Zufall aus `BCryptGenRandom()` mit der Ausgabe aus anderen Entropiequellen als grundsätzliche Vorsichtsmaßnahme empfehlenswert.

Anhang A.

Anwendungen kryptographischer Verfahren

Häufig müssen die in den vorangegangenen Kapiteln erläuterten Verfahren miteinander kombiniert werden, um den Schutz sensitiver Daten gewährleisten zu können. Insbesondere sollten zu übertragende sensitive Daten nicht nur verschlüsselt, sondern zusätzlich authentisiert werden, um es einem Empfänger zu ermöglichen, etwaige Veränderungen feststellen zu können.

Eine Schlüsseleinigung muss immer mit einer Instanzauthentisierung und einer Authentisierung aller während der Schlüsseleinigung übertragenen Nachrichten einher gehen, damit sich beide Parteien sicher sein können, mit wem sie kommunizieren. Andernfalls kann die Kommunikation durch eine sogenannte Man-in-the-Middle-Attacke kompromittiert werden. Je nach Anwendung können neben Man-in-the-Middle-Attacken auch andere Arten von Angriffen auf die Authentizität der Nachrichtenübermittlung (zum Beispiel Replay-Attacken) die Sicherheit eines informationsverarbeitenden Systems ohne Instanzauthentisierung oder ohne Datenauthentisierung gefährden. Daher werden in diesem Kapitel sowohl für eine Verschlüsselung mit Datenauthentisierung als auch zur authentisierten Schlüsselvereinbarung geeignete Verfahren angegeben.

A.1. Verschlüsselungsverfahren mit Datenauthentisierung

Grundsätzlich können bei der Kombination von Verschlüsselung und Datenauthentisierung alle in Kapitel 3 beziehungsweise Abschnitt 5.2 empfohlenen Verfahren eingesetzt werden. Dabei sind die folgenden beiden Aspekte einzuhalten:

- Verschlüsselte Daten müssen immer authentisiert übertragen werden. Zusätzlich ist es möglich, auch nicht vertrauliche Daten authentisiert, aber unverschlüsselt zu übertragen. Alle weiteren Daten derselben Datenübertragung sind *nicht* authentisiert.
- Verschlüsselungs- und Authentisierungsschlüssel müssen verschieden und dürfen nicht voneinander ableitbar sein.

Bemerkung A.1 Bei der authentisierten Übertragung von Daten wird die Nutzung eines AEAD-Verfahrens oder – im Falle verschlüsselter Daten – eines MACs im Encrypt-then-MAC-Modus empfohlen.

Bemerkung A.2 Es besteht die Möglichkeit, Verschlüsselungs- und Authentisierungsschlüssel aus einem gemeinsamen Schlüssel abzuleiten; dies steht nicht im Widerspruch zum zweiten Aspekt von oben. Empfohlene Verfahren sind in Abschnitt B.1 zusammengefasst.

Bemerkung A.3 Falls bei einer Übertragung verschlüsselter Daten zusätzlich das Sicherheitsziel der Nichtabstreitbarkeit des Klartextes angestrebt wird, sollte der Klartext durch eine digitale Signatur gesichert werden. In diesem Fall wird also zunächst der Klartext signiert, dann verschlüsselt, und schließlich wird die verschlüsselte Übertragung durch einen MAC vor Veränderung auf dem Übertragungsweg geschützt. Zusätzlich kann eine Signatur über das Chiffrat sinnvoll sein, wenn darüber hinaus die chiffrierte Nachricht unabstreitbar beziehungsweise nur der Absender (und nicht auch der legitime Empfänger) in der Lage sein soll, das Chiffrat zu ändern.

A.2. Authentisierte Schlüsselvereinbarung

Wie bereits erwähnt, muss eine Schlüsselvereinbarung immer mit einer Instanzauthentisierung kombiniert werden. Nach einigen allgemeinen Vorbemerkungen werden Verfahren angegeben, die entweder vollständig auf symmetrischen Algorithmen oder vollständig auf asymmetrischen Algorithmen basieren.

A.2.1. Vorbemerkungen

Ziele Ziel eines Verfahrens zum Schlüsselaustausch mit Instanzauthentisierung ist es, dass die beteiligten Parteien im Anschluss ein gemeinsames Geheimnis teilen und am Ende der Protokollausführung sicher sind, mit wem sie es teilen. Für die Ableitung symmetrischer Schlüssel für Verschlüsselungs- und Datenauthentisierungsverfahren aus diesem Geheimnis siehe Abschnitt B.1.

Voraussetzungen an die Umgebung Voraussetzung für einen authentisierten Schlüsselaustausch mit symmetrischen Verfahren ist die Existenz vorverteilter Geheimnisse. Bei asymmetrischen Verfahren wird in der Regel vorausgesetzt, dass eine Public-Key-Infrastruktur vorhanden ist, die in der Lage ist, zuverlässig Schlüssel an Identitäten zu binden und die Herkunft eines Schlüssels durch entsprechende Zertifikate zu beglaubigen. Es wird außerdem angenommen, dass die Wurzelzertifikate der PKI allen Teilnehmern auf zuverlässigem Wege bekanntgemacht worden sind und dass alle Teilnehmer zu jedem Zeitpunkt in der Lage sind, sämtliche relevanten Zertifikate auf Gültigkeit prüfen zu können.

Hinweise zur Umsetzung Bei der konkreten Umsetzung der vorgestellten Verfahren müssen die folgenden beiden Bedingungen erfüllt sein:

- Die für die Authentisierung benutzten Zufallswerte müssen bei jeder Durchführung des Protokolls mit großer Wahrscheinlichkeit verschieden sein. Dies kann zum Beispiel erreicht werden, indem jedes Mal ein Zufallswert bezüglich der Gleichverteilung auf $\{0,1\}^{120}$ gewählt wird.
- Die für die Schlüsselvereinbarung benutzten Zufallswerte müssen mindestens eine Entropie erreichen, die den gewünschten Schlüssellängen der auszuhandelnden Schlüssel entspricht.[1] Zusätzlich sollte jeder Teilnehmer an der Schlüsselvereinbarung mindestens 120 Bits Min-Entropie zu dem auszuhandelnden Schlüssel beitragen.

A.2.2. Symmetrische Verfahren

Grundsätzlich kann jedes Verfahren zur Instanzauthentisierung aus Abschnitt 6.1 mit jedem Verfahren zur Schlüsselvereinbarung aus Abschnitt 3.3 miteinander kombiniert werden. Die Kombination muss dabei so erfolgen, dass die ausgetauschten Schlüssel tatsächlich authentisiert sind und somit insbesondere Man-in-the-Middle-Attacken ausgeschlossen werden können. Für diese Anwendung wird folgendes Verfahren empfohlen:

Key Establishment Mechanism 5 aus [65].

Tabelle A.1: Empfohlenes symmetrisches Verfahren zur Schlüsselvereinbarung mit Instanzauthentisierung.

[1] Es wird an dieser Stelle davon ausgegangen, dass nur symmetrische Schlüssel ausgehandelt werden.

Bemerkung A.4 Als Verschlüsselungsverfahren können im Key Establishment Mechanism 5 aus [65] alle in dieser Technischen Richtlinie empfohlenen authentisierten Verschlüsselungsverfahren verwendet werden (siehe Abschnitt A.1).

A.2.3. Asymmetrische Verfahren

Analog wie für symmetrische Verfahren kann auch hier grundsätzlich jedes Verfahren zur Instanzauthentisierung aus Abschnitt 6.2 mit jedem Verfahren zur Schlüsselvereinbarung aus Kapitel 2 kombiniert werden. Um dabei jedoch Fehler in selbst konstruierten Protokollen auszuschließen, werden die in Tabelle A.2 aufgelisteten Verfahren zur Schlüsselvereinbarung mit Instanzauthentisierung basierend auf asymmetrischen Verfahren empfohlen.

Alle empfohlenen Verfahren setzen als Vorbedingung die Existenz eines Mechanismus zur manipulationssicheren Verteilung öffentlicher Schlüssel voraus. Dieser Mechanismus muss die folgenden Eigenschaften aufweisen:

- Der durch einen Nutzer erzeugte öffentliche Schlüssel muss zuverlässig an dessen Identität gebunden sein.
- Der zugehörige private Schlüssel sollte zuverlässig an die Identität des Nutzers gebunden sein (es sollte einem Nutzer nicht möglich sein, einen öffentlichen Schlüssel unter seiner Identität zu registrieren, zu dem er den zugehörigen privaten Schlüssel nicht nutzen kann).

Es gibt mehrere Möglichkeiten, dies zu erreichen. Eine manipulationssichere Schlüsselverteilung kann durch eine PKI realisiert werden. Die Anforderung, dass die Besitzer aller durch die PKI ausgestellten Zertifikate tatsächlich Nutzer der zugehörigen privaten Schlüssel sein sollen, kann durch die PKI überprüft werden, indem sie vor der Ausstellung des Zertifikats eines der in Abschnitt 6.2 beschriebenen Protokolle zur Instanzauthentisierung mit dem Antragsteller unter Verwendung seines öffentlichen Schlüssels durchführt.

