Bundesamt
für Sicherheit in der
Informationstechnik

BSI – Technische Richtlinie

Bezeichnung:	**Kryptographische Verfahren: Empfehlungen und Schlüssellängen**
Kürzel:	BSI TR-02102-1
Version:	2014-01
Stand:	10.2.2014

Version	Datum	Änderungen
2013.01	21.12.2012	Grundlegende Überarbeitungen und Aktualisierungen
2013.02	09.01.2013	Editorielle Änderungen in den Abschnitten 3.4, 3.5, 5.2.1, 5.2.2 und 7.2.1, jeweils in *Schlüssellängen*; in Amerkung (b) zu Tabelle 1.2 und Tabelle 3.1 sowie in Bemerkung 4 in Kapitel 3
2014-01	10.2.2014	Anpassung des durch die vorliegende Richtlinie behandelten Vorhersagezeitraums. Sonstige geringfügige editorielle Änderungen an verschiedenen Stellen. Inhaltliche Überarbeitungen in Abschnitten 1.5 (siehe dort Spiegelpunkt 8) und 9.5. Auslagerung des Themas IPsec aus Annex C in die Technische Richtlinie TR02102-3.

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

Inhaltsverzeichnis

Tabellenverzeichnis

Notationen und Glossar

$\mathbb{F}_n$ Der Körper mit n Elementen. Wird auch als GF(n) bezeichnet.

$\mathbb{Z}_n$ Der Ring der Restklassen modulo n in $\mathbb{Z}$.

φ $\varphi : \mathbb{Z} \to \mathbb{Z}$ ist die eulersche Phi-Funktion. Sie läßt sich definieren über $\varphi(n) := \mathrm{Card}(\mathbb{Z}_n^*)$.

R^* Die Einheitengruppe des kommutativen Rings R.

Card Für eine endliche Menge M bezeichne Card(M) (auch $|M|$ geschrieben) die Anzahl ihrer Elemente.

Ceiling-Funktion Die Ceiling-Funktion $\lceil \cdot \rceil : \mathbb{R} \to \mathbb{Z}$ kann definiert werden über $\lceil x \rceil := \min\{z \in \mathbb{Z} : z \geq x\}$.

Floor-Funktion Die Floor-Funktion $\lfloor \cdot \rfloor : \mathbb{R} \to \mathbb{Z}$ ist definiert über $\lfloor x \rfloor := \max\{z \in \mathbb{Z} : z \leq x\}$.

A

AES Von NIST in FIPS 197 [39] standardisierte Blockchiffre mit einer Blockgröße von 128 Bit. 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.

Authentisierte Verschlüsselung Verschlüsselungsverfahren heißen *authentisiert*, wenn nicht nur die Vertraulichkeit, sondern auch die Integrität der zu verschlüsselnden Daten geschützt wird.

Authentisierung Ziel der sicheren Identifikation einer Person oder einer Maschine. Im gegebenen Kontext geht es dabei um Personen oder Maschinen, die Quelle oder Ziel einer Kommunikationsverbindung darstellen und die Authentisierung erfolgt unter Ausnutzung eines kryptographischen Geheimnisses.

Authentizität Authentizität einer Nachricht bedeutet, dass seit der Erstellung der Nachricht keine Veränderungen an der Nachricht vorgenommen wurden und dass keine falschen Informationen über den Absender der Nachricht beim Empfänger vorliegen. Im Sprachgebrauch der vorliegenden Technischen Richtlinie wird damit die Authentizität einer Nachricht durch Verfahren zur Datenauthentisierung nur dann zuverlässig geschützt, wenn der Zugriff auf die verwendeten Authentisierungsschlüssel durch ein Verfahren zur Instanzauthentisierung zuverlässig geschützt wird und durch kryptographische Mechanismen ein Wiedereinspielen von alten Nachrichten verhindert wird.

B

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.

C

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.

D

Datenauthentisierung Schutz der Integrität einer Nachricht durch kryptographische Verfahren.

Diffie-Hellman-Problem (DH) Gegeben sind g, g^a, g^b, wobei g ein Erzeuger der zyklischen Gruppe G ist. Zu berechnen ist g^{ab}. 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 (DL) Problem der Berechnung von d gegeben g^d in einer durch 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.

F

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

Festplattenverschlüsselung Festplattenverschlüsselung bezeichnet die vollständige Verschlüsselung eines Datenträgers. Das Ziel einer Festplattenverschlüsselung ist es, zu erreichen, dass aus dem verschlüsselten System zumindest in dessen abgeschaltetem Zustand keine vertraulichen Informationen ausgelesen werden können.

Forward Secrecy (für deterministische Zufallsgeneratoren) Im Zusammenhang mit deterministischen Zufallsgeneratoren bedeutet Forward Secrecy, 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 [57].

Forward Secrecy (für kryptographische Protokolle), deutsch fortgesetzte Geheimhaltung Sicherheitseigenschaft eines kryptographischen Protokolls, die besagt, dass eine Preisgabe kryptographischer Langzeitgeheimnisse es einem Angreifer nicht möglich macht, vergangene Sitzungen des Protokolls zu kompromittieren [33]. 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 [57] garantiert. Sollen darüber hinaus nicht durch einen Angreifer manipulierte *künftige* Sitzungen im Fall einer Kompromittierung aller langfristigen Geheimnisse geschützt bleiben, dann muss bei der Erzeugung der Ephemeralschlüssel ein Zufallsgenerator eingesetzt werden, der zusätzlich Enhanced Forward Secrecy [57] bietet.

Forward Security Mit den Konzepten der *Forward Secrecy* für kryptographische Protokolle und deterministische Zufallsgeneratoren verwandt (aber mit keinem der beiden identisch) ist der Begriff der *Forward Security* für Verschlüsselungs- und Signaturverfahren, siehe zum Beispiel den Übersichtsartikel [53].

G

GCM Galois Counter Mode, ein Betriebsmodus für Blockchiffren, der aus der Blockchiffre ein authentisiertes Verschlüsselungsverfahren konstruiert. Auch die Authentisierung nicht verschlüsselter Daten wird unterstützt.

GHASH Schlüsselabhängige Prüfsumme, die innerhalb des authentisierten Blockchiffren-Betriebsmodus GCM verwendet wird. Der Hash einer Nachricht $M = M_0 M_1 \ldots M_n$ (M_i 128-Bit-Blöcke) ergibt sich in GHASH als $\sum_{i=0}^{n} M_i H^{n-i+1}$, wo $H \in GF(2^{128})$ der Hash-Key ist. Zu beachten ist, dass GHASH für sich genommen nicht als kryptographische Hashfunktion oder als Message Authentication Code verwendet werden kann! Günstige Sicherheitseigenschaften ergeben sich nur für GCM (oder die Authentisierung von GCM ohne Verschlüsselung, also den GMAC-Authentisierungscode) als Ganzes.

Gleichverteilung Im Kontext dieser Technischen Richtlinie bedeutet *gleichverteilte* Erzeugung einer Zufallszahl aus einer Grundmenge M immer, 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 Eine Funktion $h : M \to N$, die effizient berechenbar ist und für die M deutlich größer ist als N. h heißt *kryptographische* Hashfunktion, wenn sie kollisionsresistent

und resistent gegen Berechnung erster und zweiter Urbilder ist. In der Regel wird in der vorliegenden Technischen Richtlinie der Begriff *Hashfunktion* eine kryptographische Hashfunktion meinen.

Hybride Verschlüsselung Verschlüsselungsverfahren, das Public-Key-Kryptographie zum Schlüsseltransport für ein symmetrisches Verschlüsselungsverfahren nutzt, welches wiederum zur Verschlüsselung der Nachricht verwendet wird.

I

Informationstheoretische Sicherheit Ein kryptographisches Verfahren heißt *informationstheoretisch sicher*, wenn jeder Angreifer bei dem Versuch, das System zu brechen, an *Mangel an Information* scheitert. In diesem Fall wird das Sicherheitsziel *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 8). In der Regel gibt es in Verfahren dieser Art *keinerlei* Sicherheitsgarantien, wenn die Einsatzvoraussetzungen des Verfahrens nicht exakt eingehalten werden.

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

Integrität Ziel 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 Schlüssel unbemerkt verändert werden kann.

K

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

M

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

Min-Entropie Die Min-Entropie einer diskreten Zufallsvariablen X (intuitiv eines Zufallsexperiments mit einer abzählbaren Menge möglicher Ergebnisse) ist definiert als $-\log_2(p)$, wo p die Wahrscheinlichkeit des wahrscheinlichsten Wertes für X bezeichnet.

P

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

Public-Key-Kryptographie Siehe Asymmetrische Kryptographie.

R

Related-Key-Attacke Angriff auf ein kryptographisches Verfahren, in dem der Angreifer Verschlüsselungen und gegebenenfalls Entschlüsselungen nicht nur unter dem eigentlich verwendeten Schlüssel K, sondern außerdem 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. In manchen Situationen können related-Key-Attacken dennoch praktisch relevant sein, zum Beispiel im Zusammenhang der Konstruktion einer kryptographischen Hashfunktion aus einer Blockchiffre.

S

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.

Secret Sharing Secret Sharing bezeichnet Verfahren zur Verteilung geheimer Daten (zum Beispiel eines kryptographischen Schlüssels) auf mehrere 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.

Seitenkanal-Angriff Angriff auf ein kryptographisches System, der die Ergebnisse von physikalischen Messungen am System (zum Beispiel Energieverbrauch, elektromagnetische Abstrahlung, Zeitverbrauch einer Operation) ausnutzt, um Einblick in sensible Daten zu erhalten. Seitenkanalangriffe sind für die praktische Sicherheit informationsverarbeitender Systeme von hoher Relevanz.

Shannon-Entropie Die Shannon-Entropie einer diskreten Zufallsvariablen X (intuitiv eines Zufallsexperiments mit einer abzählbaren Menge möglicher Ergebnisse) 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.

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 (z.B. 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

TDEA Triple DES.

U

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

V

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

Volume-Verschlüsselung Siehe Partitions-Verschlüsselung.

1. Einleitung

Das Bundesamt für Sicherheit in der Informationstechnik (BSI) gibt mit dieser Technischen Richtlinie eine Bewertung der Sicherheit und langfristige Orientierung für ausgewählte kryptographische Verfahren. Dabei wird allerdings kein Anspruch auf Vollständigkeit erhoben, d.h. nicht aufgeführte Verfahren werden vom BSI nicht unbedingt als unsicher beurteilt.

Umgekehrt wäre der Schluss falsch, dass kryptographische Systeme, die als Grundkomponenten nur in der vorliegenden Technischen Richtlinie empfohlene Verfahren verwenden, automatisch sicher wären: 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 2014 die Einführung neuer kryptographischer Systeme planen. Deshalb wird in diesem Dokument bewusst auf die Angabe kryptographischer Verfahren verzichtet, die zwar zum heutigen Zeitpunkt noch als sicher gelten, mittelfristig aber nicht mehr empfohlen werden können, da sie, wenn auch noch nicht ausnutzbare, so doch zumindest theoretische Schwächen zeigen.

Verschiedene andere vom BSI und von der Bundesnetzagentur herausgegebene Dokumente können bei der Entwicklung neuer kryptographischer Systeme ebenfalls eine Rolle spielen, etwa [2, 3, 4, 15, 23, 24, 25]. Für bestimmte Anwendungen sind die in diesen Dokumenten enthaltenen Vorgaben im Gegensatz zu den Empfehlungen der vorliegenden Richtlinie sogar bindend. Eine Diskussion der enthaltenen Regelungen (Stand 2011) findet sich in [44].