Falls die PKI eine solche Prüfung nicht durchführt, wird empfohlen, die unten empfohlenen Verfahren um einen Schritt zur Schlüsselbestätigung zu ergänzen, in dem geprüft wird, dass beide Seiten das gleiche gemeinsame Geheimnis K ermittelt haben und in dem dieses Geheimnis an die Identitäten der beiden Parteien gebunden wird. Zur Schlüsselbestätigung wird das in [98, Abschnitt 5.6.2] beschriebene Verfahren empfohlen. Im zweiten empfohlenen Verfahren (KAS2-bilateral-confirmation nach [98]) ist dieser Schritt bereits enthalten.

- Elliptic Curve Key Agreement of ElGamal Type (ECKA-EG), siehe [27],
- Instanzauthentisierung mit RSA und Schlüsselvereinbarung mit RSA, siehe KAS2-bilateral-confirmation nach [98, Abschnitt 8.3.3.4],
- MTI(A0), siehe [69, Annex D.7].

Tabelle A.2: Empfohlene asymmetrische Verfahren zur Schlüsselvereinbarung mit Instanzauthentisierung.

Bemerkung A.5 Um konform mit der vorliegenden Technischen Richtlinie zu sein, muss bei der konkreten Umsetzung der Protokolle darauf geachtet werden, dass dabei lediglich die in diesem Dokument empfohlenen kryptographischen Komponenten zur Anwendung kommen.

Bemerkung A.6 Bei dem Verfahren ECKA-EG findet keine gegenseitige Authentisierung statt. Hier beweist eine Partei der anderen lediglich im Besitz eines privaten Schlüssels zu sein, und auch dies

geschieht nur implizit, und zwar durch den im Anschluss an die Ausführung des Protokolls vorhandenen Besitz des ausgehandelten Geheimnisses.

Anhang B.

Zusätzliche Funktionen und Algorithmen

Für einige der in dieser Technischen Richtlinie empfohlenen kryptographischen Verfahren werden zusätzliche Funktionen und Algorithmen benötigt, um beispielsweise Systemparameter zu generieren oder aus den von Zufallszahlengeneratoren oder Schlüsseleinigungsverfahren erhaltenen Werten symmetrische Schlüssel abzuleiten. Diese Funktionen und Algorithmen müssen sorgfältig gewählt werden, um das in dieser Technischen Richtlinie geforderte Sicherheitsniveau zu erreichen und kryptoanalytische Angriffe zu verhindern.

B.1. Schlüsselableitung

B.1.1. Schlüsselableitung nach Schlüsseleinigung

Nach einem Schlüsseleinigungsverfahren sind beide Parteien im Besitz eines gemeinsamen Geheimnisses. Häufig müssen aus diesem Geheimnis mehrere symmetrische Schlüssel, zum Beispiel für die Verschlüsselung und zur Datenauthentisierung (siehe auch Bemerkung A.2), abgeleitet werden, was mithilfe einer Schlüsselableitungsfunktion ermöglicht werden kann. Daneben können durch Verwendung einer Schlüsselableitungsfunktion auch folgende Ziele erreicht werden:

- Bindung von Schlüsselmaterial an Protokolldaten (zum Beispiel Sendername, Empfängername,...) durch Verwendung der Protokolldaten in der Schlüsselableitungsfunktion.
- Ableitung von Sitzungsschlüsseln oder Schlüsseln für verschiedene Zwecke aus einem Masterschlüssel auch in rein symmetrischen Kryptosystemen.
- Nachbearbeitung von Zufallsdaten zur Beseitigung statistischer Schiefen bei der Erzeugung kryptographischer Schlüssel.

Für sämtliche Anwendungen von Schlüsselableitungsfunktionen werden folgende Verfahren empfohlen:

- Two-Step KDF, siehe [100, Abschnitt 5],
- HKDF, siehe [74].

Tabelle B.1: Empfohlene Verfahren zur Schlüsselableitung.

Im Rahmen der vorliegenden Technischen Richtlinie wird empfohlen, im Falle von Two-Step KDF als MAC-Funktion HMAC oder AES-CMAC einzusetzen, siehe Abschnitt 5.2. Die Ausgabe des MAC-Algorithmus darf dabei nicht eingekürzt werden. Für Schlüssellängen von 128 Bit können gemäß [100, Tabelle 4,5] beide Verfahren gleichermaßen eingesetzt werden, für längere Schlüssel

wird ausschließlich die Verwendung von HMAC empfohlen. Wird die Schlüsselableitung mit HMAC durchgeführt, so entspricht das hier empfohlene Verfahren im Wesentlichen HKDF [74]. Der Unterschied zu HKDF besteht lediglich in der Reihenfolge der Nachrichtenbestandteile, was aus kryptographischer Sicht nicht relevant ist, möglicherweise aber zu Interoperabilitätsproblemen führen kann.

B.1.2. Passwort-basierte Schlüsselableitung

Bei der Passwort-basierten Schlüsselableitung wird ein kryptographischer Schlüssel (etwa für eine Festplattenverschlüsselung) direkt aus einem von einem Nutzer eingegebenen Passwort abgeleitet. Bei Verwendung nutzergenerierter Passwörter kann dadurch in der Regel mangels Entropie kein Sicherheitsniveau von 120 Bits erreicht werden.

Die vorliegende Technische Richtlinie empfiehlt in solchen Situationen die Ableitung des entsprechenden Schlüssels durch Anwendung eines MACs mit einem geheimen, nur für diesen Zweck genutzten Schlüssel auf das Passwort. Es wird empfohlen, den MAC auf kryptographisch sicherer, lokal im authentisierenden System vorhandener Hardware zu berechnen. Hierbei sollte ein CMAC oder ein HMAC mit mindestens 128 Bits Schlüssellänge genutzt werden und das Passwort sollte mit einem Salt-Wert von mindestens 32 Bits Länge kombiniert werden. Beim Scheitern einer Authentisierung oder einer Schlüsselableitung sollte die Hardware-Komponente verzögert reagieren, um lokale Brute-Force-Attacken auszuschließen. Die Qualität der Passwörter muss in diesem Fall den Anforderungen aus Abschnitt 6.3.1 entsprechen, wobei Offline-Attacken als ausgeschlossen betrachtet werden können.

Falls die Verwendung einer kryptographischen Sicherheitskomponente zur passwortbasierten Schlüsselableitung nicht möglich ist, sollte die Hashfunktion `Argon2id` [12] genutzt werden. Die Sicherheitsparameter von `Argon2id` und die Anforderungen an die Passwörter sind in Abhängigkeit von dem Anwendungsszenario mit einem Experten abzusprechen.

B.2. Erzeugung unvorhersagbarer Initialisierungsvektoren

Wie bereits in Abschnitt 3.1.2 erwähnt, müssen Initialisierungsvektoren für symmetrische Verschlüsselungsverfahren, die die Betriebsart Cipher Block Chaining Mode (CBC) einsetzen, unvorhersagbar sein. Dies bedeutet nicht, dass die Initialisierungsvektoren vertraulich behandelt werden müssen, sondern lediglich, dass ein möglicher Angreifer praktisch nicht in der Lage sein darf, zukünftig eingesetzte Initialisierungsvektoren zu erraten. Darüber hinaus darf der Angreifer auch nicht in der Lage sein, die Wahl der Initialisierungsvektoren zu beeinflussen.

In dieser Technischen Richtlinie werden die folgenden beiden Verfahren zur Erzeugung unvorhersagbarer Initialisierungsvektoren empfohlen, wobei n die Blockgröße der eingesetzten Blockchiffre bezeichnet:

Zufällige Initialisierungsvektoren:	Erzeugung einer zufälligen Bitfolge der Länge n mithilfe eines geeigneten Zufallszahlengenerators (siehe Kapitel 8) und Nutzung dieser Bitfolge als Initialisierungsvektor.
Verschlüsselte Initialisierungsvektoren:	Nutzung eines deterministischen Verfahrens zur Erzeugung von Prä-Initialisierungsvektoren (zum Beispiel einem Zähler). Verschlüsselung des Prä-Initialisierungsvektors mit der einzusetzenden Blockchiffre und einem vom eigentlichen Schlüssel verschiedenen Schlüssel (zum Beispiel Hashwert des einzusetzenden Verschlüsselungsschlüssels), und Nutzung des Chiffretexts als Initialisierungsvektor.

Tabelle B.2: Empfohlene Verfahren zur Erzeugung unvorhersagbarer Initialisierungsvektoren.

Bei der zweiten Methode muss darauf geachtet werden, dass sich die Prä-Initialisierungsvektoren während der Lebensdauer des Systems nicht wiederholen. Falls ein Zähler als Prä-Initialisierungsvektor verwendet wird, bedeutet dies, dass Zählerüberläufe während der gesamten Systemlebensdauer nicht auftreten dürfen.

B.3. Erzeugung von EC-Systemparametern

Die Sicherheit asymmetrischer Verfahren auf Basis elliptischer Kurven beruht auf der angenommenen Schwierigkeit der Berechnung diskreter Logarithmen in diesen Gruppen.

Zur Festlegung von EC-Systemparametern werden folgende Komponenten benötigt:

1.) Eine Primzahl p,

2.) Kurvenparameter $a, b \in \mathbb{F}_p$ mit $4a^3 + 27b^2 \neq 0$, die eine elliptische Kurve

$$E(\mathbb{F}_p) = \{(x, y) \in \mathbb{F}_p \times \mathbb{F}_p;\ y^2 = x^3 + ax + b\} \cup \{\mathcal{O}_E\}$$

festlegen, und

3.) ein Basispunkt P auf $E(\mathbb{F}_p)$.

Die *EC-Systemparameter* sind dann durch die Werte (p, a, b, P, q, i) gegeben, wobei $q := \mathrm{ord}(P)$ die Ordnung des Basispunktes P in $E(\mathbb{F}_p)$ bezeichnet, $p > 3$ und $i := \mathrm{Card}(E(\mathbb{F}_p))/q$ der sogenannte *Kofaktor* ist.

Nicht alle EC-Systemparameter sind für die in dieser Technischen Richtlinie empfohlenen asymmetrischen Verfahren, die auf elliptischen Kurven basieren, geeignet, dafür einige Parameterkonstellationen das diskrete Logarithmusproblem in den von diesen elliptischen Kurven generierten Gruppen effizient lösbar ist. Neben einer ausreichenden Bitlänge von q müssen zusätzlich die folgenden Bedingungen erfüllt sein, siehe [79] für weitere Informationen:

- Die Ordnung $q = \mathrm{ord}(P)$ des Basispunktes P ist eine von p verschiedene Primzahl,
- $p^r \neq 1 \bmod q$ für alle $1 \leq r \leq 10^4$, und
- die Klassenzahl der Hauptordnung des Quotientenkörpers des Endomorphismenrings von E ist größer als 10^7.

EC-Systemparameter, die die obigen Bedingungen erfüllen, werden auch als *kryptographisch stark* bezeichnet.

Bemerkung B.1 Es wird empfohlen, die EC-Systemparameter nicht selbst zu erzeugen, sondern stattdessen auf standardisierte Werte zurückzugreifen, die von einer vertrauenswürdigen Instanz zur Verfügung gestellt werden.

Die in Tabelle B.3 aufgelisteten Systemparameter werden empfohlen:

- brainpoolP256r1, siehe [79],
- brainpoolP320r1, siehe [79],
- brainpoolP384r1, siehe [79],
- brainpoolP512r1, siehe [79].

Tabelle B.3: Empfohlene EC-Systemparameter für asymmetrische Verfahren, die auf elliptischen Kurven basieren.

B.4. Generierung von Zufallszahlen für probabilistische asymmetrische Verfahren

In dieser Technischen Richtlinie werden mehrere asymmetrische Verfahren behandelt, die Zufallszahlen $k \in \{0, \ldots, q-1\}$ (zum Beispiel als Ephemeralschlüssel) benötigen, wobei q in der Regel keine 2er-Potenz ist. Bereits in den Bemerkungen 2.11, 2.13, 5.6 und 5.8 wurde darauf hingewiesen, dass k nach Möglichkeit (zumindest nahezu) gleichverteilt gewählt werden sollte. Hingegen erzeugen die in Kapitel 8 vorgestellten Zufallszahlengeneratoren gleichverteilte Zufallszahlen auf $\{0, 1, \ldots, 2^n - 1\}$ („zufällige n-Bitstrings“). Die Aufgabe besteht also darin, aus diesen Zufallszahlen (wenigstens nahezu) gleichverteilte Zufallszahlen auf $\{0, 1, \ldots, q-1\}$ herzuleiten.

In den Algorithmen B.1 und B.2 werden zwei Verfahren genannt, die dies ermöglichen, wobei $n \in \mathbb{N}$ so gewählt ist, dass $2^{n-1} \leq q \leq 2^n - 1$ gilt, das heißt q hat die Bitlänge n.

Algorithmus B.1: Verfahren 1 zur Berechnung von Zufallswerten auf $\{0, \ldots, q-1\}$.

Input: $q \in \mathbb{N}$ mit $2^{n-1} \leq q \leq 2^n - 1$
Output: $k \in \{0, 1, \ldots, q-1\}$ gleichverteilt

1: Wähle $k \in \{0, 1, \ldots, 2^n - 1\}$ gleichverteilt.
2: **while** $k \geq q$ **do**
3: Wähle $k \in \{0, 1, \ldots, 2^n - 1\}$ gleichverteilt.
4: **end while**

Algorithmus B.2: Verfahren 2 zur Berechnung von Zufallswerten auf $\{0, \ldots, q-1\}$.

Input: $q \in \mathbb{N}$ mit $2^{n-1} \leq q \leq 2^n - 1$
Output: $k \in \{0, 1, \ldots, q-1\}$ (nahezu) gleichverteilt.

1: Wähle $k' \in \{0, 1, \ldots, 2^{n+64} - 1\}$ gleichverteilt.
2: Setze $k = k' \bmod q$.

Bemerkung B.2 (i) Verfahren 1 in Algorithmus B.1 überführt eine Gleichverteilung auf $\{0, \ldots, 2^n - 1\}$ in eine Gleichverteilung auf $\{0, \ldots, q-1\}$, genauer liefert Verfahren 1 die bedingte Verteilung auf $\{0, \ldots, q-1\} \subset \{0, \ldots, 2^n - 1\}$. Hingegen erzeugt Verfahren 2 in

Algorithmus B.2 selbst für ideale Zufallszahlengeneratoren mit Werten in $\{0,\ldots,2^{n+64}-1\}$ keine (perfekte) Gleichverteilung auf $\{0,\ldots,q-1\}$. Die Abweichungen sind jedoch so gering, dass sie nach derzeitigem Wissensstand von einem Angreifer nicht ausgenutzt werden können.

(ii) Das zweite Verfahren besitzt den Vorteil, dass etwaige vorhandene Schiefen auf $\{0,\ldots,2^{n+64}-1\}$ in aller Regel reduziert werden. Für PTG.2-konforme Zufallszahlengeneratoren wurde daher nur dieses Verfahren empfohlen. Es sei jedoch darauf hingewiesen, dass die direkte Nutzung von PTG.2-Generatoren nicht mehr empfohlen wird.

(iii) Verfahren 1 besitzt den Nachteil, dass die Anzahl der Iterationen (und damit die Laufzeit) nicht konstant ist. Für manche Anwendungen kann es jedoch notwendig sein, eine obere Laufzeitschranke zu garantieren. An dieser Stelle sei angemerkt, dass die Wahrscheinlichkeit, dass eine auf $k \in \{0,1,\ldots,2^n-1\}$ gleichverteilte Zufallszahl kleiner als q ist, größer als $q/2^n \geq 2^{n-1}/2^n = 1/2$ ist.

B.5. Erzeugung von Primzahlen

B.5.1. Vorbemerkungen

Bei der Festlegung der Systemparameter für RSA-basierte asymmetrische Verfahren müssen zwei Primzahlen p und q gewählt werden. Für die Sicherheit der Verfahren ist es nötig, dass diese Primzahlen geheim gehalten werden. Dies setzt insbesondere voraus, dass p und q zufällig gewählt werden. Im Hinblick auf die Benutzerfreundlichkeit einer Anwendung, in der RSA-basierte Verfahren zum Einsatz kommen, ist es zudem wichtig, dass die Primzahlerzeugung effizient durchgeführt werden kann. Dabei ist zu beachten, dass proprietäre Geschwindigkeitsoptimierungen in der Schlüsselerzeugung zu signifikanten kryptographischen Schwächen führen können, siehe etwa [109]. Es wird daher dringend empfohlen, Verfahren einzusetzen, die öffentlich bekannt und hinsichtlich ihrer Sicherheit untersucht worden sind.

Routinen zur Erzeugung von zufälligen Primzahlen werden ferner für die Erzeugung von Systemparametern für ECC- beziehungsweise Körperarithmetik-basierte Kryptosysteme ohne spezielle Eigenschaften benötigt. Die Anforderungen an diese Primzahlen unterscheiden sich insofern von denen für das RSA-Verfahren, als dass Primzahlen nicht geheim gehalten werden müssen, sondern stattdessen eine *nachweisbare Zufälligkeit* ihrer Erzeugung von Relevanz sein kann. Weitere Einzelheiten und Hinweise zu diesem Thema finden sich in Abschnitt B.3.

B.5.2. Verfahren zur Erzeugung von Primzahlen

Zur Erzeugung zufälliger Primzahlen, die in einem vorgegebenen Intervall $[a,b]\cap\mathbb{N}$ liegen, sind drei Verfahren zulässig, die sich wie folgt kurz zusammenfassen lassen:

1.) Gleichverteilte Erzeugung zufälliger Primzahlen durch Verwerfungsmethode (englisch *rejection sampling*);

2.) Gleichverteilte Auswahl einer invertierbaren Restklasse r bezüglich $B\#$, wobei $B\#$ die *Primfakultät* von B ist, also das Produkt aller Primzahlen kleiner B, gefolgt von der Wahl einer Primzahl von geeigneter Größe mit Rest $r \bmod B\#$ durch Verwerfungsmethode;

3.) Erzeugung einer zufälligen Zahl s passender Größe, die zu $B\#$ teilerfremd ist, und Suche nach der nächsten Primzahl in der arithmetischen Folge, die durch $s, s+B\#, s+2\cdot B\#, \ldots$ gegeben ist.

Die ersten beiden Verfahren werden gleichermaßen empfohlen, das dritte Verfahren erzeugt gewisse statistische Schiefen in der Verteilung der erzeugten Primzahlen, die grundsätzlich unerwünscht sind. Es ist allerdings in der Praxis weit verbreitet (siehe etwa Table 1 in [119]) und es gibt zum aktuellen Zeitpunkt keine Hinweise darauf, dass sich die induzierten statistischen Schiefen für Angriffe nutzen lassen. Daher wird dieses Verfahren in der vorliegenden Technischen Richtlinie als Legacy-Verfahren akzeptiert.