Die folgenden beiden Abschnitte beschreiben zunächst sowohl die Sicherheitsziele als auch die Auswahlkriterien der empfohlenen kryptographischen Verfahren. Weiter werden sehr allgemeine Hinweise zur konkreten Umsetzung der empfohlenen Verfahren gegeben.

In den Kapiteln 2 bis 9 werden die empfohlenen kryptographischen Verfahren für folgende Anwendungen aufgelistet

1. Symmetrische Verschlüsselung,
2. Asymmetrische Verschlüsselung,
3. Kryptographische Hashfunktionen,
4. Datenauthentisierung,
5. Instanzauthentisierung,
6. Schlüsselvereinbarung,
7. Secret Sharing und
8. Zufallszahlengeneratoren.

Geforderte Schlüssellängen und andere zu beachtende Nebenbedingungen werden in den jeweiligen Abschnitten angegeben.

Häufig müssen verschiedene kryptographische Algorithmen miteinander kombiniert werden, damit ein eingesetztes Verfahren die Sicherheitsanforderungen erfüllt. So ist es oft notwendig, vertrauliche Daten nicht nur zu verschlüsseln, der Empfänger muss sich auch sicher sein, von wem die Daten versendet wurden bzw. ob diese während der Übertragung manipuliert wurden. Die zu übertragenden Daten müssen also zusätzlich mit einem geeigneten Verfahren authentisiert werden.

Ein weiteres Beispiel sind Schlüsselvereinbarungsverfahren. Hier ist es wichtig zu wissen, mit wem die Schlüsselvereinbarung durchgeführt wird, um sogenannte Man-in-the-Middle-Attacken und Unknown-Key-Share-Attacken [14] ausschließen zu können. Dies geschieht durch Verfahren, die Schlüsselvereinbarung und Instanzauthentisierung kombinieren.

Deshalb werden in Anhang A für diese beiden Einsatzszenarien entsprechende Verfahren angegeben, die durch Kombination der in den Kapiteln 2 bis 9 aufgelisteten Verfahren konstruiert werden und das in dieser Technischen Richtlinie geforderte Sicherheitsniveau erfüllen.

Zusätzlich werden im Anhang B häufig verwendete Algorithmen empfohlen, die zum Beispiel zur Erzeugung von Primzahlen und anderen Systemparametern für asymmetrische Verfahren, zur Schlüsselableitung für symmetrische Verfahren usw. benötigt werden.

In Anhang C werden Empfehlungen ausgesprochen zur Verwendung einzelner kryptographischer Protokolle, in der gegenwärtigen Version dieser Richtlinie zu SRTP und SSH. Entsprechende Empfehlungen zur Verwendung von TLS finden sich in der Technischen Richtlinie TR02102-2 [20], zu IPsec in der Technischen Richtlinie TR02102-3 [21].

Es ist vorgesehen, die in dieser Technischen Richtlinie ausgesprochenen Empfehlungen jährlich zu überprüfen und bei Bedarf anzupassen.

1.1. Sicherheitsziele und Auswahlkriterien

Die Sicherheit kryptographischer Verfahren hängt primär von der Stärke der zugrunde liegenden Algorithmen ab. Aus diesem Grund werden hier nur Verfahren empfohlen, die aufgrund der heute vorliegenden Ergebnisse langjähriger Diskussionen und Analysen entsprechend eingeschätzt werden können. Weitere Faktoren für die Sicherheit sind die konkreten Implementierungen der Algorithmen und die Zuverlässigkeit eventueller Hintergrundsysteme, wie zum Beispiel benötigte Public Key Infrastrukturen für den sicheren Austausch von Zertifikaten. Die Umsetzung konkreter Implementierungen wird hier aber genauso wenig betrachtet wie eventuell auftretende patentrechtliche Probleme. **Zwar wurde bei der Auswahl der Verfahren darauf geachtet, dass die Algorithmen frei von Patenten sind, garantieren kann das BSI dies aber nicht. Ebenso finden sich in dieser Technischen Richtlinie einzelne Hinweise auf mögliche Probleme bei der Implementierung kryptographischer Verfahren, diese sind aber nicht als erschöpfende Liste möglicher solcher Probleme zu verstehen.**

Insgesamt erreichen alle in dieser Technischen Richtlinie angegebenen kryptographischen Verfahren mit den in den einzelnen Abschnitten geforderten Parametern ein Sicherheitsniveau von mindestens 100 Bit.

Die in dieser Technischen Richtlinie zur Verwendung in neuen kryptographischen Systemen empfohlenen Bitlängen richten sich nach diesem Minimalniveau aber nur insoweit, als dieses für kein empfohlenes Verfahren unterschritten wird. Die effektive Stärke der in der vorliegenden Richtlinie empfohlenen Verfahren ist in vielen Fällen höher als 100 Bit. Damit wird ein gewisser Sicherheitsspielraum gegenüber möglichen künftigen Fortschritten in der Kryptoanalyse geschaffen.

Im Umkehrschluss gilt, wie bereits in der Einleitung festgestellt wurde, auch 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 das in diesem Dokument minimal geforderte Sicherheitsniveau von 100 Bit erfüllt ist.

Symmetrische Verfahren		**asymmetrische Verfahren**		
Ideale Blockchiffre	Idealer MAC	RSA	DSA	ECDSA
100	100	1900	1900	200

Tabelle 1.1.: Beispiele für Schlüssellängen für ein Sicherheitsniveau von mindestens 100 Bit

Die *empfohlenen* Schlüssellängen verschiedener Typen kryptographischer Primitive werden in Tabelle 1.2 zusammengefasst.

Tabelle 1.2.: Empfohlene Schlüssellängen für verschiedene kryptographische Verfahren

Blockchiffre	MAC	RSA	DH F_p	DH (elliptische Kurve)	ECDSA
128	128	2000 [b]	2000 [a b]	224[a]	224[a]

[a] Für einen Einsatzzeitraum nach 2015 wird für den Erzeuger der verwendeten zyklischen Gruppe eine Ordnung $\geq 2^{250}$ empfohlen. Im Fall elliptischer Kurven sind damit bei der Wahl einer 256-Bit-Kurve kleine Kofaktoren erlaubt.

[b] Für einen Einsatzzeitraum nach 2015 kann es sinnvoll sein, eine Schlüssellänge von 3000 Bit zu nutzen, um ein gleichartiges Sicherheitsniveau für alle asymmetrischen Verfahren zu erreichen. Jede Schlüssellänge von ≥ 2000 Bits bleibt aber bis 2020 *konform* zu der vorliegenden Richtlinie; es handelt sich dabei um die in der vorliegenden Richtlinie *empfohlene* Mindest-Schlüssellänge für RSA, DLIES und DSA. Genauere Informationen finden sich in Bemerkung 4, Kapitel 3.

Im Fall von Message Authentication Codes ist die Länge des Digest-Outputs neben der Schlüssellänge ein wichtiger Sicherheitsparameter. Für einen MAC, für den ohne Kenntnis des Schlüssels eine Unterscheidung des MAC-Outputs von einer Zufallsfunktion mit entsprechender Digest-Länge bei bekanntem MAC zu $\ll 2^{64}$ vom Angreifer gewählten Nachrichten nach aktuellem Kenntnisstand nicht mit einem Aufwand $< 2^{100}$ MAC-Berechnungen möglich ist, wird in vielen Anwendungen eine Digest-Länge von 64 Bit ausreichend sein. Realistische Angreifer können unter diesen Voraussetzungen einzelne Nachrichten nicht mit einer Wahrscheinlichkeit wesentlich über 2^{-64} fälschen oder manipulieren.
Im Fall von Blockchiffren ist auch 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 ab vom verwendeten Betriebsmodus. In der vorliegenden Technischen Richtlinie werden keine Blockchiffren mit unter 128 Bit Blockbreite empfohlen.
Ein wichtiger Typ kryptographischer Primitive, die überhaupt keine geheimen Daten verarbeiten, sind überdies *kryptographische Hashfunktionen*. Hier ist die Länge des zurückgegebenen Digest-Wertes der wichtigste Sicherheitsparameter und sollte für allgemeine Anwendungen mindestens 200 Bit betragen, damit das in dieser Richtlinie minimal geforderte Sicherheitsniveau erreicht wird. Die in dieser Technischen Richtlinie empfohlenen Hashfunktionen haben eine Mindesthashlänge von 224 Bits. Für einen Einsatzzeitraum nach 2015 wird die Verwendung von Hashfunktionen mit einer Mindesthashlänge von 256 Bits empfohlen. Auf Abweichungen von dieser Regel für besondere Anwendungen wird in der vorliegenden Technischen Richtlinie an gegebener Stelle eingegangen werden.
Schlüsseltauschverfahren auf Diffie-Hellman-Basis sind in Tabellen 1.1 und 1.2 entsprechend zu DSA/ECDSA einzugruppieren.

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 Hashfunktionen 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 [58, 59].

Unter der Annahme, dass es in den nächsten zehn Jahren zu keiner kryptographisch relevanten Anwendung von Quantencomputern kommt, lässt sich die Leistungsfähigkeit von Rechnern über einen Zeitraum von 10 Jahren relativ gut vorhersagen. Dies gilt leider nicht für den wissenschaftlichen Fortschritt im Hinblick auf kryptoanalytische Verfahren. Jede Vorhersage über einen Zeitraum von 6-7 Jahren hinaus ist schwierig, insbesondere bei asymmetrischen Verfahren, und selbst **für diesen Zeitraum von 6-7 Jahren können sich die Prognosen aufgrund unvorhersehbarer Entwicklungen als falsch erweisen.**

Die Angaben dieser Technischen Richtlinie werden daher also nur beschränkt auf einen Zeitraum bis Ende 2020 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. Als unmittelbare Konsequenzen ergeben sich:

- Die Übertragung und die Speicherung vertraulicher Daten sollte auf das notwendige Maß beschränkt werden. Dies betrifft nicht nur Klartexte, sondern zum Beispiel in besonderem Maße auch die Vermeidung einer Speicherung von Sitzungsschlüsseln auf jeglichen nichtflüchtigen Medien und 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.

- Für Daten, deren Vertraulichkeit langfristig gesichert bleiben soll, ist zu empfehlen, von vornherein für die Verschlüsselung der Übertragung über allgemein zugängliche Kanäle, wie das Internet, aus den in dieser Technischen Richtlinie empfohlenen Verfahren möglichst starke 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* den besten bekannten Angriffen gegenüber als der AES-128 [12] – sollte nach Möglichkeit bereits hier der Rat eines Experten eingeholt werden.

- Im Hinblick auf die Auswahl kryptographischer Komponenten für eine neue Anwendung ist dabei grundsätzlich zu berücksichtigen, dass das Gesamtsystem in der Regel nicht stärker sein wird als die schwächste Komponente. Wird daher ein Sicherheitsniveau von beispielsweise 128 Bit für das Gesamtsystem angestrebt, dann 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 wie im letzten Punkt angesprochen 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 in Software-Implementierungen der hier vorgestellten kryptographischen Verfahren dem 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 schließt dies Abhängigkeiten der Lösung von Eigenschaften der verwendeten Hardware mit ein.