Die folgenden Tabellen liefern eine genauere Beschreibung der drei durch diese Technische Richtlinie unterstützten Verfahren:

Algorithmus B.3: Empfohlenes Verfahren 1 zur Erzeugung von Primzahlen durch Verwerfungsmethode.

Input: Intervall $I := [a, b] \cap \mathbb{N}$
Output: $p \in I$ prim

1: Wähle $p \in I$ ungerade und gleichverteilt auf I.
2: **while** p zusammengesetzt **do**
3: Wähle $p \in I$ ungerade und gleichverteilt auf I.
4: **end while**

Algorithmus B.4: Empfohlenes Verfahren 2 zur Erzeugung von Primzahlen durch effizienzoptimierte Verwerfungsmethode.

Input: Intervall $I := [a, b] \cap \mathbb{N}$, $B \in \mathbb{N}$ mit $S := B\# \ll b - a$
Output: $p \in I$ prim

1: Wähle r in $(\mathbb{Z}/S)^*$ gleichverteilt (äquivalent: Wähle $r < S$ zufällig mit $\mathrm{ggT}(r, S) = 1$).
2: Wähle $k \in \mathbb{N}$ zufällig, so dass $p := kS + r \in I$ (äquivalent: Wähle k gleichverteilt auf $[\lceil (a - r)/S \rceil, \lfloor (b - r)/S \rfloor]$).
3: **while** p zusammengesetzt **do**
4: Wähle $k \in \mathbb{N}$ zufällig, so dass $p := kS + r \in I$ (äquivalent: Wähle k gleichverteilt auf $[\lceil (a - r)/S \rceil, \lfloor (b - r)/S \rfloor]$).
5: **end while**

Algorithmus B.5: Legacy-Verfahren zur Erzeugung von Primzahlen durch inkrementelle Suche.

Input: Intervall $I := [a, b] \cap \mathbb{N}$, $B \in \mathbb{N}$ mit $S := B\# \ll b - a$
Output: $p \in I$ prim

1: **repeat**
2: Wähle r in $(\mathbb{Z}/S)^*$ gleichverteilt (äquivalent: Wähle $r < S$ zufällig mit $\mathrm{ggT}(r, S) = 1$).
3: Wähle $k \in \mathbb{N}$ zufällig, so dass $p := kS + r \in I$ (äquivalent: Wähle k gleichverteilt auf $[\lceil (a - r)/S \rceil, \lfloor (b - r)/S \rfloor]$).
4: **while** p zusammengesetzt, $p \in I$ **do**
5: $p \leftarrow p + S$
6: **end while**
7: **until** p prim

Als Primzahltest in den oben beschriebenen Algorithmen kommt aus Effizienzgründen meist ein probabilistischer Primzahltest zum Einsatz. Im Rahmen dieser Technischen Richtlinie wird der folgende Algorithmus empfohlen:

Miller-Rabin, siehe [84, Algorithmus 4.24].

Tabelle B.4: Empfohlener probabilistischer Primzahltest.

Bemerkung B.3 (Miller-Rabin-Algorithmus) Der Miller-Rabin-Algorithmus benötigt neben der zu untersuchenden Zahl p einen Zufallswert $x \in \{2, 3, \ldots, p-2\}$, die sogenannte Basis. Ist x zufällig bezüglich der Gleichverteilung auf $\{2, 3, \ldots, p-2\}$ gewählt, so beträgt die Wahrscheinlichkeit, dass p zusammengesetzt ist, obwohl der Miller-Rabin-Algorithmus ausgibt, dass p eine Primzahl sei, höchstens $1/4$.

Worst Case: Um die Wahrscheinlichkeit, dass eine feste Zahl p mittels des Miller-Rabin-Algorithmus als Primzahl ausgegeben wird, obwohl sie zusammengesetzt ist, auf 2^{-120} zu beschränken, muss der Algorithmus 60-mal mit jeweils unabhängig voneinander bezüglich der Gleichverteilung gewählten Basen $x_1, \ldots, x_{60} \in \{2, 3, \ldots, p-2\}$ aufgerufen werden, siehe auch Abschnitt B.4 für empfohlene Verfahren zur Berechnung gleichverteilter Zufallszahlen aus $\{2, 3, \ldots, p-2\}$.

Average Case: Um eine zufällig bezüglich der Gleichverteilung gewählte ungerade Zahl $p \in [2^{b-1}, 2^b - 1]$ mit der gewünschten Sicherheit auf ihre Primzahleigenschaft zu testen, reichen bei weitem weniger Iterationen des Miller-Rabin-Algorithmus aus, als es die eben genannte Abschätzung nahelegen würde, vergleiche [46], [103, Appendix C] und [67, Annex A]. So werden für $b = 1536$ lediglich vier Iterationen benötigt, um bei einer verbleibenden Fehlerwahrscheinlichkeit von 2^{-128} auszuschließen, dass p zusammengesetzt ist, obwohl der Miller-Rabin-Algorithmus p als Primzahl erkennt [67]. Auch hierfür müssen die Basen unabhängig und zufällig bezüglich der Gleichverteilung aus $\{2, 3, \ldots, p-2\}$ gewählt werden. Die konkrete Anzahl an notwendigen Operationen hängt von der Bitlänge von p ab, da die Zahlen, für die die Abschätzungen des Worst-Case-Falles zutreffen, mit steigender Größe der Zahlen in ihrer Dichte deutlich abnehmen.

Optimierungen: Zur Optimierung der Laufzeit etwa von Algorithmus B.3 kann es hilfreich sein, zusammengesetzte Zahlen mit sehr kleinen Faktoren durch Probedivision oder Siebtechniken vor Anwendung des probabilistischen Primzahltests zu eliminieren. Ein solcher Vortest hat nur geringfügige Auswirkungen auf die Wahrscheinlichkeit, dass vom Test als Primzahl klassifizierte Zahlen doch zusammengesetzt sind. Die Empfehlungen zur erforderlichen Anzahl der Wiederholungen des Miller-Rabin-Tests gelten daher für auf solche Art optimierte Varianten des Verfahrens unverändert.

Sonstige Anmerkungen: Bei der Erzeugung von Primzahlen, die in besonders sicherheitskritischen Funktionen eines Kryptosystems Verwendung finden sollen oder deren Erzeugung wenig zeitkritisch ist, wird empfohlen, eine Verifikation der Primzahleigenschaft mit 60 Runden des Miller-Rabin-Tests durchzuführen, siehe auch [93, 103]. Dies betrifft zum Beispiel Primzahlen, die als dauerhafte Parameter eines kryptographischen Verfahrens einmal erzeugt und dann über einen längeren Zeitraum nicht gewechselt sowie unter Umständen von vielen Nutzern verwendet werden.

Zur Erzeugung der benötigten Zufallszahlen kann ein Zufallszahlengenerator der Funktionalitätsklassen PTG.3, DRG.4, DRG.3 oder NTG.1 genutzt werden. Bei Verwendung eines deterministischen Zufallszahlengenerators ist aus informationstheoretischer Sicht zwar keine gleichverteilte Primzahlerzeugung möglich, allerdings entsteht dadurch keine Sicherheitslücke: Ein Zufallszahlengenerator der Funktionalitätsklasse DRG.3 oder DRG.4 erzeugt unter

kryptographischen Standardannahmen Zufallszahlen mit einer Verteilung, die bei Verwendung klassischer Computer durch keinen bekannten Angriff mit realistischem praktischem Aufwand von einer idealen Verteilung unterschieden werden kann.

Es ist allerdings in diesem Zusammenhang zu beachten, dass das Sicherheitsniveau der erzeugten RSA-Moduln in diesem Fall möglicherweise durch das Sicherheitsniveau der Zufallszahlenerzeugung beschränkt wird. Dies wäre etwa der Fall, wenn ein Zufallszahlengenerator mit 120 Bits Sicherheitsniveau zur Erzeugung von RSA-Schlüsseln einer Länge von 4096 Bits genutzt würde.

Alternative Primzahltests: Die Auswahl eines Primzahltests ist aus kryptoanalytischer Sicht nicht sicherheitskritisch, solange der gewählte Test Primzahlen nicht fälschlicherweise als zusammengesetzt klassifiziert und solange die Wahrscheinlichkeit, dass zusammengesetzte Zahlen ihn bestehen, vernachlässigbar gering ist. Daher können andere Tests, für die diese Eigenschaften in der Literatur nachgewiesen worden sind, anstelle des Miller-Rabin-Tests eingesetzt werden, ohne dass die Konformität zu der vorliegenden Technischen Richtlinie verloren geht. Die Verwendung des sehr weit bekannten Miller-Rabin-Verfahrens ist allerdings unter anderem im Hinblick auf eine Überprüfung der Korrektheit einer Implementierung sowie auf eine Prüfung der Seitenkanalresistenz vorteilhaft.

B.5.3. Erzeugung von Primzahlpaaren

Um die Sicherheit von Schlüsselpaaren, für die die zugrundeliegenden RSA-Moduln durch Multiplikation zweier unabhängig voneinander mit einem der geeigneten Verfahren erzeugten Primzahlen berechnet wurden, zu gewährleisten, ist es wichtig, dass das Intervall $I := [a, b] \cap \mathbb{N}$ nicht zu klein ist. Wenn Schlüsselpaare erzeugt werden sollen, deren Modulus N eine vorher festgelegte Bitlänge n aufweist, bietet es sich an, $I = [\lceil \frac{2^{(n/2)}}{\sqrt{2}} \rceil, \lfloor 2^{(n/2)} \rfloor] \cap \mathbb{N}$ zu wählen. Eine andere Wahl von I ist zu dieser Technischen Richtlinie konform, wenn für p und q das gleiche Intervall I genutzt wird und $\mathrm{Card}(I) \geq 2^{-8} b$ ist.