Fokus des vorliegenden Dokuments Die Sicherheitsbewertung der in diesem Dokument empfohlenen kryptographischen Verfahren erfolgt ohne Berücksichtigung der Einsatzbedingungen. Für konkrete Szenarien können sich andere Sicherheitsanforderungen ergeben. Es ist möglich, dass dies Anforderungen sind, denen die in dieser Richtlinie empfohlenen Verfahren nicht gerecht werden. Beispiele hierfür bestehen etwa in der Verschlüsselung von Datenträgern, in der verschlüsselten Speicherung und Verarbeitung von Daten auf durch externe Anbieter betriebenen Systemen („Cloud Computing“ beziehungsweise „Cloud Storage“) oder in kryptographischen Anwendungen auf Geräten mit extrem geringen rechentechnischen Ressourcen („Lightweight Cryptography“). Hinweise zu einigen solchen Situationen finden sich in Abschnitt 1.5.

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 Folgend sollen in Stichpunkten einige Grundsätze wiedergegeben werden, deren Beachtung bei der Entwicklung kryptographischer Systeme generell empfohlen wird.

- Es wird empfohlen, schon während früher Phasen der Planung von Systemen, für die kryptographische Komponenten benötigt werden, die Zusammenarbeit mit Experten auf kryptographischem Gebiet zu suchen.
- Die in dieser Technischen Richtlinie aufgeführten kryptographischen Verfahren müssen in vertrauenswürdigen technischen Komponenten implementiert werden, um das geforderte Sicherheitsniveau zu erreichen.
- Weiter sind die Implementierungen der kryptographischen Verfahren und Protokolle selbst in die Sicherheitsanalyse mit einzubeziehen, um z.B. Seitenkanalangriffe zu verhindern.
- Die Sicherheit von technischen Komponenten und Implementierungen muss jeweils dem vorgesehenen Schutzprofil entsprechend durch Common Criteria Zertifikate oder ähnliche Verfahren des BSI, wie zum Beispiel im Zuge einer Zulassung, nachgewiesen werden, wenn eine Übereinstimmung eines Produktes mit den Anforderungen dieser Technischen Richtlinie nachgewiesen werden soll.
- Nach der Entwicklung eines kryptographischen Systems sollte vor dem Produktiveinsatz eine Evaluierung des Systems durch Experten durchgeführt werden, die an der Entwicklung nicht beteiligt waren. Eine Einschätzung der Verfahrenssicherheit durch die Entwickler allein sollte nicht als belastbar betrachtet werden, selbst wenn die Entwickler des Systems über gute kryptographische Kenntnisse verfügen.
- Die Folgen eines Versagens der eingesetzten Sicherheitsmechanismen sollten 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 einen Übergang in einen geeigneten sicheren Zustand gewahrt bleiben.

1.3. Kryptographische Hinweise

Häufig kann ein kryptographisches Verfahren für verschiedene Anwendungen eingesetzt werden, so zum Beispiel Signaturverfahren zur Datenauthentisierung und zur Instanzauthentisierung. In diesem Fall sollten grundsätzlich für die unterschiedlichen Anwendungen jeweils verschiedene Schlüssel eingesetzt werden.

Ein weiteres Beispiel sind symmetrische Schlüssel zur Verschlüsselung und symmetrischen Datenauthentisierung. Hier ist es technisch möglich, einen Schlüssel für beide Verfahren zu benutzen. Allerdings muss bei konkreten Implementierungen dafür gesorgt werden, dass für beide Verfahren jeweils verschiedene Schlüssel eingesetzt werden, die insbesondere nicht auseinander ableitbar sind, siehe auch Abschnitt A.1.

In einigen Fällen beschränkt sich diese Technische Richtlinie auf eine informative Beschreibung der kryptographischen Primitive. Die kryptographische Sicherheit ist aber nur im Rahmen der jeweiligen genauen Spezifikation und des jeweils verwendeten Protokolls bewertbar. Es sind daher die entsprechenden hier angegebenen Standards zu beachten.

Weitere konkrete Hinweise werden, so nötig, in den entsprechenden Abschnitten angegeben.

1.4. Umgang mit Legacy-Algorithmen

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

1. Triple-DES (TDEA) mit Keying Option 1 [74]: Die Hauptpunkte, die gegen eine Verwendung von 3-Key-Triple-DES in neuen Systemen sprechen, sind die geringe Blockbreite von nur 64 Bits, die im Vergleich zum AES verringerte Sicherheit gegenüber generischen Angriffen auf Blockchiffren, sowie verschiedene auch unter diesen Voraussetzungen suboptimale kryptographische Eigenschaften. Zu erwähnen ist zum Beispiel die Existenz von Related-Key-Angriffen gegen Triple-DES mit einer Rechenzeit von $\approx 2^{56}$ Triple-DES-Berechnungen [55]. Auch ohne Berücksichtigung von Related-Key-Angriffen besitzt Triple-DES mit Keying Option 1 kryptographische Eigenschaften, die zwar nach heutigem Kenntnisstand nicht auf praktisch nutzbare Schwächen hinweisen, aber die immerhin negativer sind, als man es für eine ideale Blockchiffre mit 112 Bit effektiver Schlüssellänge erwarten würde [60]. Insgesamt wird hier empfohlen, Triple-DES nicht in neuen Systemen zu verwenden, es sei denn es wäre aus Gründen der Rückwärtskompatibilität zu bestehender Infrastruktur zwingend erforderlich. Auch in diesem Fall sollte eine Migration zu AES in absehbarer Zukunft vorbereitet werden.

2. HMAC-MD5: Die mangelnde Kollisionsresistenz von MD5 ist für MD5 verwendet in der HMAC-Konstruktion noch nicht direkt ein Problem [8], weil 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.

3. HMAC-SHA1: SHA-1 sollte nicht mehr als kollisionsresistente Hashfunktion betrachtet werden. Gegen die Verwendung in Konstruktionen, die keine Kollisionsresistenz benötigen (zum Beispiel als Grundlage für einen HMAC oder als Komponente eines Pseudozufallsgenerators) spricht aber nach gegenwärtigem Kenntnisstand sicherheitstechnisch nichts. Es wird empfohlen, auch in diesen Anwendungen als grundsätzliche Sicherungsmaßnahme eine Hashfunktion der SHA-2-Familie einzusetzen.

Triple-DES mit Keying Option 2 nach [74] zeigt insgesamt deutlich ernsthaftere Schwächen gegenüber chosen-Plaintext und known-Plaintext-Angriffen im Single-Key-Setting als mit Keying Option 1 [64, 65]. Auch wenn keine letztendlich praktischen Angriffe gegen TDEA mit Keying Option 2 bekannt sind, wird hier empfohlen, diese Chiffre nicht nur in neuen Systemen nicht zu verwenden, sondern auch bestehende Kryptoverfahren, die Triple-DES mit zwei Schlüsseln verwenden, so bald wie möglich nach AES (oder wenigstens zu Keying Option 1 nach [74]) zu migrieren.
Soweit Triple-DES noch verwendet wird, sind alle Vorgaben zur Verwendung aus [74] zu beachten.

1.5. Wichtige in dieser Richtlinie nicht behandelte Themen

Ohne Anspruch auf Vollständigkeit zählen wir an dieser Stelle explizit noch einmal einige wichtige Themenbereiche auf, die in der vorliegenden Technischen Richtlinie nicht behandelt werden:

1. Lightweight Cryptography: hierbei treten besonders restriktive Anforderungen an Rechenzeit und Speicherbedarf der eingesetzten kryptographischen Verfahren auf, abhängig von der Anwendung können außerdem auch die Sicherheitsanforderungen andere sein als sonst üblich.

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

3. Festplattenverschlüsselung: Hierbei tritt das Problem auf, dass in den meisten Kontexten eine Verschlüsselung mit Datenexpansion ebenso wie eine deutliche Expansion der Menge an Daten, die vom Speichermedium gelesen beziehungsweise auf das Speichermedium geschrieben werden müssen, nicht akzeptabel ist. Keiner der empfohlenen Verschlüsselungsmodi ist ohne weiteres geeignet als Grundlage einer Lösung zur Festplattenverschlüsselung. Unter der Voraussetzung, dass ein Angreifer nicht Abbilder des Plattenzustandes zu mehreren verschiedenen Zeitpunkten miteinander kombinieren kann, bietet XTS-AES relativ gute Sicherheitseigenschaften und gute Effizienz [68]. Wenn der Angreifer zu einer größeren Anzahl verschiedener Zeitpunkte Kopien des verschlüsselten Speichermediums erstellen kann, dann ist von einem wesentlichen 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.

4. Bei der Verschlüsselung eines Solid State Drives ist in diesem Zusammenhang darauf zu achten, 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-Kontrollers könnte dies potentiell nutzen, um aufeinanderfolgende Zustände einer logischen Speicheradresse nachzuverfolgen. Ein einzelnes Abbild des verschlüsselten Speichermediums ist bei Verwendung einer SSD damit potentiell wertvoller für einen Angreifer als ein einzelnes Abbild einer magnetischen Festplatte.

5. 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 („Cloud-Speicherung“). Wird dem Anbieter

des entfernten Servers oder dessen Sicherheitsmaßnahmen nicht in hohem Maße vertraut, dann muss allerdings in dieser Situation davon ausgegangen werden, dass ein Angreifer Platten-Abbilder zu beliebigen Zeitpunkten, oft und unbemerkt anfertigen kann. Dateien mit sensiblen Daten sollten daher auf nicht unter der physischen Kontrolle des Nutzers stehenden Speichersystemen nur nach Anwendung einer kryptographisch zuverlässigen Dateiverschlüsselung abgelegt werden, selbst wenn vor Übermittlung der Daten eine Volume-Verschlüsselung stattfindet. 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 (zum Beispiel ist dies die Erfordernis unvorhersagbarer Initialisierungsvektoren).

6. Seitenkanalangriffe, physikalische Sicherheitsfragen werden nur am Rande behandelt. Soweit vorhanden, sind Ausführungen zu Seitenkanalangriffen in der vorliegenden Richtlinie nur als beispielhafte Hinweise auf mögliche Gefährdungen aus dieser Richtung **ohne Anspruch auf Vollständigkeit** zu verstehen! 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 Richtlinie nicht behandelt.

7. 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 der 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, 28, 85]. Traffic Flow Confidentiality ist ein Ziel, das im Regelfall nur mit hohem Aufwand vollständig zu erreichen ist und das deshalb auch in vielen Anwendungen, die sensible Informationen verarbeiten, nicht erreicht werden wird. Es sollte allerdings in jedem Einzelfall durch Experten überprüft werden, *wieviel* 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, Widerstandsfähigkeit gegen Preisgabe sensibler Informationen unter Verkehrsflussanalyse in der Entwicklung neuer kryptographischer Systeme von Anfang an als Ziel zu berücksichtigen.

8. 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 sichergestellt werden muss, um einen Datenabfluss zu vermeiden, zwar verringern, das Grundproblem der Endpunktsicherheit aber nicht lösen.

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

2. Symmetrische Verschlüsselungsverfahren

Symmetrische Verschlüsselungsverfahren dienen der Gewährleistung der Vertraulichkeit von Daten, die zum Beispiel über einen öffentlichen Kanal, wie Telefon oder Internet, ausgetauscht werden. Die Authentizität bzw. Integrität der Daten wird dadurch nicht gewährleistet, für einen Integritätsschutz siehe Kapitel 5 und Abschnitt A.1.

In diesem Zusammenhang sollte betont werden, dass 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, eine Vernachlässigung integritätssichernder Mechanismen leicht zu Schwächen im kryptographischen Gesamtsystem führen kann, die das System auch für Angriffe auf die Vertraulichkeit anfällig machen. Besonders Schwächen manchen Typen aktiver Seitenkanalangriffe gegenüber können auf solche Weise entstehen, siehe etwa [88] für ein Beispiel.

Es werden in diesem Kapitel zunächst symmetrische Verfahren behandelt, d.h. Verfahren, in denen der Verschlüsselungs- und der Entschlüsselungsschlüssel gleich sind (im Gegensatz zu asymmetrischen Verfahren, bei denen aus dem öffentlichen Schlüssel der geheime Schlüssel ohne zusätzliche Informationen praktisch nicht berechnet werden kann). Für asymmetrische Verschlüsselungsverfahren, die in der Praxis lediglich als Schlüsseltransportverfahren eingesetzt werden, siehe Kapitel 3.

2.1. Blockchiffren

Allgemeine Empfehlungen Eine Blockchiffre ist ein Algorithmus, der einen Klartext fester Bitlänge (z. B. 128 Bit) mittels eines Schlüssels zu einem Chiffretext gleicher Bitlänge verschlüsselt. Diese Bitlänge heißt auch *Blockgröße* der Chiffre. Für die Verschlüsselung von Klartexten anderer Länge werden sogenannte Betriebsarten eingesetzt, siehe 2.1.1. Für neue Anwendungen sollten nur noch Blockchiffren eingesetzt werden, deren Blockgröße mindestens 128 Bit beträgt.

Folgende Blockchiffren werden zur Verwendung in neuen kryptographischen Systemen empfohlen:

AES-128, AES-192, AES-256, siehe [39].

Tabelle 2.1.: Empfohlene Blockchiffren

In Version 1.0 dieser Technischen Richtlinie wurden auch die Blockchiffren Serpent und Twofish empfohlen. Es liegen keine negativen Erkenntnisse zu diesen Blockchiffren vor, allerdings wurde die Sicherheit von Serpent und Twofish seit dem Ende des AES-Wettbewerbs deutlich weniger intensiv untersucht als die des AES. 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-Angriffe und AES In 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 [12] stellen die einzigen bislang 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-Angriff auf AES-256 aus [12] mit Rechenzeit- und Datenkomplexität von $2^{99.5}$ ist aufgrund der technischen Voraussetzungen von related-Key Boomerang-Angriffen nicht als Verletzung des in dieser Technischen Richtlinie minimal angestrebten Sicherheitsniveaus von 100 Bit zu betrachten.

Die besten bekannten Angriffe gegen AES, die keine related-Keys benötigen, erzielen nur einen geringen Vorteil gegenüber generischen Angriffen [16].

2.1.1. Betriebsarten

Wie in Abschnitt 2.1 bereits festgestellt wurde, liefert eine Blockchiffre an sich lediglich einen Mechanismus zur Verschlüsselung von Klartexten einer einzigen festen Länge. Um Klartexte anderer Länge zu verschlüsseln, muss aus der Blockchiffre mittels einer geeigneten *Betriebsart* eine Verschlüsselungsverfahren für Klartexte (annähernd) beliebiger Länge konstruiert werden. Als weiterer Effekt einer kryptographisch starken Betriebsart ist zu erwähnen, dass das resultierende Verschlüsselungsverfahren in mancher Hinsicht stärker sein wird als die zugrundeliegende Blockchiffre, zum Beispiel wenn die Betriebsart den Verschlüsselungsvorgang randomisiert und damit das Wiedererkennen mehrfach verschlüsselter 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 eventuell noch zu kurz und muss entsprechend aufgefüllt werden, siehe Abschnitt 2.1.3 für geeignete Verfahren. 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 selben Schlüssel zu verschlüsseln (diese Betriebsart heißt auch Electronic Code Book (ECB)). Dies führt aber dazu, dass gleiche Klartextblöcke zu gleichen Chiffretextblöcken verschlüsselt werden. Der Chiffretext liefert damit zumindest Informationen über die Struktur des Klartextes und bei niedriger Entropie pro Block des Klartextes kann ein Wörterbuchangriff realistisch werden. Um dies zu verhindern, sollte der $n-$te Chiffreblock nicht nur vom $n-$ten Klartextblock und dem eingesetzten Schlüssel abhängen, sondern von einem weiteren Wert, wie zum Beispiel dem $(n-1)-$ten Chiffretextblock oder einem Zähler (auch Counter genannt).

Folgende Betriebsarten sind für die unter 2.1 aufgeführten Blockchiffren geeignet:

1. Galois-Counter-Mode (GCM), siehe [70],
2. Cipher-Block Chaining (CBC), siehe [67], und
3. Counter Mode (CTR), siehe [67].

Tabelle 2.2.: Empfohlene Betriebsarten für Blockchiffren

Bemerkung 1 Der Galois-Counter-Mode (GCM) liefert, eine ausreichende Taglänge vorausgesetzt, zusätzlich eine kryptographisch sichere Datenauthentisierung. Für die beiden anderen

Betriebsmodi wird generell empfohlen, separate Mechanismen zur Datenauthentisierung im Gesamtsystem vorzusehen.

2.1.2. Betriebsbedingungen

Für die unter 2.1.1 aufgeführten Betriebsarten werden Initialisierungsvektoren benötigt, außerdem müssen bestimmte weitere Randbedingungen für einen sicheren Betrieb eingehalten werden. Diese Bedingungen sind folgend zusammengefasst.

1. Für GCM:
   - Die Zählerstände im Zähleranteil des Initialisierungsvektors dürfen sich bei gleichem Schlüssel nicht wiederholen. Nichtbeachtung dieser Bedingung führt zu einem praktisch vollständigen Verlust der Vertraulichkeit!
   - Außerdem müssen im GCM Noncen für den integrierten Authentisierungsmechanismus erzeugt werden. Für diese wird nach [70] eine Bitlänge von 96 Bit empfohlen. Dieser Empfehlung schließt sich die vorliegende Technische Richtlinie an, insbesondere mit Verweis auf die Resultate aus [54][1]. Diese Initialisierungsvektoren dürfen sich ebenfalls innerhalb der Lebensdauer eines Authentisierungsschlüssels nicht wiederholen. In [70] wird gefordert, dass die Wahrscheinlichkeit einer Wiederholung der GHASH-Initialisierungsvektoren unter einem gegebenen Schlüssel $\leq 2^{-32}$ sein soll. Daraus ergibt sich ein Schlüsselwechselintervall von $\approx 2^{32}$ Aufrufen von GHASH *unabhängig von der GMAC-Taglänge* [70]. Bei einer Wiederholung von GHASH-Initialisierungsvektoren droht ein vollständiges Versagen des Authentisierungsmechanismus!
   - 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 [70] strikt eingehalten werden.
2. Für CTR: Die Zählerstände dürfen sich bei gleichem Schlüssel nicht wiederholen. Nichtbeachtung dieser Bedingung führt zu einem praktisch vollständigen Verlust der Vertraulichkeit!
3. Für CBC: Es sind nur unvorhersagbare Initialisierungsvektoren zu verwenden, siehe auch Abschnitt B.2.

Für empfohlene Verfahren zur Erzeugung unvorhersagbarer Initialisierungsvektoren siehe Abschnitt B.2.

Bei Anwendungen, für die die hier angegebenen Anforderungen an die Eigenschaften der Initialisierungsvektoren nicht erfüllt werden können, wird dringend die Einbeziehung eines Experten empfohlen.

2.1.3. Paddingverfahren

Wie bereits in Abschnitt 2.1.1 erläutert, benötigt der CBC-Modus einen zusätzlichen Padding-Schritt: Es kann bei der Partitionierung eines zu verschlüsselnden Klartextes geschehen, dass der

[1] In [54] 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 Nonce-Länge von exakt 96 Bit als vorteilhaft.

letzte Klartextblock kleiner als die Blockgröße der eingesetzten Chiffre ist. Eine Formatierung durch das Auffüllen dieses letzten Blocks zu der geforderten Größe heißt auch *Padding*.

Folgende Paddingverfahren werden empfohlen:

1. ISO-Padding, siehe [52], padding method 2 und [67], Appendix A.
2. Padding gemäß [80], Abschnitt 6.3.
3. ESP-Padding, siehe [79] Abschnitt 2.4.

Tabelle 2.3.: Empfohlene Paddingverfahren für Blockchiffren

Bemerkung 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 [88]. Allgemeiner gilt, dass bei Verschlüsselungsverfahren, in denen ein Angreifer Änderungen am Chiffrat so durchführen kann, dass kontrollierte Änderungen am Klartext resultieren, einem Angreifer keine Seitenkanalinformation zur Verfügung stehen darf, die Aufschluss darüber liefert, ob ein gegebenes Chiffrat zu einem gültigen Klartext korrespondiert oder ob es von ungültigem Format ist.

2.2. Stromchiffren

Bei Stromchiffren wird aus einem Schlüssel und einem Initialisierungsvektor zunächst ein sogenannter Schlüsselstrom generiert, das heißt eine pseudozufällige Folge von Bits, die dann auf die zu verschlüsselnde Nachricht bitweise XOR-addiert wird.

Zur Zeit werden keine dedizierten Stromchiffren empfohlen. AES im Counter-Modus kann allerdings natürlich als Stromchiffre aufgefasst werden.

Wird eine Stromchiffre eingesetzt, dann 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.

2.3. Seitenkanalangriffe auf symmetrische Verfahren

Neben der Sicherheit der verwendeten Algorithmen gegen Kryptoanalyse ist die Sicherheit der Implementierung gegen Seitenkanal- und Fault-Attacken für die Sicherheit eines Kryptosystems von entscheidender Bedeutung. Dies gilt auch für symmetrische Verschlüsselungsverfahren. Eine detaillierte Behandlung dieses Themas liegt außerhalb des Rahmens der vorliegenden Technischen Richtlinie, die zu treffenden Gegenmaßnahmen sind auch in hohem Maße vom Einzelfall abhängig. Folgende Maßnahmen sollen hier dennoch empfohlen werden:

- Wo 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 wesentlichem unbemerktem 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.

- Auf Protokollebene sollte der Entstehung von Fehlerorakeln vorgebeugt werden. Am wirkungsvollsten kann das 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.

Generell trifft im Übrigen auch hier die generische Empfehlung zu, wo immer möglich Komponenten zu verwenden, die bereits einer intensiven Analyse durch eine breite Öffentlichkeit unterzogen wurden und frühzeitig entsprechende Experten in die Entwicklung neuer kryptographischer Infrastrukturen einzubinden.

3. Asymmetrische Verschlüsselungsverfahren

Asymmetrische Verschlüsselungsverfahren werden aufgrund ihrer verglichen mit symmetrischen Standardverfahren geringen Effizienz in der Praxis meist zur Übertragung symmetrischer Schlüssel eingesetzt, siehe auch Kapitel 7. Die zu verschlüsselnde Nachricht (d. h. der symmetrische Schlüssel) wird mit dem öffentlichen Schlüssel des Empfängers verschlüsselt. Der Empfänger kann dann die Verschlüsselung mit dem zum öffentlichen Schlüssel assoziierten geheimen Schlüssel wieder rückgängig machen. Dabei darf es praktisch nicht möglich sein, den Klartext ohne Kenntnis des geheimen Schlüssels aus dem Chiffretext zu rekonstruieren. Dies impliziert insbesondere, dass der geheime Schlüssel praktisch nicht aus dem öffentlichen Schlüssel konstruiert werden kann. Um eine Zuordnung des öffentlichen Schlüssels zum Besitzer des zugehörigen geheimen Schlüssels zu garantieren, wird üblicherweise eine Public Key Infrastruktur benötigt.