B.5.4. Hinweise zur Sicherheit der empfohlenen Verfahren

Im Folgenden bezeichne π die Primzahlfunktion, also $\pi(x) := \mathrm{Card}(\{n \in \mathbb{N} : n \leq x, n \text{ prim}\})$. Nach dem Primzahlsatz ist $\pi(x)$ asymptotisch äquivalent zu $x/\ln(x)$, das heißt

$$\lim_{x \to \infty} \frac{\pi(x) \cdot \ln(x)}{x} = 1.$$

Die Sicherheit der hier empfohlenen Verfahren zur Primzahlerzeugung stützt sich auf die folgenden Beobachtungen:

- Alle drei Verfahren können jede Primzahl erzeugen, die in dem vorgegebenen Intervall enthalten ist, falls der zugrundeliegende Zufallszahlengenerator alle Kandidaten aus dem jeweils zulässigen Bereich erzeugen kann.

- Die ersten beiden Verfahren erzeugen Primzahlen, deren Verteilung bei Verwendung der empfohlenen Sicherheitsparameter praktisch nicht von einer Gleichverteilung unterschieden werden kann. Dies ist unmittelbar ersichtlich für das erste Verfahren; für das zweite Verfahren ergibt es sich heuristisch aus dem *Dirichlet'schen Primzahlsatz*: Die relative Häufigkeit von Primzahlen ist in allen invertierbaren Restklassen modulo S asymptotisch gleich, und die Restklasse modulo S der zu erzeugenden Primzahl wird gemäß der Gleichverteilung auf $(\mathbb{Z}/S)^*$ gewählt.

- Das im vorhergehenden Punkt genannte Argument für die Sicherheit des zweiten Verfahrens liefert strenggenommen keine Garantie dafür, dass für ein konkretes S und ein konkretes Intervall I die Häufigkeit von Primzahlen während der Suche tatsächlich nicht von der gewählten Restklasse $r \bmod S$ abhängig ist. In der Tat ist klar, dass diese asymptotische Aussage nicht gültig sein wird, wenn S sich der Größenordnung von $b - a$ annähert. Es ist aber anzunehmen, dass es keine wesentlichen Unterschiede bezüglich der Primzahldichte zwischen den verschiedenen Restklassen gibt, wenn die Anzahl der Primzahlen in den einzelnen Restklassen groß ist. Das Intervall I enthält $\pi(b) - \pi(a)$ Primzahlen, für jede Restklasse $\mathrm{mod} S$ werden daher $\frac{\pi(b)-\pi(a)}{\varphi(S)}$ Primzahlen erwartet. Für Zahlen der Größenordnung von etwa 1000 Bits kann dieser Erwartungswert mit einem geringen relativen Fehler auf $\frac{b\ln(a)-a\ln(b)}{\ln(a)\ln(b)\varphi(S)}$ geschätzt werden, solange $\varphi(S)$ klein im Vergleich zum Zähler des Bruches ist. Es wird empfohlen, S so zu wählen, dass $\frac{b\ln(a)-a\ln(b)}{\ln(a)\ln(b)\varphi(S)} \geq 2^{64}$ gilt.

- Die oben wiedergegebenen qualitativen Erwägungen sind ausreichend, um das zweite Verfahren als geeignet einzuschätzen. In der Literatur gibt es genauere Untersuchungen zu eng verwandten Mechanismen zur Primzahlerzeugung, siehe etwa [52].

- Das dritte Verfahren erzeugt Primzahlen, die nicht gleichverteilt sind, auch wenn die erzeugten Schiefen in der Verteilung der Primzahlen nach derzeitigem Kenntnisstand als praktisch durch einen Angreifer nicht ausnutzbar gelten. Die Wahrscheinlichkeit einer Primzahl p in dem Intervall I, durch dieses Verfahren ausgegeben zu werden, ist proportional zur Länge des primzahlfreien Abschnitts in der arithmetischen Folge $p - kS$, $p - (k-1)S, \ldots, p - S$, p, die durch p beendet wird. Da die Primzahldichte in diesen arithmetischen Folgen für große S tendenziell zunimmt, wird erwartet, dass dieser Effekt für $S = 2$ am stärksten ausgeprägt ist. Auch hier bedeutet er aber in der Praxis nur einen sehr begrenzten Entropieverlust. Man kann die Verteilungsschiefe nach oben begrenzen, indem die Suche abgebrochen und mit einem neuen Startwert wieder aufgenommen wird, falls nach einer angemessenen Zahl T von Schritten keine Primzahl gefunden wurde: In diesem Fall werden alle Primzahlen, die einer Lücke der Länge $\geq T$ folgen, mit gleicher Wahrscheinlichkeit ausgegeben.

Anhang C.

Protokolle für spezielle kryptographische Anwendungen

In diesem Kapitel werden Protokolle behandelt, die als Bausteine kryptographischer Lösungen benutzt werden können. In der aktuellen Version der vorliegenden Technischen Richtlinie betrifft dies die Protokolle SRTP (Secure Real-Time Transport Protocol) und MLS (Messaging Layer Security). Die entsprechenden Informationen für TLS [23], IPsec [24] und SSH [25] sind in den Teilen zwei bis vier der Technischen Richtlinie zu finden, siehe [23, 24, 25].

Im Allgemeinen hat die Verwendung etablierter Protokolle bei der Entwicklung kryptographischer Systeme den Vorteil, dass auf eine umfangreiche öffentliche Analyse zurückgegriffen werden kann. Eigenentwicklungen können demgegenüber leicht Schwächen enthalten, die für einen Entwickler nur schwer zu erkennen sind. Es wird daher empfohlen, wo immer es möglich ist, allgemein zugängliche, gegebenenfalls standardisierte und vielfach evaluierte Protokolle eigenen Protokollentwicklungen vorzuziehen.

C.1. SRTP

SRTP ist ein Protokoll, das das Audio- und Videoprotokoll RTP (Real-Time Transport Protocol) um Funktionen zur Sicherstellung von Vertraulichkeit und Integrität der übertragenen Nachrichten ergänzt. Es wird in RFC 3711 [8] definiert. SRTP muss mit einem Protokoll zum Schlüsselmanagement kombiniert werden, da es keine eigenen Mechanismen zur Aushandlung eines Kryptokontextes vorsieht.

Im Rahmen dieser Technischen Richtlinie werden folgende Spezifikationen bei der Verwendung von SRTP empfohlen:

- Als symmetrisches Verschlüsselungsverfahren mit kombiniertem Integritätsschutz wird AES im Galois/Counter Mode wie in [83] empfohlen.
- Als alternative Verschlüsselungsverfahren werden sowohl AES im Counter-Modus als auch im f8-Modus wie in [8] empfohlen. Als Integritätsschutz darf hier ein auf SHA1 basierender HMAC verwendet werden, da in [8] die Nutzung von Hashfunktionen der SHA2- oder SHA3-Familie nicht spezifiziert ist. Dieser HMAC darf im Kontext des Protokolls auf 80 Bits gekürzt werden.
- Als Schlüsselmanagementsystem sollte MIKEY [4] verwendet werden. Dabei werden die folgenden Schlüsselmanagementverfahren aus [4] empfohlen: DH-Schlüsselaustausch mit Authentisierung über PKI, RSA mit PKI sowie Pre-Shared-Keys. Generell sollten innerhalb von MIKEY und SRTP als Komponenten nur in dieser Richtlinie empfohlene kryptographische Verfahren verwendet werden.
- zRTP sollte nur eingesetzt werden, wenn es mit unverhältnismäßig hohem Aufwand verbunden ist, das Problem der Schlüsselverteilung durch ein Public-Key-Verfahren unter Verwendung einer PKI oder durch Vorverteilung geheimer Schlüssel zu lösen.

- Es wird dringend empfohlen, die in [8] vorgesehenen Mechanismen zum Replay- und Integritätsschutz in SRTP zu nutzen.

Bei Anwendungen zur sicheren Übertragung von Audio- und Videodaten in Echtzeit sollte besonders darauf geachtet werden, die Entstehung von Seitenkanälen, beispielsweise durch Datenübertragungsrate, die zeitliche Abfolge verschiedener Signale oder eine sonstige Verkehrsanalyse, zu minimieren. Andernfalls sind Angriffe wie die in [6] vorgestellten möglich.

C.2. MLS

Messaging Layer Security (MLS) ist ein Protokoll zum Austausch von Schlüsseln für die sichere Kommunikation im Kontext des (Group) Messagings. Es wurde entwickelt, um den Bedarf an Ende-zu-Ende-Verschlüsselung und Vertraulichkeit auch in goßen Gruppenchats zu decken und stellt eine Weiterentwicklung des Double Ratchet-Protokolls dar. MLS bietet einen Protokollrahmen, der auf einem asynchronen Schlüsselkapselungsverfahren für Baumstrukturen, dem sogenannten TreeKEM, basiert, welcher es den Mitgliedern einer Gruppe ermöglicht, gemeinsame Schlüssel abzuleiten und zu aktualisieren. Der damit verbundene Aufwand skaliert mit dem Logarithmus der Gruppengröße, was ein effizientes und asynchrones Verteilen von Gruppenschlüsseln mit Forward Secrecy and Post-Compromise Security erlaubt.