Für die Spezifizierung von asymmetrischen Verschlüsselungsverfahren sind folgende Algorithmen festzulegen:

1. Ein Algorithmus zur Generierung von Schlüsselpaaren (inklusive Systemparameter).

2. Ein Algorithmus zum Verschlüsseln und ein Algorithmus zum Entschlüsseln der Daten.

Die praktisch relevantesten asymmetrischen Verschlüsselungs- und Signaturverfahren beruhen etwas vereinfacht ausgedrückt entweder auf der Schwierigkeit des Problems der Berechnung diskreter Logarithmen in geeigneten Repräsentationen endlicher zyklischer Gruppen oder auf der Schwierigkeit, große ganze Zahlen in ihre Primfaktoren zu zerlegen. Es taucht gelegentlich die Frage auf, welcher dieser beiden Ansätze als kryptographisch stabiler 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, Verfahren auf Grundlage diskreter Logarithmen gegenüber auf Faktorisierung basierenden Verfahren zu bevorzugen oder umgekehrt. Für besonders hohe Sicherheitsniveaus wird die Verwendung von EC-Verfahren aus Effizienzgründen vorteilhaft, siehe hierzu auch Tabelle 3.2.

Zusätzlich geben wir Empfehlungen für minimale Schlüssellängen an.

Bemerkung 3 Für asymmetrische Verfahren gibt es in der Regel eine Anzahl äquivalenter praktisch relevanter 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 Definition der Schlüssellänge für die empfohlenen asymmetrischen kryptographischen Verfahren wird daher auf das Glossar (Eintrag Schlüssellänge) verwiesen.

Die folgende Tabelle 3.1 gibt einen Überblick über die empfohlenen asymmetrischen Verschlüsselungsverfahren und Schlüssellängen l in Bit.

Tabelle 3.1.: Empfohlene asymmetrische Verschlüsselungsverfahren sowie Schlüssellängen und normative Referenzen.

Verfahren	ECIES	DLIES	RSA
l	224/250[a]	2000 [a b]	2000 [b]
Referenz	[1, 45, 61]	[1, 45]	[82]
Näheres in	Abschnitt 3.3	Abschnitt 3.4	Abschnitt 3.5

[a] ord(g) soll $\geq 2^{224}$ sein und $\geq 2^{250}$ für einen Einsatzzeitraum nach 2015.

[b] Für einen Einsatzzeitraum nach 2015 kann es sinnvoll sein, RSA/DLIES-Schlüssel von 3000 Bits Länge zu verwenden, um ein gleichmäßiges Sicherheitsniveau in allen empfohlenen asymmetrischen Verschlüsselungsverfahren zu erzielen. Die Schlüssellänge von 2000 Bits bleibt bis 2020 zur vorliegenden Richtlinie konform und wird als Mindest-Schlüssellänge für RSA, DLIES und DSA empfohlen. Für Details siehe die Bemerkung 4.

Bemerkung 4 Für Verfahren basierend auf dem Diffie-Hellman-Problem/der Berechnung diskreter Logarithmen in elliptischen Kurven enthalten die vorliegenden empfohlenen Schlüssellängen einen etwas größeren Sicherheitsspielraum verglichen mit den minimalen Sicherheitszielen dieser Technischen Richtlinie, als es bei RSA-Verfahren und Verfahren basierend auf diskreten Logarithmen in endlichen Körpern der Fall ist. Kurz gesagt ist dies auf folgende Gründe zurückzuführen:

1. Die Parametersätze für EC-Verfahren sind standardisiert, ein gegebener Satz von Sicherheitsparametern wird daher für viele unterschiedliche Anwendungen durch eine große Anzahl von Nutzern verwendet werden und stellt somit ein besonders lohnendes Angriffsziel dar.

2. Für *generische* elliptische Kurven führt der effizienteste bekannte Weg zur Lösung von zufälligen Instanzen des Diffie-Hellman-Problems über die Berechnung diskreter Logarithmen durch Varianten des Pollard-Rho-Verfahrens. Häufig werden aber (zum Beispiel aus Effizienzgründen) in EC-Verfahren Kurvenparameter verwendet, die offenkundige nichtgenerische Eigenschaften aufweisen oder deren Erzeugung nur unvollständig dokumentiert wurde. Es ist nicht undenkbar, dass insbesondere für solche Kurven spezielle Eigenschaften gefunden werden, die die Berechnung diskreter Logarithmen dort einfacher machen als im generischen Fall.

Es wird daher 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 sind die Brainpool-Kurven [34].

Bemerkung 5 Die hier empfohlenen asymmetrischen kryptographischen Funktionen benötigen als Bestandteile weitere Unterkomponenten (wie Hashfunktionen, Message Authentication Codes, Zufallszahlenerzeugung, Schlüsselableitungsfunktionen, Blockchiffren), die ihrerseits den Anforderungen der vorliegenden Richtlinie genügen müssen, wenn das angestrebte Sicherheitsniveau erreicht werden soll. In einschlägigen Standards [45, 61] wird dabei teilweise die Verwendung von Verfahren empfohlen, die in der vorliegenden Richtlinie nicht empfohlen werden, und zwar an manchen Stellen auch aus Sicherheitserwägungen heraus (z.B. Two-Key Triple-DES in [45]). Grundsätzlich wird empfohlen, bei der Implementierung eines Standards zwei Grundsätzen zu folgen:

- Für kryptographische Unterkomponenten sollten nur die jeweils in dieser Richtlinie empfohlenen Verfahren verwendet werden.
- Sofern sich dies nicht mit Standardkonformität vereinbaren lässt, ist ein Experte hinzuziehen und die letztlich getroffenen Entscheidungen hinsichtlich der gewählten kryptographischen Unterkomponenten sind gründlich zu dokumentieren und sollten in der Dokumentation unter Sicherheitsgesichtspunkten begründet werden.

Bei der Auswahl der empfohlenen asymmetrischen Verschlüsselungsverfahren wurde darauf geachtet, dass lediglich probabilistische Algorithmen[1] zum Einsatz kommen. Hier 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, wo notwendig, in den Abschnitten zu den entsprechenden Verfahren angegeben.

3.1. Vorbemerkung zu asymmetrischen Schlüssellängen

3.1.1. Allgemeine Vorbemerkungen

Die in dieser Technischen Richtlinie enthaltenen Einschätzungen zur Sicherheit kryptograpischer Verfahren und Schlüssellängen sind, wie bereits in der Einleitung erwähnt wurde, nur bis 2020 gültig. Diese Beschränkung der Aussagekraft dieser Richtlinie ist für asymmetrische Verschlüsselungsverfahren von besonderer Bedeutung. Wir erläutern im Folgenden kurz die Gründe dafür. Daran anschliessend wird kurz auf die Frage eingegangen werden, auf welchem Wege die angegebenen Schlüssellängen hergeleitet werden können.

3.1.1.1. Sicherheit asymmetrischer Verfahren

Die Sicherheit von asymmetrischen kryptographischen Verfahren beruht, soweit es die in dieser Richtlinie behandelten Verfahren betrifft, auf der angenommenen Schwierigkeit von Problemen aus der algorithmischen Zahlentheorie. Im Fall von RSA ist dies das Problem, e-te Wurzeln in $\mathbb{Z}_n$ zu berechnen, wobei n eine hinreichend große Zahl von unbekannter Faktorisierung in zwei Primfaktoren p, q ist und $e > 2^{16}$ teilerfremd zu $\varphi(n) = (p-1)(q-1)$. Die Sicherheit von DLIES und ECIES kann (was die asymmetrische Komponente anbelangt) auf das Diffie-Hellman-Problem in den verwendeten Gruppen zurückgeführt werden. Es existieren damit für alle empfohlenen Verfahren Reduktionen auf natürlich erscheinende Probleme, die allgemein als schwierig eingeschätzt werden.

Im Vergleich zur Situation bei symmetrischen Verschlüsselungsverfahren, die natürlich grundsätzlich durch unvorhergesehene wissenschaftliche Fortschritte auch in ihrer langfristigen Sicherheit bedroht sind, sind aber folgende Punkte hervorzuheben:

- Hinsichtlich des Faktorisierungsproblems für allgemeine zusammengesetzte Zahlen und des Problems der Berechnung diskreter Logarithmen in $\mathbb{F}_p^*$ hat es seit der Einführung asymmetrischer kryptographischer Verfahren mehr praktisch relevante Fortschritte gegeben als bei der Kryptoanalyse der am besten untersuchten Blockchiffren.
- In symmetrischen Chiffren kann die Bedrohung durch aktive Angriffe (vor allem Chosen-Plaintext und Chosen-Ciphertext-Angriffe) zum Teil abgewehrt werden durch ein geeignetes Schlüsselmanagement, insbesondere durch eine sichere Löschung symmetrischer Schlüssel nach Ablauf ihrer vorgesehenen Lebensdauer. Zeigt ein symmetrisches kryptographisches Verfahren erste Schwächen gegen Chosen-Plaintext-Attacken oder

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

Chosen-Ciphertext-Attacken, dann kann zudem eine Migration auf ein anderes Verfahren erfolgen. Bei asymmetrischen Kryptosystemen dagegen verfügt der Angreifer bis in alle Zukunft zumindest noch über die zu den ihn interessierenden Chiffraten gehörenden öffentlichen Schlüssel.

- Überdies würden alle in dieser Richtlinie empfohlenen asymmetrischen Verschlüsselungsverfahren unsicher werden, falls es zu erheblichen Fortschritten bei der Entwicklung von Quantencomputern käme.

Im Vergleich zur Situation bei digitalen Signaturverfahren kommt hinzu, dass ein Angreifer beliebige Chiffrate, zu denen er Zugang hat, für eine Entschlüsselung zu einem beliebigen späteren Zeitpunkt abspeichern kann. Das Ziel der Authentizitätssicherung eines signierten Dokumentes dagegen lässt sich durch rechtzeitige Erzeugung einer neuen Signatur auch noch nachträglich sicherstellen, solange der Beweiswert des alten Signaturverfahrens zum Zeitpunkt der Erstellung der neuen Signatur als gegeben angesehen werden kann. Umgekehrt ist es außerdem auf der rechtlichen Seite möglich, Signaturen mit kryptographisch gebrochenen Verfahren zum Zeitpunkt der Signaturprüfung nicht mehr zu akzeptieren, wenn keine Übersignatur mit einem gültigen Verfahren erfolgte. Im Gegensatz dazu gibt es in der Regel keine nachträglichen Maßnahmen zum Schutz der Vertraulichkeit eines Klartextes zu einem gegebenen Chiffrat.

3.1.1.2. Äquivalente Schlüssellängen für asymmetrische und symmetrische kryptographische Verfahren

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

- Für Verfahren basierend auf elliptische Kurven: Es wird angenommen, dass 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. 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. Es werden für RSA und Verfahren basierend auf diskreten Logarithmen in $\mathbb{F}_p^*$ 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.

- Es wird angenommen, dass es nicht zu einer Anwendung von Angriffen mit Hilfe von Quantencomputern kommt.

Diese Annahmen sind insofern aus Angreifersicht pessimistisch, als sie keinen Spielraum für strukturelle Fortschritte in der Kryptoanalyse asymmetrischer Verfahren enthalten. Fortschritte, die mit den obigen Annahmen inkompatibel sind, können von sehr spezialisierter Natur sein und sich zum Beispiel auf neue Erkenntnisse zu *einer einzigen* elliptischen Kurve beziehen. Obwohl grundsätzlich eine Berechnung mit 2^{100} Elementaroperationen für den für diese Richtlinie relevanten Zeitraum als nicht praktisch durchführbar angesehen wird, liegen daher

$log_2(R)$	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 3.2.: Ungefährer Rechenaufwand R (in Vielfachen des Rechenaufwandes für eine DES-Auswertung) für die Berechnung diskreter Logarithmen in elliptischen Kurven (ECDLP) beziehungsweise Faktorisierung allgemeiner zusammengesetzter Zahlen mit den angegebenen Bitlängen.

alle empfohlenen Schlüssellängen oberhalb des in dieser Richtlinie minimal angestrebten 100-Bit-Sicherheitsniveaus.
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 aus [19, 29].
Zur Abschätzung der Laufzeiten folgen wir [35], Kapitel 6. Insbesondere nehmen wir wie [35] an, 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 fraglichen Gruppen - etwa die in Tabelle 3.2 wiedergegebenen Äquivalenzen (vergleiche [35], Tabelle 7.2). Zu empfohlenen Schlüssellängen siehe Tabelle 3.1.

3.1.2. 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, d.h. deutlich über das Jahr 2020 hinaus. Eine zuverlässige Prognose über die Eignung von kryptographischen Verfahren ist dann über den gesamten Lebenszyklus des Systems hinweg nicht mehr möglich. Es wird für diese Situation empfohlen, unter Hinzuziehung eines Experten über die Minimalforderungen dieser Richtlinie wesentlich hinausgehende Schutzmechanismen vorzusehen. Beispielhaft werden folgend verschiedene Wege zur Risikominimierung erläutert:

- Bei der Neuentwicklung von kryptographischen Systemen mit projektiert 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 Praktikabilität solcher Verfahrenswechsel sollte, wenn eine lange Einsatzdauer vorgesehen ist, schon in der Entwicklung des ursprünglichen Systems wo möglich berücksichtigt werden.

- Bei hohem und langfristigem Schutzbedarf der übermittelten Informationen sollten bereits

bei Einführung des Systems höhere asymmetrische Schlüssellängen als in dieser Richtlinie gefordert eingesetzt werden. Eine naheliegende Möglichkeit besteht darin, für alle Systemkomponenten ein einheitliches Sicherheitsniveau von ≥ 128 Bit anzustreben. Hinweise zu den für verschiedene Sicherheitsniveaus minimal erforderlichen asymmetrischen Schlüssellängen können in diesem Fall aus Tabelle 3.2 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 besonders für Informationen, die mit einem hybriden oder asymmetrischen Kryptoverfahren verschlüsselt übertragen werden.

- Um ein gewisses Maß an Quantencomputer-Sicherheit zu erreichen, können zudem asymmetrische Verfahren durch die Verwendung zusätzlicher symmetrischer Verfahren (unter Verwendung symmetrischer Langzeitschlüssel) verstärkt werden. Folgende Möglichkeiten hierzu bieten sich beispielhaft etwa an:
  - Üblicherweise wird asymmetrische Kryptographie wie bereits gesagt lediglich benötigt, um ein gemeinsames Geheimnis zwischen den Kommunikationspartnern auszutauschen, aus dem dann symmetrische Sitzungsschlüssel abgeleitet werden. Dabei ist es möglich, in die Schlüsselableitungsfunktion ein kryptographisches Langzeitgeheimnis eingehen zu lassen. Ein Angreifer, der das dem asymmetrischen Verfahren zugrundeliegende mathematische Problem effizient lösen kann, scheitert in diesem Fall an der korrekten Ableitung der Sitzungsschlüssel, solange er den durch die Schlüsselableitung genutzten symmetrischen Schlüssel nicht kennt.
  - Ebenso ist es möglich, einen asymmetrischen Schlüsseltausch mit Hilfe eines vorverteilten Geheimnisses symmetrisch zu verschlüsseln.

  In diesem Fall muss jeweils natürlich das Problem der Verteilung der erwähnten Langzeitschlüssel gelöst werden.

- Eine weitere Möglichkeit, künftige Angriffe durch Quantencomputer abzuwehren, besteht natürlich in der Anwendung asymmetrischer kryptographischer Verfahren, für die Resistenz gegen Quantencomputer-Angriffe angenommen wird. Die vorliegende Richtlinie empfiehlt keine Quantencomputer-resistenten Verschlüsselungsverfahren, da die Bedrohung durch Quantencomputer nicht als akut angesehen wird. Umgekehrt ist zudem die Wahl der Sicherheitsparameter für die in dieser Richtlinie empfohlenen Verfahren unter der Annahme rein klassischer Attacken durch die Forschung deutlich besser geklärt als es für Quantencomputer-resistente Verfahren derzeit der Fall ist.

Für eine ausführlichere Diskussion über langfristig sichere Schlüssellängen in asymmetrischen kryptographischen Verfahren verweisen wir auf [35, 59].

3.2. Sonstige Bemerkungen

3.2.1. Seitenkanalangriffe und Fault-Attacken

Für asymmetrische Verschlüsselungsverfahren beziehungsweise asymmetrische digitale Signaturverfahren können verschiedene Seitenkanalattacken relevant sein, deren Anwendbarkeit auf die jeweils gegebene Situation überprüft werden muss. Dieses Thema kann in der vorliegenden Richtlinie nicht umfassend behandelt werden. Die Sicherheit der Implementierung gegen Seitenkanalangriffe sollte bei Bestehen relevanter Bedrohungen überprüft werden. Gleiches gilt für Fault-Attacken.
Detaillierte Empfehlungen zu diesem Thema finden sich in [4] für kryptographische Verfahren

auf Basis elliptischer Kurven. Ein entsprechendes Dokument für RSA, $\mathbb{F}_p$-DH und entsprechende Signaturverfahren befindet sich in Vorbereitung.

Seitenkanalangriffe betreffen natürlich auch symmetrische Primitive, siehe Abschnitt 2.3.

3.2.2. 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 also nur gültig, wenn Man-in-the-Middle-Angriffe durch ein weiteres Verfahren zuverlässig verhindert werden.

Solche Angriffe können nur dann zuverlässig abgewehrt werden, wenn eine authentische Verteilung der öffentlichen Schlüssel aller Teilnehmer sichergestellt ist. Hierzu gibt es verschiedene Möglichkeiten, in der Regel wird aber eine Public-Key-Infrastruktur (PKI) herangezogen. 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 Signatursystem wird empfohlen, die folgend aufgelisteten Punkte zu berücksichtigen. Es handelt sich hierbei nicht um eine erschöpfende Liste von Entwicklungsanforderungen an Public-Key-Infrastrukturen, sondern lediglich um eine Liste vergleichsweise generischer Punkte, die zu beachten bei der Entwicklung einer PKI sinnvoll erscheint. Es werden sich in der Regel bei der Entwicklung und Evaluierung eines Systems weitere Anforderungen ergeben, die hier nicht aufgelistet sind. Die Entwicklung einer geeigneten PKI für eine neue kryptographische Anwendung ist keine triviale Aufgabe und sollte in enger Abstimmung mit entsprechenden Experten angegangen werden.

1. 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 möglich verbunden mit einem sicheren Transport zum Endbenutzer.

2. Es sollte Möglichkeiten zur zeitnahen Deaktivierung von Zertifikaten geben und es sollte einem Angreifer nicht möglich sein, unbemerkt zu verhindern, dass die Information über den aktuellen Status eines Zertifikats zum Zeitpunkt der Prüfung dem prüfenden Teilnehmer zur Verfügung steht.

3. Zertifikate sollten nur zeitlich befristet ausgestellt werden.

4. Alle Zertifikatsaussteller müssen vertrauenswürdig sein.

5. 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.

6. Die Länge von Zertifikatsketten sollte (durch einen möglichst niedrigen Wert) nach oben beschränkt werden.

3.3. ECIES-Verschlüsselungsverfahren

Allgemeine Beschreibung ECIES steht für *Elliptic Curve Integrated Encryption Scheme*. Es handelt sich hierbei um ein hybrides Verschlüsselungsverfahren. Die Sicherheit der asymmetrischen Komponente basiert auf dem Diffie-Hellman-Problem in der jeweils verwendeten elliptischen Kurve. Wir beschreiben im Folgenden eine Version von ECIES, die mit den übrigen

Empfehlungen der vorliegenden Technischen Richtlinie vereinbar ist. Dabei lehnen wir uns in der Verfahrensbeschreibung eng an [1] an.
Die hier wiedergegebene Beschreibung von ECIES ist fast durchgehend identisch zur Beschreibung des eng verwandten Verfahrens DLIES in Abschnitt 3.4. Der Hauptgrund für eine separate Behandlung beider Verfahren sind Schwierigkeiten, die sich aus Differenzen in den Notationen ergeben könnten sowie die für beide Verfahren unterschiedlichen Empfehlungen hinsichtlich sicherer Schlüssellängen. Als normative Referenz wird ECIES-HC in [61] empfohlen.
Für einen Überblick zur Standardisierung von ECIES und DLIES empfehlen wir [62].

Komponenten ECIES benötigt folgende Komponenten:

- Ein symmetrisches Verschlüsselungsverfahren E_K. Alle in der vorliegenden Richtlinie empfohlenen Kombinationen aus Blockchiffre und Betriebsmodus sind hier geeignet.
- Einen Message Authentication Code MAC_{KM}. Es können die in Abschnitt 5.1 empfohlenen Verfahren verwendet werden.
- Eine Schlüsselableitungsfunktion H. H kann einfach eine Hashfunktion sein, falls deren Ausgabe mindestens die Länge des gesamten abzuleitenden symmetrischen Schlüsselmaterials besitzt. Alternativ kann auch die im Abschnitt B.1 empfohlene Schlüsselableitungsfunktion oder eine der in [61] vorgeschlagenen Schlüsselableitungsfunktionen verwendet werden, um aus den gegebenen Daten abgeleitetes Schlüsselmaterial der gewünschten Länge zu erzeugen.

Benötigt wird außerdem 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 $G := d \cdot P$.

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

Verschlüsselung Sei gegeben 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 dann eine zufällige Zahl $k \in \{1, \ldots, q-1\}$ und berechnet $B := k \cdot P$. Er berechnet weiter $X := k \cdot G$ 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 werden nun ein Chiffrat $C := E_K(M)$ sowie ein MAC $T := \mathrm{MAC}_{KM}(C)$ berechnet. Schliesslich sendet S das Tupel (B, C, T) an E.

Entschlüsselung E empfängt (B, C, T) und berechnet $X := d \cdot B$ 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 Für die Ordnung q des Basispunktes P muss $q \geq 2^{224}$ gelten. Für einen Einsatzzeitraum nach Ende 2015 wird $q \geq 2^{250}$ empfohlen.

Eine notwendige Voraussetzung für die Sicherheit des ECIES-Verfahrens ist die praktische Unmöglichkeit, das Diffie-Hellman-Problem in der von P erzeugten Untergruppe zu lösen. Bei den empfohlenen Kurvenparametern ist das nach heutigem Kenntnisstand der Fall.

Bemerkung 6 Wie DLIES ist das hier vorgestellte Verfahren ein probabilistischer Algorithmus. Hier muss ebenfalls ein Zufallswert $k \in \{1, \ldots, q-1\}$ annähernd ideal zufällig gewählt werden. Siehe Abschnitt B.4 für einen empfohlenen Algorithmus zur Berechnung des Zufallswertes k.

3.4. DLIES-Verschlüsselungsverfahren

Allgemeine Beschreibung DLIES steht für *Discrete Logarithm Integrated Encryption Scheme*. Es handelt sich um ein hybrides Verschlüsselungsverfahren, das in der asymmetrischen Komponente auf der Schwierigkeit einer Lösung von Instanzen des Diffie-Hellman-Problems in einer geeigneten Untergruppe von $\mathbb{F}_p^*$ beruht. Wir beschreiben im Folgenden eine Version von DLIES, die mit den übrigen Empfehlungen der vorliegenden Technischen Richtlinie vereinbar ist. Dabei lehnen wir uns in der Verfahrensbeschreibung eng an [1] an.