Das Protokoll erfordert die Spezifikation einer Cipher Suite, die aus der Protokollversion und der Menge der zu verwendenden kryptographischen Algorithmen besteht. Neben der Sicherheitsstufe (in Bits) müssen die folgenden kryptographischen Algorithmen angegeben werden:

- KEM-Algorithmus, der für HPKE bei Ratchet-Tree-Operationen verwendet wird,
- AEAD-Algorithmus, der für HPKE und Nachrichtenschutz verwendet wird,
- Hash-Algorithmus, der für HPKE und den MLS-Transkript-Hash verwendet wird,
- Signaturalgorithmus, der für die Authentifizierung von Nachrichten verwendet wird.

Für MLS 1.0 werden in dieser Technischen Richtlinie folgende Cipher Suiten empfohlen:

- MLS_128_DHKEMP256_AES128GCM_SHA256_P256, siehe [7, Abschnitt 17.1],
- MLS_256_DHKEMP384_AES256GCM_SHA384_P384, siehe [7, Abschnitt 17.1],
- MLS_256_DHKEMP521_AES256GCM_SHA512_P521, siehe [7, Abschnitt 17.1].

Tabelle C.1: Empfohlene Cipher Suiten für MLS 1.0.

Literaturverzeichnis

[1] M. Abdalla, M. Bellare, and P. Rogaway. DHIES: An encryption scheme based on the Diffie-Hellman problem. 2001. https://cseweb.ucsd.edu/~mihir/papers/dhies.pdf.

[2] M. R. Albrecht, , D. J. Bernstein, T. Chou, C. Cid, J. Gilcher, T. Lange, V. Maram, I. von Maurich, R. Misoczki, R. Niederhagen, K. G. Paterson, E. Persichetti, C. Peters, P. Schwabe, N. Sendrier, J. Szefer, C. J. Tjhai, M. Tomlinson, and W. Wang. Classic McEliece: conservative code-based cryptography: cryptosystem specification, 2022. https://classic.mceliece.org/mceliece-spec-20221023.pdf.

[3] E. Alkim, J. W. Bos, L. Ducas, P. Longa, I. Mironov, M. Naehrig, V. Nikolaenko, C. Peikert, A. Raghunathan, and D. Stebila. FrodoKEM: Learning With Errors Key Encapsulation. Einreichung zur dritten Runde des NIST PQC-Standardisierungswettbewerbs, 2021. https://frodokem.org/files/FrodoKEM-specification-20210604.pdf.

[4] J. Arkko, E. Carrara, F. Lindholm, M. Naslund, and K. Norrman. MIKEY: Multimedia Internet KEYing. RFC 3830, 2004. https://datatracker.ietf.org/doc/html/rfc3830.

[5] atsec information security GmbH. Documentation and Analysis of the Linux Random Number Generator. Dauerstudie im Auftrag des Bundesamtes für Sicherheit in der Informationstechnik. https://www.bsi.bund.de/LinuxRNG.

[6] L. Ballard, S. Coull, F. Monrose, G. Masson, and C. Wright. Spot me if you can: recovering spoken phrases in encrypted VoIP conversations. *IEEE Symposium on Security and Privacy*, 2008.

[7] R. Barnes, B. Beurdouche, R. Robert, J. Millican, E. Omara, and K. Cohn-Gordon. The Messaging Layer Security (MLS) Protocol. RFC 9420, 2023. https://datatracker.ietf.org/doc/html/rfc9420.

[8] M. Baugher, D. McGrew, M. Naslund, E. Carrara, and K. Norrman. The Secure Real-time Transport Protocol (SRTP). RFC 3711, 2004. https://datatracker.ietf.org/doc/html/rfc3711.

[9] M. Bellare, R. Canetti, and H. Krawczyk. Keying Hash Functions for Message Authentication. In *Advances in Cryptology – CRYPTO 1996*, volume 1109 of *LNCS*, pages 1–15. Springer, 1996.

[10] M. Bellare, R. Canetti, and H. Krawczyk. HMAC: Keyed-Hashing for Message Authentication. RFC 2104, 1997. https://datatracker.ietf.org/doc/html/rfc2104.

[11] D. J. Bernstein. Cost analysis of hash collisions: Will quantum computers make SHARCS obsolete? *SHARCS*, 2009.

[12] A. Biryukov, D. Dinu, D. Khovratovich, and S. Josefsson. Argon2 Memory-Hard Function for Password Hashing and Proof-of-Work Applications. RFC 9106, 2021. https://datatracker.ietf.org/doc/rfc9106/.

[13] A. Biryukov, O. Dunkelman, N. Keller, D. Khovratovich, and A. Shamir. Key Recovery Attacks of Practical Complexity on AES-256 Variants With Up to 10 Rounds. In *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 299–319, 2010.

[14] A. Biryukov and D. Khovratovich. Related-Key Cryptanalysis of the Full AES-192 and AES-256. In *Advances in Cryptology – ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 1–18, 2009.

[15] S. Blake-Wilson and A. Menezes. Unknown Key-Share Attacks on the Station-to-Station (STS) Protocol. In *Public Key Cryptography*, Volume 1560 of LNCS, pages 154–170. Springer, 1999.

[16] D. Bleichenbacher. Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS# 1. In *Advances in Cryptology – CRYPTO 1998*, volume 1462 of *LNCS*, pages 1–12. Springer, 1998.

[17] H. Böck, J. Somorovsky, and C. Young. Return Of Bleichenbacher's Oracle Threat (ROBOT). In *27th USENIX Security Symposium (USENIX Security 18)*, pages 817–849. USENIX Association, 2018.

[18] A. Bogdanov, D. Khovratovich, and C. Rechberger. Biclique cryptanalysis of the full AES. In *Advances in Cryptology – ASIACRYPT 2011*, Volume 7073 of LNCS, pages 344–371. Springer, 2011.

[19] D. Boneh and G. Durfee. Cryptanalysis of RSA with private key d less than $N^{0.292}$. *IEEE transactions on Information Theory*, 46(4):1339–1349, 2000.

[20] G. Brassard, P. Høyer, and A. Tapp. Quantum cryptanalysis of hash and claw-free functions. *ACM SIGACT News*, 28(2).

[21] D. R. L. Brown. Generic Groups, Collision Resistance, and ECDSA. *Designs, Codes and Cryptography*, 35(1):119–152, 2005.

[22] D. R. L. Brown and R. P. Gallant. The Static Diffie-Hellman Problem. Cryptology ePrint Archive, Report 2004/306, 2004. https://ia.cr/2004/306.

[23] Bundesamt für Sicherheit in der Informationstechnik. BSI TR-02102-2: Kryptographische Verfahren: Empfehlungen und Schlüssellängen, Teil 2 – Verwendung von Transport Layer Security (TLS). https://www.bsi.bund.de/TR-02102.

[24] Bundesamt für Sicherheit in der Informationstechnik. BSI TR-02102-3: Kryptographische Verfahren: Empfehlungen und Schlüssellängen, Teil 3 – Verwendung von IPsec. https://www.bsi.bund.de/TR-02102.

[25] Bundesamt für Sicherheit in der Informationstechnik. BSI TR-02102-4: Kryptographische Verfahren: Empfehlungen und Schlüssellängen, Teil 4 – Verwendung von Secure Shell (SSH). https://www.bsi.bund.de/TR-02102.

[26] Bundesamt für Sicherheit in der Informationstechnik. BSI TR-03110-2: Advanced Security Mechanisms for Machine Readable Travel Documents and eIDAS Token, Part 2 – Protocols for electronic IDentification, Authentication and trust Services (eIDAS). https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/TechGuidelines/TR03110/BSI_TR-03110_Part-2-V2_2.pdf.

[27] Bundesamt für Sicherheit in der Informationstechnik. BSI TR-03111: Elliptic Curve Cryptography. https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/TechGuidelines/TR03111/BSI-TR-03111_V-2-1_pdf.pdf.

[28] Bundesamt für Sicherheit in der Informationstechnik. BSI TR-03116-2: Kryptographische Vorgaben für Projekte der Bundesregierung, Teil 2 – Hoheitliche Ausweisdokumente. https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/TechnischeRichtlinien/TR03116/BSI-TR-03116-2.pdf.

[29] Bundesamt für Sicherheit in der Informationstechnik. BSI TR-03116-4: Kryptographische Vorgaben für Projekte der Bundesregierung, Teil 4 – Kommunikationsverfahren in Anwendungen. https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/TechnischeRichtlinien/TR03116/BSI-TR-03116-4.pdf.

[30] Bundesamt für Sicherheit in der Informationstechnik. BSI TR-03116: Kryptographische Vorgaben für Projekte der Bundesregierung. https://www.bsi.bund.de/DE/Themen/Unternehmen-und-Organisationen/Standards-und-Zertifizierung/Technische-Richtlinien/TR-nach-Thema-sortiert/tr03116/TR-03116_node.html.

[31] Bundesamt für Sicherheit in der Informationstechnik. BSI TR-03125 – TR-ESOR: Beweiswerterhaltung kryptographisch signierter Dokumente. https://www.bsi.bund.de/DE/Themen/Unternehmen-und-Organisationen/Standards-und-Zertifizierung/Technische-Richtlinien/TR-nach-Thema-sortiert/tr03125/TR-03125_node.html.