Eine normative Beschreibung findet sich in [45].

Komponenten DLIES benötigt folgende Komponenten:

- Ein symmetrisches Verschlüsselungsverfahren E_K. Alle in der vorliegenden Richtlinie empfohlenen Kombinationen aus Blockchiffre und Betriebsmodus sind hier geeignet.
- Einen Message Authentication Code MAC_{KM}.
- Eine Schlüsselableitungsfunktion H. H kann einfach eine Hashfunktion sein, falls deren Ausgabe mindestens die Länge des gesamten abzuleitenden symmetrischen Schlüsselmaterials besitzt.

Hinsichtlich der empfohlenen Realisierung dieser Komponenten gelten die diesbezüglichen Empfehlungen aus Abschnitt 3.3 entsprechend. Benötigt wird außerdem Schlüsselmaterial wie im folgenden Abschnitt zur Schlüsselgenerierung beschrieben.

Schlüsselgenerierung

1. Wähle eine Primzahl $p \in \mathbb{N}$.
2. Wähle ein Element g der multiplikativen Gruppe $\mathbb{F}_p^*$, dessen Ordnung r prim ist.
3. Wähle eine zufällige Zahl $a \in \{1, \ldots, r-1\}$ und setze $A := g^a$.

Dann ist (p, g, A, r) der öffentliche Schlüssel und a der geheime Schlüssel.

Verschlüsselung Sei gegeben eine Nachricht $M \in \{0,1\}^*$ und ein öffentlicher Schlüssel (p, g, A, r), der auf zuverlässige Weise dem berechtigten Empfänger E der Nachricht zugeordnet werden kann. Zur Verschlüsselung wählt der Sender S dann eine zufällige Zahl $b \in \{1, \ldots, r-1\}$ und berechnet $B := g^b$. Er berechnet weiter $X := A^b$ 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 werden nun ein Chiffrat $C := E_K(M)$ sowie ein MAC $T := \mathrm{MAC}_{KM}(C)$ berechnet. Schliesslich sendet S das Tupel (B, C, T) an E.

Entschlüsselung E empfängt (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 2000 Bit betragen, die Länge der Primzahl r sollte mindestens 250 Bit betragen. Fußnote b) zu Tabelle 3.1 und Bemerkung 4 aus Kapitel 3 gelten entsprechend.

Eine notwendige Voraussetzung für die Sicherheit des DLIES-Verfahrens ist die praktische Unmöglichkeit, den diskreten Logarithmus in der von g erzeugten Untergruppe zu bestimmen. Bei der empfohlenen Mindestgröße der erzeugten Untergruppe von 2^{250} und der empfohlenen Länge von p ist das nach gegenwärtigem Kenntnisstand der Fall.

Bemerkung 7 Das DLIES-Verfahren ist ein sogenannter probabilistischer Algorithmus, d.h. für die Berechnung des Chiffretextes wird eine Zufallszahl k benötigt. Hier ist $k \in \{1, \ldots, r-1\}$ und sollte bezüglich der Gleichverteilung auf $\{1, \ldots, r-1\}$ gewählt werden. In Abschnitt B.4 werden drei Algorithmen zur Berechnung von k besprochen.

3.5. RSA

Schlüsselgenerierung

1. Wähle zwei Primzahlen p und q zufällig und unabhängig voneinander unter der Nebenbedingung
$$0.1 < |\log_2 p - \log_2 q| < 30.$$

2. Bei der empfohlenen Schlüssellänge von 2000 Bit (s.u.), wähle den öffentlichen Exponenten $e \in \mathbb{N}$ unter den Nebenbedingungen
$$\mathrm{ggT}(e, (p-1)\cdot(q-1)) = 1 \text{ und } 2^{16} + 1 \leq e \leq 2^{1824} - 1.$$

3. Berechne den geheimen Exponenten $d \in \mathbb{N}$ in Abhängigkeit von e unter der Nebenbedingung
$$e \cdot d = 1 \bmod \mathrm{kgV}(p-1, q-1).$$

Mit $n = p \cdot q$ (dem sogenannten Modulus) ist dann (n, e) der öffentliche Schlüssel und d der geheime Schlüssel. Weiter müssen natürlich auch die beiden Primzahlen p und q geheim gehalten werden, da sonst jeder aus dem öffentlichen Schlüssel (n, e) wie unter Punkt 3. den geheimen Exponenten berechnen kann. Es wird empfohlen, aus der Schlüsselgenerierung keine Daten abgesehen von den erzeugten Schlüsseln 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 8 (*i*) Die Reihenfolge der Wahl der Exponenten, d.h. erst die Wahl von e und dann von d soll die zufällige Wahl kleiner geheimer Exponenten verhindern, siehe [17].

(*ii*) Bei der Verwendung probabilistischer Primzahltests zur Erzeugung der beiden Primzahlen p und q sollte die Wahrscheinlichkeit dafür, dass eine der Zahlen doch zusammengesetzt ist, höchstens 2^{-100} betragen, siehe Abschnitt B.5 für geeignete Verfahren.

Verschlüsselung und Entschlüsselung Für die Ver- und Entschlüsselung siehe [82]. Allerdings muss zusätzlich die Nachricht vor Anwendung des geheimen Schlüssels d auf die Bitlänge des Modulus n formatiert werden. Das Formatierungsverfahren ist dabei sorgfältig zu wählen. Das folgende Verfahren wird empfohlen:

EME-OAEP, siehe [82].

Tabelle 3.3.: Empfohlenes Formatierungsverfahren für den RSA-Verschlüsselungsalgorithmus

Schlüssellänge Die Länge des Modulus n sollte mindestens 2000 Bit betragen. Fußnote b) zu Tabelle 3.1 und Bemerkung 4 aus Kapitel 3 gelten entsprechend.

Eine notwendige Voraussetzung für die Sicherheit des RSA-Verfahrens ist die praktische Unmöglichkeit, den Modul n ohne Kenntnis von p und q in seine Primfaktoren zu zerlegen. Bei der empfohlenen Mindestbitlänge von 2000 Bits ist das nach gegenwärtigem Kenntnisstand der Fall.

4. Hashfunktionen

Hashfunktionen 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. Diese Funktionen spielen in vielen kryptographischen Verfahren eine große Rolle, so zum Beispiel bei der Ableitung kryptographischer Schlüssel oder bei der Datenauthentisierung.

Hashfunktionen $H : \{0,1\}^* \longrightarrow \{0,1\}^n$, die in kryptographischen Verfahren eingesetzt werden, müssen je nach Anwendung die folgenden drei Bedingungen erfüllen:

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

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

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

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

Mathematisch präziser lassen sich diese drei Begriffe fassen jeweils durch einen Vergleich der besten bekannten Angriffe auf diese Eigenschaften mit optimalen generischen Angriffen.

Die Länge des Hash-Outputs ist dabei ein Sicherheitsparameter von zentraler Bedeutung, weil er den Aufwand generischer Angriffe bestimmt. Für das in dieser Technischen Richtlinie minimal geforderte Sicherheitsniveau von 100 Bit muss wegen des Geburtstagsparadoxons für eine Hashfunktion $H : \{0,1\}^* \longrightarrow \{0,1\}^n$ mindestens die Bedingung $n \geq 200$ gelten.

Bemerkung 9 Es gibt kryptographische Anwendungen von Hashfunktionen, in denen nicht alle drei angegebenen Eigenschaften einer starken Hashfunktion benötigt werden. Umgekehrt gibt es sinnvolle weitere 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, [18]), die im Zusammenhang mit ECDSA-Signaturen von Bedeutung ist. Die 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.

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

- SHA-224, SHA-256, SHA-512/256, SHA-384, SHA-512, SHA-512/224, siehe [37].
- Für einen Einsatzzeitraum nach Ende 2015 wird empfohlen, SHA-256, SHA-384, SHA-512 oder SHA-512/256 zu verwenden.

Tabelle 4.1.: Empfohlene Hashfunktionen

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

Bemerkung 10 (*i*) Die Kollisionsangriffe der Arbeitsgruppe um die chinesische Kryptologin X. Wang (siehe [89]) auf die Hashfunktion SHA-1 (spezifiziert in [37]) und darauf aufbauende weitere Arbeiten lassen es als sehr wahrscheinlich erscheinen, dass die Erzeugung von Kollisionen gegen SHA-1 heute prinzipiell praktisch möglich wäre. Der derzeit beste bekannte Angriff auf die Kollisionsresistenz von SHA-1 findet sich in [87], wo der Aufwand zur Erzeugung einer SHA-1-Kollision unter Verwendung der dort beschriebenen Techniken mit zwischen $2^{60.3}$ und $2^{65.3}$ SHA-1-Operationen abgeschätzt wird. In Anwendungen, die eine kollisionsresistente Hashfunktion benötigen, sollte daher SHA-1 definitiv nicht mehr eingesetzt werden.
(*ii*) Man beachte, dass schon eine einzige Kollision einer Hashfunktion zu einer Unsicherheit bei Signaturverfahren führen kann, vergleiche z.B. [32] und [42].

5. Datenauthentisierung

Unter Datenauthentisierung verstehen wir in dieser Technischen Richtlinie kryptographische Verfahren, die garantieren, dass übersandte oder gespeicherte Daten nicht durch Unbefugte verändert wurden. Genauer 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) prüft 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.

Man unterscheidet symmetrische und asymmetrische Verfahren. Bei symmetrischen Verfahren benutzen Beweisender und Prüfer den selben kryptographischen Schlüssel, ein Dritter kann also in diesem Fall nicht überprüfen, wer die Prüfsumme berechnet hat oder ob sie überhaupt richtig berechnet wurde. Bei asymmetrischen Verfahren wird der private Schlüssel für die Berechnung der Prüfsumme benutzt und mit dem assoziierten öffentlichen Schlüssel überprüft.

5.1. Message Authentication Code (MAC)

Message Authentication Codes sind symmetrische Verfahren zur Datenauthentisierung, die sich üblicherweise auf Blockchiffren oder Hashfunktionen stützen. Beweisender und Prüfer müssen also vorab einen gemeinsamen symmetrischen Schlüssel vereinbart haben. Diese Verfahren werden üblicherweise dann eingesetzt, wenn große Datenmengen authentisiert werden müssen oder wenn Prüfung oder Erstellung von Prüfsummen aus anderen Gründen besonders effizient sein müssen. Häufig muss sowohl die Vertraulichkeit als auch die Authentizität der Daten gewährleistet werden, siehe Abschnitt A.1 für solche Verfahren. Siehe weiter Kapitel 7 für Verfahren, mit denen Schlüssel über unsichere Kanäle ausgetauscht werden können.

Grundsätzlich gelten die folgenden Verfahren als sicher, wenn im CMAC-Verfahren und im GMAC-Verfahren eine aus Tabelle 2.1 aufgeführte Blockchiffre einsetzt wird, bzw. im HMAC-Verfahren eine aus Tabelle 4.1 aufgeführte Hashfunktion eingesetzt wird und die Länge des Schlüssels für beide Verfahren mindestens 16 Byte beträgt:

- CMAC, siehe [69],
- HMAC, siehe [8],
- GMAC, siehe [70].