[32] Bundesamt für Sicherheit in der Informationstechnik. A proposal for: Functionality classes for random number generators. Version 2, 2011. https://www.bsi.bund.de/dok/ais-20-31-appx-2011.

[33] Bundesamt für Sicherheit in der Informationstechnik. AIS 20: Funktionalitätsklassen und Evaluationsmethodologie für deterministische Zufallszahlengeneratoren. Version 3, 2013. https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Zertifizierung/Interpretationen/AIS_20_pdf.pdf.

[34] Bundesamt für Sicherheit in der Informationstechnik. AIS 31: Funktionalitätsklassen und Evaluationsmethodologie für physikalische Zufallszahlengeneratoren. Version 3, 2013. https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Zertifizierung/Interpretationen/AIS_31_pdf.pdf.

[35] Bundesamt für Sicherheit in der Informationstechnik. Kryptografie quantensicher gestalten – Grundlagen, Entwicklungen, Empfehlungen. 2021. https://bsi.bund.de/dok/997274.

[36] Bundesamt für Sicherheit in der Informationstechnik. Entwicklungsstand Quantencomputer. Version 2.0, 2023. https://www.bsi.bund.de/qcstudie.

[37] Bundesamt für Sicherheit in der Informationstechnik. A Proposal for Functionality Classes for Random Number Generators. Version 3.0, 2024. https://www.bsi.bund.de/dok/ais-20-31-appx-2024.

[38] Bundesamt für Sicherheit in der Informationstechnik. Basics of Evaluating Side-Channel and Fault Attack Resistance - Part of AIS 46. Version 1, 2024. https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Zertifizierung/Interpretationen/AIS_46_SCA_basics.pdf.

[39] Bundesamt für Sicherheit in der Informationstechnik. Guidelines for Evaluating Machine-Learning based Side-Channel Attack Resistance - Part of AIS 46. Version 1, 2024. https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Zertifizierung/Interpretationen/AIS_46_AI_guide.pdf.

[40] Bundesamt für Sicherheit in der Informationstechnik. Guidelines for Evaluating Side-Channel and Fault Attack Resistance of Elliptic Curve Implementations - Part of AIS 46. Version 3.0, 2024. https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Zertifizierung/Interpretationen/AIS_46_ECCGuide_e_pdf.pdf.

[41] Bundesamt für Sicherheit in der Informationstechnik. Guidelines for Evaluating Side-Channel-Attack Resistance of RSA, DSA and Diffie-Hellman Key Exchange Implementations - Part of AIS 46. version 2024-01, 2024. https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Zertifizierung/Interpretationen/AIS_46_BSI_guidelines_SCA_RSA_V2024-01_e_pdf.pdf.

[42] Bundesamt für Sicherheit in der Informationstechnik, Ministerie van Binnenlandse Zaken en Koninkrijksrelaties, and Agence nationale de la sécurité des systèmes d'information. Securing Tomorrow, Today: Transitioning to Post-Quantum Cryptography. A joint statement from partners from 18 EU member states, 2024. https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Crypto/PQC-joint-statement.pdf.

[43] S. Chen, R. Wang, X. Wang, and K. Zhang. Side-Channel Leaks in Web Applications: A Reality Today, a Challenge Tomorrow. In *2010 IEEE Symposium on Security and Privacy*, pages 191–206. IEEE, 2010. https://ieeexplore.ieee.org/document/5504714.

[44] J. H. Cheon. Security analysis of the strong Diffie-Hellman problem. In *Advances in Cryptology – EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 1–11. Springer, 2006.

[45] J.-S. Coron, D. Naccache, and J. P. Stern. On the security of RSA padding. In *Advances in Cryptology – CRYPTO 1999*, volume 1666 of *LNCS*, pages 1–18. Springer, 1999.

[46] I. Damgård, P. Landrock, and C. Pomerance. Average Case Error Estimates for the Strong Probable Prime Test. *Mathematics of computation*, 61(203):177–194, 1993.

[47] W. Diffie, P. C. Van Oorschot, and M. J. Wiener. Authentication and Authenticated Key Exchanges. *Designs, Codes and Cryptography*, 2(2):107–125, 1992.

[48] ECRYPT – CSA. Algorithms, Key Size and Protocols Report, 2018. https://www.ecrypt.eu.org/csa/documents/D5.4-FinalAlgKeySizeProt.pdf.

[49] ECRYPT – II. Algorithms, Key Size and Protocols Report, 2012. https://www.ecrypt.eu.org/ecrypt2/documents/D.SPA.20.pdf.

[50] ETSI. ETSI TS 103 744: CYBER; Quantum-safe Hybrid Key Exchanges. V1.1.1, 2020. https://www.etsi.org/deliver/etsi_ts/103700_103799/103744/01.01.01_60/ts_103744v010101p.pdf.

[51] N. Ferguson. Authentication weaknesses in GCM. *Comments submitted to NIST Modes of Operation Process*, 2005. https://csrc.nist.gov/csrc/media/projects/block-cipher-techniques/documents/bcm/comments/cwc-gcm/ferguson2.pdf.

[52] P.-A. Fouque and M. Tibouchi. Close to uniform prime number generation with fewer random bits. *IEEE Transactions on Information Theory*, 65(2):1307–1317, 2019.

[53] French Cybersecurity Agency (ANSSI), Federal Office for Information Security (BSI), Netherlands National Communications Security Agency (NLNCSA), and Swedish National Communications Security Authority, Swedish Armed Forces. Position Paper on Quantum Key Distribution, 2024. https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Crypto/Quantum_Positionspapier.pdf.

[54] M. Gebhardt, G. Illies, and W. Schindler. A note on the practical value of single hash collisions for special file formats. In *Sicherheit 2006, Sicherheit – Schutz und Zuverlässigkeit*, pages 333–344. Gesellschaft für Informatik e.V., 2006. https://dl.gi.de/bitstream/handle/20.500.12116/24792/GI-Proceedings-77-41.pdf.

[55] D. Gillmor. Negotiated Finite Field Diffie-Hellman Ephemeral Parameters for Transport Layer Security (TLS). RFC 7919, 2016. https://datatracker.ietf.org/doc/html/rfc7919.

[56] L. K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, pages 212–219. Association for Computing Machinery, 1996.

[57] S. Gueron, A. Langley, and Y. Lindell. AES-GCM-SIV: Nonce Misuse-Resistant Authenticated Encryption. RFC 8452, 2019. https://www.rfc-editor.org/rfc/rfc8452.html.

[58] N. Heninger, Z. Durumeric, E. Wustrow, and J. A. Halderman. Mining Your Ps and Qs: Detection of Widespread Weak Keys in Network Devices. In *21st USENIX Security Symposium (USENIX Security 12)*, pages 205–220. USENIX Association, 2012. https://www.usenix.org/system/files/conference/usenixsecurity12/sec12-final228.pdf.

[59] R. Housley. Cryptographic Message Syntax (CMS). RFC 5652, 2009. https://datatracker.ietf.org/doc/html/rfc5652.

[60] A. Hülsing, D. Butin, S.-L. Gazdag, J. Rijneveld, and A. Mohaisen. XMSS: eXtended Merkle Signature Scheme. RFC 8391, 2018. https://datatracker.ietf.org/doc/html/rfc8391.

[61] G. Illies, M. Lochter, and O. Stein. Behördliche Vorgaben zu kryptografischen Algorithmen. *Datenschutz und Datensicherheit-DuD*, 35(11):807–811, 2011.

[62] International Organization for Standardization. ISO/IEC 18033-2:2006 Information technology – Security techniques – Encryption algorithms – Part 2: Asymmetric ciphers, 2006.

[63] International Organization for Standardization. ISO/IEC 14888-2:2008 Information technology – Security techniques – Digital signatures with appendix – Part 2: Integer factorization based mechanisms, 2008.

[64] International Organization for Standardization. ISO/IEC 9797-1:2011 Information technology – Security techniques – Message Authentication Codes (MACs) – Part 1: Mechanisms using a block cipher, 2011.

[65] International Organization for Standardization. ISO/IEC 11770-2:2018 Information security – Key management – Part 2: Mechanisms using symmetric techniques, 2018.

[66] International Organization for Standardization. ISO/IEC 14888-3:2018 Information technology – Digital signatures with appendix – Part 3: Discrete logarithm based mechanisms, 2018.

[67] International Organization for Standardization. ISO/IEC 18032:2020 Information security – Prime number generation, 2020.

[68] International Organization for Standardization. ISO/IEC 9796-2:2010 Information technology – Security techniques – Digital signature schemes giving message recovery – Part 2: Integer factorization based mechanisms, 2020.

[69] International Organization for Standardization. ISO/IEC 11770-3:2021 Information security – Key management – Part 3: Mechanisms using asymmetric techniques, 2021.

[70] T. Iwata, K. Ohashi, and K. Minematsu. Breaking and Repairing GCM Security Proofs. In *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *LNCS*, pages 31–49. Springer, 2012.

[71] K. Jang, A. Baksi, H. Kim, G. Song, H. Seo, and A. Chattopadhyay. Quantum analysis of AES. Cryptology ePrint Archive, Paper 2022/683, 2022. https://eprint.iacr.org/2022/683.

[72] S. Kent. IP Encapsulating Security Payload (ESP). RFC 4303, 2005. https://datatracker.ietf.org/doc/html/rfc4303.

[73] T. Kivinen and M. Kojo. More Modular Exponential (MODP) Diffie-Hellman Groups for Internet Key Exchange (IKE). RFC 3526, 2003. https://datatracker.ietf.org/doc/html/rfc3526.

[74] H. Krawczyk and P. Eronen. HMAC-based Extract-and-Expand Key Derivation Function (HKDF). RFC 5869, 2010. https://datatracker.ietf.org/doc/rfc5869/.

[75] A. K. Lenstra. Key lengths. In *Handbook of Information Security*, volume II, 2006.

[76] A. K. Lenstra and E. R. Verheul. Selecting Cryptographic Key Sizes. *Journal of Cryptology*, 14(4):255–293, 2001.

[77] G. Leurent and T. Peyrin. From Collisions to Chosen-Prefix Collisions Application to Full SHA-1. In *Advances in Cryptology – EUROCRYPT 2019*, volume 11478 of *LNCS*, pages 527–555. Springer, 2019.

[78] G. Leurent and T. Peyrin. SHA-1 is a Shambles: First Chosen-Prefix Collision on SHA-1 and Application to the PGP Web of Trust. In *29th USENIX Security Symposium (USENIX Security 20)*, pages 1839–1856. USENIX Association, 2020. https://www.usenix.org/system/files/sec20-leurent.pdf.

[79] M. Lochter and J. Merkle. Elliptic Curve Cryptography (ECC) Brainpool Standard Curves and Curve Generation. RFC 5639, 2010. https://datatracker.ietf.org/doc/html/rfc5639.

[80] S. Lucks and M. Daum. The Story of Alice and her Boss: Hash Functions and the Blind Passenger Attack. Presentation. https://www.cits.rub.de/imperia/md/content/magnus/rumpec05.pdf.

[81] V. G. Martínez, F. H. Álvarez, L. H. Encinas, and C. S. Ávila. A Comparison of the Standardized Versions of ECIES. In *Sixth International Conference on Information Assurance and Security, IAS 2010*, pages 1–4. IEEE, 2010.

[82] D. McGrew, M. Curcio, and S. Fluhrer. Leighton-Micali Hash-Based Signatures. RFC 8554, 2019. https://datatracker.ietf.org/doc/html/rfc8554.

[83] D. McGrew and K. Igoe. AES-GCM Authenticated Encryption in the Secure Real-Time Transport Protocol (SRTP). RFC 7714, 2015. https://datatracker.ietf.org/doc/html/rfc7714.

[84] A. J. Menezes, P. C. V. Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.

[85] R. C. Merkle. Secure Communications over Insecure Channels. *Communications of the ACM*, 21(4):294–299, 1978.

[86] K. Moriarty, B. Kaliski, J. Jonsson, and A. Rusch. PKCS #1: RSA Cryptography Specifications Version 2.2. RFC 8017, 2016. https://datatracker.ietf.org/doc/html/rfc8017.

[87] National Institute of Standards and Technology. Special Publication NIST SP 800-38A: Recommendation for Block Cipher Modes of Operation: Methods and Techniques, 2001. https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38a.pdf.

[88] National Institute of Standards and Technology. Federal Information Processing Standards FIPS PUB 197: Advanced Encryption Standard (AES), 2001, updated 2023. https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197-upd1.pdf.

[89] National Institute of Standards and Technology. Special Publication NIST SP 800-38C: Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality, 2007. https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-38c.pdf.

[90] National Institute of Standards and Technology. Special Publication NIST SP 800-38D: Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC, 2007. https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38d.pdf.

[91] National Institute of Standards and Technology. Special Publication NIST SP 800-38E: Recommendation for Block Cipher Modes of Operation: The XTS-AES Mode for Confidentiality on Storage Devices, 2010. https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-38e.pdf.

[92] National Institute of Standards and Technology. Special Publication NIST SP 800-38F: Recommendation for Block Cipher Modes of Operation: Methods for Key Wrapping, 2012. https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38f.pdf.

[93] National Institute of Standards and Technology. Federal Information Processing Standards FIPS PUB 186-4: Digital Signature Standard (DSS), 2013. https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf.

[94] National Institute of Standards and Technology. Federal Information Processing Standards FIPS PUB 180-4: Secure Hash Standard, 2015. https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf.

[95] National Institute of Standards and Technology. Federal Information Processing Standards FIPS PUB 202: SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions, 2015. https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.202.pdf.

[96] National Institute of Standards and Technology. Special Publication NIST SP 800-185: SHA-3 Derived Functions: cSHAKE, KMAC, TupleHash, and ParallelHash, 2016. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-185.pdf.

[97] National Institute of Standards and Technology. Special Publication NIST SP 800-38B: Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication, 2016. https://nvlpubs.nist.gov/nistpubs/specialpublications/nist.sp.800-38b.pdf.

[98] National Institute of Standards and Technology. Special Publication NIST SP 800-56B Revision 2: Recommendation for Pair-Wise Key Establishment Using Integer Factorization Cryptography, 2019. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-56Br2.pdf.

[99] National Institute of Standards and Technology. Special Publication NIST SP 800-208: Recommendation for Stateful Hash-Based Signature Schemes, 2020. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-208.pdf.

[100] National Institute of Standards and Technology. Special Publication NIST SP 800-56C Revision 2: Recommendation for Key-Derivation Methods in Key-Establishment Schemes, 2020. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-56Cr2.pdf.

[101] National Institute of Standards and Technology. Special Publication NIST SP 800-57 Part 1 Revision 5: Recommendation for Key Management: Part 1 – General, 2020. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r5.pdf.

[102] National Institute of Standards and Technology. Special Publication NIST SP 800-108r1-upd1: Recommendation for Key Derivation Using Pseudorandom Functions, 2022. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-108r1-upd1.pdf.

[103] National Institute of Standards and Technology. Federal Information Processing Standards FIPS PUB 186-5: Digital Signature Standard (DSS), 2023. https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-5.pdf.

[104] National Institute of Standards and Technology. Federal Information Processing Standards FIPS PUB 203: Module-Lattice-Based Key-Encapsulation Mechanism Standard, 2024. https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.203.pdf.

[105] National Institute of Standards and Technology. Federal Information Processing Standards FIPS PUB 204: Module-Lattice-Based Digital Signature Standard, 2024. https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.204.pdf.

[106] National Institute of Standards and Technology. Federal Information Processing Standards FIPS PUB 205: Stateless Hash-Based Digital Signature Standard, 2024. https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.205.pdf.

[107] National Institute of Standards and Technology. Special Publication NIST SP 800-227 ipd: Recommendations for Key Encapsulation Mechanisms, 2025. https://doi.org/10.6028/NIST.SP.800-227.ipd.

[108] National Security Agency . Announcing the Commercial National Security Algorithm Suite 2.0, 2022. https://media.defense.gov/2022/Sep/07/2003071834/-1/-1/0/CSA_CNSA_2.0_ALGORITHMS_.PDF.

[109] M. Nemec, M. Sys, P. Svenda, D. Klinec, and V. Matyas. The Return of Coppersmith's Attack: Practical Factorization of Widely Used RSA Moduli. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1631–1648. Association for Computing Machinery, 2017.

[110] P. Q. Nguyen and I. E. Shparlinski. The Insecurity of the Elliptic Curve Digital Signature Algorithm with Partially Known Nonces. *Designs, Codes and Cryptography*, 30(2):201–217, 2003.

[111] R. Perlner, J. Kelsey, and D. Cooper. Breaking category five SPHINCS+ with SHA-256. In *Post-Quantum Cryptography*, pages 501–522. Springer International Publishing, 2022.

[112] J.-F. Raymond and A. Stiglic. Security Issues in the Diffie-Hellman Key Agreement Protocol. 2000.

[113] T. Ristenpart and S. Yilek. When Good Randomness Goes Bad: Virtual Machine Reset Vulnerabilities and Hedging Deployed Cryptography. In *Network and Distributed System Security Symposium (NDSS) 2010*, 2010.

[114] A. Shamir. How to Share a Secret. *Communications of the ACM*, 22(11):612–613, 1979.

[115] P. W. Shor. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM J. Comput.*, 26(5):1484–1509, 1997.

[116] D. X. Song, D. A. Wagner, and X. Tian. Timing Analysis of Keystrokes and Timing Attacks on SSH. In *10th USENIX Security Symposium (USENIX Security 01)*. USENIX Association, 2001. https://www.usenix.org/legacy/events/sec2001/full_papers/song/song.pdf.

[117] M. Stevens, E. Bursztein, P. Karpman, A. Albertini, and Y. Markov. The First Collision for Full SHA-1. In *Advances in Cryptology – CRYPTO 2017*, volume 10401 of *LNCS*, pages 570–596. Springer, 2017.

[118] S. Vaudenay. Security Flaws Induced by CBC Padding – Applications to SSL, IPSEC, WTLS, ... In *Advances in Cryptology – EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 534–545. Springer, 2002.

[119] P. Švenda, M. Nemec, P. Sekan, R. Kvašňovskỳ, D. Formánek, D. Komárek, and V. Matyáš. The Million-Key Question – Investigating the Origins of RSA Public Keys. In *25th USENIX Security Symposium (USENIX Security 16)*, pages 893–910. USENIX Association, 2016. https://www.usenix.org/system/files/conference/usenixsecurity16/sec16_paper_svenda.pdf.

[120] M. Zhandry. A note on the quantum collision and set equality problems. *Quantum Info. Comput.*, 15(7–8):557–567, 2015.