Tabelle 5.1.: Empfohlene MAC-Verfahren

Zur Verwendung dieser Verfahren sind folgende Empfehlungen zu beachten:

1. Als Taglänge werden für allgemeine kryptographische Anwendungen in allen drei Verfahren $\geq$ 96 Bits empfohlen, als absolutes Minimum für allgemeine Anwendungen gibt die vorliegende Richtlinie 64 Bit an. Kürzere Taglängen sollten nur verwendet werden nach Abwägung aller die jeweilige Anwendung betreffenden Umstände durch Experten. Für GMAC-Tags gilt, dass Angriffe existieren, in denen Fälschungen von Tags der Länge t für

Nachrichten von n Blocks Länge mit einer Wahrscheinlichkeit von $2^{-t+\log_2(n)}$ pro Versuch möglich sind und sich diese Wahrscheinlichkeit bei Detektion erfolgreicher Fälschungen weiter steigert [40]. Dies bedeutet, dass bei gleicher Taglänge GMAC (und damit auch der authentisierte Verschlüsselungsmodus GCM) einen schwächeren Integritätsschutz liefert als es für CMAC oder HMAC jeweils mit den in dieser Technischen Richtlinie empfohlenen Blockchiffren beziehungsweise Hashfunktionen erwartet wird. Die praktische Relevanz dieser Angriffe wächst erheblich, wenn kurze Authentisierungs-Tags (< 64 Bit) eingesetzt werden. Von einer Verwendung kurzer Tags mit GMAC/GCM wird daher dringend abgeraten.

2. Die verwendeten Authentisierungsschlüssel sind ebenso gut zu schützen wie sonstige kryptographische Geheimnisse im gleichen Kontext.

3. Allgemein müssen alle Auflagen aus [8, 69, 70] bei dem jeweils verwendeten Verfahren eingehalten und ihre Einhaltung dokumentiert werden.

Hinsichtlich des GMAC-Verfahrens gelten die sonstigen Bemerkungen zu den Betriebsbedingungen für GCM aus Abschnitt 2.1.2 entsprechend, soweit sie die Authentisierungsfunktion betreffen. Die folgende Tabelle fasst die Empfehlungen zu Schlüssel- und Prüfsummenlänge bei Verwendung von MAC-Verfahren zusammen:

Verfahren	CMAC	HMAC	GMAC
Schlüssellänge	≥ 128	≥ 128	≥ 128
Taglänge empfohlen	≥ 96	≥ 96	≥ 96

Tabelle 5.2.: Parameter für empfohlene MAC-Verfahren

5.2. Signaturverfahren

In Signaturalgorithmen werden die zu signierenden Daten zunächst gehasht und dann aus diesem Hashwert die Prüfsumme bzw. die Signatur mit dem geheimen Schlüssel des Beweisenden berechnet. Der Prüfer verifiziert dann die Signatur anhand der Daten mit dem zu dem geheimen Schlüssel assoziierten öffentlichen Schlüssel. Wie schon bei asymmetrischen Verschlüsselungsverfahren darf es dabei praktisch nicht möglich sein, die Signatur ohne Kenntnis des geheimen Schlüssels zu berechnen. Dies impliziert insbesondere, dass der geheime Schlüssel praktisch nicht aus dem öffentlichen Schlüssel konstruiert werden kann.

Zur Verteilung der öffentlichen Schlüssel an die Verifizierer wird üblicherweise eine Public Key Infrastruktur genutzt. Ein zuverlässiger (vor Manipulationen sicherer) Weg zur Verteilung der öffentlichen Schlüssel ist in jedem Fall wie bei allen Public-Key-Verfahren unerlässlich.

Für die Spezifizierung von Signaturverfahren sind also folgende Algorithmen festzulegen:

1. Ein Algorithmus zur Generierung von Schlüsselpaaren.

2. Eine Hashfunktion, die die zu signierenden Daten auf einen Datenblock fester Bitlänge abbildet.

3. Ein Algorithmus zum Signieren der gehashten Daten und ein Algorithmus zum Verifizieren der Signatur.

Zusätzlich geben wir Empfehlungen für minimale Schlüssellängen an.

Für die Berechnung des Hashwertes sind grundsätzlich alle der in Tabelle 4.1 aufgelisteten Hashfunktionen geeignet. Wir müssen also in den folgenden vier Unterabschnitten jeweils nur

noch die unter Punkt 1. und 3. aufgeführten Algorithmen und Schlüssellängen angeben. Im übrigen können alle empfohlenen Verfahren sowohl zur Signierung von Daten, als auch zum Ausstellen von Zertifikaten genutzt werden.

Tabelle 5.3 gibt einen Überblick über die im Folgenden empfohlenen Signaturverfahren.

1. RSA, siehe [48],
2. DSA, siehe [49] und [38],
3. DSA-Varianten auf elliptischen Kurven:
   a) ECDSA, siehe [23],
   b) ECKDSA, ECGDSA, siehe [23, 49], und
4. Merkle-Signaturen, siehe [27][a]

[a] Merkle-Signaturen unterscheiden sich in wesentlichen Aspekten von den anderen an dieser Stelle empfohlenen Signaturverfahren. Für eine genauere Beschreibung der wichtigsten Punkte wird auf Abschnitt 5.2.4 verwiesen.

Tabelle 5.3.: Empfohlene Signaturverfahren

Hierbei ist zu bemerken, dass nicht alle diese Verfahren zur Erstellung *qualifizierter* elektronischer Signaturen zugelassen sind. Welche Verfahren zur Erstellung qualifizierter elektronischer Signaturen geeignet sind, geht hervor aus dem Algorithmenkatalog zum deutschen Signaturgesetz, siehe [15].

Bemerkung 11 Mit Ausnahme des DS 3 (vergl. Tabelle 5.4) sind die empfohlenen asymmetrischen Signaturverfahren probabilistische Algorithmen[1]. Hier wird also bei jeder Berechnung einer Signatur ein **neuer** Zufallswert benötigt. Anforderungen an diese Zufallswerte werden in den entsprechenden Abschnitten angegeben.

5.2.1. RSA

Die Sicherheit dieses Verfahrens beruht auf der vermuteten Schwierigkeit der Berechnung e-ter Wurzeln in $\mathbb{Z}/(n)$, wenn 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 exakt wie beim RSA-Verschlüsselungsverfahren, zu Details siehe Abschnitt 3.5. Der Signaturprüfschlüssel ist von der Form (n, e) (n zusammengesetzt, $n \geq 2^{2000}$, $2^{16} < e < 2^{1824}$) und der Signaturschlüssel ist $d := e^{-1} (\mathrm{mod}\ \varphi(n))$.

Signaturerzeugung und Signaturverifikation Für die Signaturerzeugung bzw. -verifikation siehe [48]. Allerdings muss zusätzlich der Hashwert der Nachricht vor Anwendung des geheimen Schlüssels d auf die Bitlänge des Moduls n formatiert werden. Das Formatierungsverfahren ist dabei sorgfältig zu wählen, siehe zum Beispiel [30]. Die folgenden Verfahren werden empfohlen:

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

1. EMSA-PSS, siehe [82].
2. Digital Signature Scheme (DS) 2 und 3, siehe [51].

Tabelle 5.4.: Empfohlene Formatierungsverfahren für den RSA-Signaturalgorithmus

Schlüssellänge Die Länge des Modulus n sollte mindestens 2000 Bit betragen. Fußnote b) zu Tabelle 3.1 und Bemerkung 4 aus Kapitel 3 gelten entsprechend.

5.2.2. Digital Signature Algorithm (DSA)

Die Sicherheit dieses Verfahrens beruht auf der vermuteten Schwierigkeit des Diskreten Logarithmenproblems in $\mathbb{F}_p^*$.

Schlüsselgenerierung

1. Wähle zwei Primzahlen p und q so, dass

   $$q \text{ teilt } p-1$$

   gilt.

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

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

Signaturerzeugung und Signaturverifikation Für die Signaturerzeugung bzw. -verifikation siehe [49] und [38].
Signaturerzeugung und Signaturverifikation benötigen eine kryptographische Hashfunktion. Dabei sollte eine der in der vorliegenden Richtlinie empfohlenen Hashfunktionen verwendet werden. Die Länge der Hashwerte sollte der Bitlänge von q entsprechen. Sollte keine der in Tabelle 4.1 empfohlenen Hashfunktionen eine geeignete Hashlänge aufweisen, dann sollten die q führenden Bits der Hash-Ausgabe verwendet werden. Ist die Länge L_H des Hashwertes *geringer* als die Bitlänge von q, dann ergibt sich ein Signaturverfahren mit einem Sicherheitsniveau von (höchstens) $L_H/2$ Bit.

Schlüssellänge Die Länge der Primzahl p sollte mindestens 2000 Bit (Fußnote b) zu Tabelle 3.1 und Bemerkung 4 aus Kapitel 3 gelten entsprechend) und die Länge der Primzahl q mindestens 224 Bit betragen. Für einen Einsatzzeitraum nach 2015 sollte q mindestens eine Bitlänge von 250 Bit besitzen.

Bemerkung 12 Das DSA-Verfahren ist ein so genannter probabilistischer Algorithmus, da zur Berechnung der Signatur eine Zufallszahl k benötigt wird. Hier ist $k \in \{1, \ldots, q-1\}$, und diese Zufallszahl sollte bezüglich der Gleichverteilung auf $\{1, \ldots, q-1\}$ gewählt werden. Andernfalls existieren Angriffe, vergleiche [76]. In Abschnitt B.4 werden zwei Algorithmen zur Berechnung von k besprochen.

5.2.3. DSA-Varianten basierend auf elliptischen Kurven

Die Sicherheit dieser Verfahren beruht auf der vermuteten Schwierigkeit des Diskreten Logarithmenproblems 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 geheimen Schlüssel.

Signaturerzeugung und Signaturverifikation Folgende Algorithmen sind grundsätzlich geeignet:

1. ECDSA, siehe [23].
2. ECKDSA, ECGDSA, siehe [23, 49].

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 5.2.2 gelten entsprechend.

Schlüssellänge Alle in Tabelle 5.5 aufgeführten Signaturverfahren garantieren ein Sicherheitsniveau von 100 Bit, wenn für die Ordnung q des Basispunktes P gilt $q \geq 2^{200}$ und wenn angenommen wird, dass die Berechnung diskreter Logarithmen auf den verwendeten Kurven nicht effizienter möglich ist als durch generische Verfahren. Empfohlen wird, $q \geq 2^{224}$ zu wählen. Für einen Einsatzzeitraum nach 2015 wird $q \geq 2^{250}$ empfohlen bei einer Kurvengröße von $\geq 2^{256}$.

Bemerkung 13 Wie das DSA-Verfahren sind alle in diesem Abschnitt empfohlenen Signaturverfahren probabilistische Algorithmen. Hier muss ebenfalls ein Zufallswert $k \in \{1, \ldots, q-1\}$ gemäß der Gleichverteilung gewählt werden, da andernfalls Angriffe existieren, vergleiche [76]. werden. In Abschnitt B.4 werden zwei Verfahren zur Berechnung von k vorgestellt.

5.2.4. Merklesignaturen

Im Gegensatz zu den bisher beschriebenen Signaturverfahren beruht die Sicherheit des in [27] beschriebenen Algorithmus nur auf der kryptographischen Stärke einer Hashfunktion und einer pseudozufälligen Funktionenfamilie. Insbesondere werden keine Annahmen zur Abwesenheit effizienter Lösungsalgorithmen für Probleme aus der algorithmischen Zahlentheorie wie das RSA-Problem oder die Berechnung diskreter Logarithmen benötigt. Es wird deshalb allgemein angenommen, dass Merklesignaturen im Gegensatz zu allen anderen in dieser Technischen Richtlinie empfohlenen Signaturverfahren auch gegen Angriffe unter Verwendung von Quantencomputern sicher bleiben würden.[2]

[2]Eine Diskussion der Quantencomputer-Sicherheit der Kollisionsresistenz von Hashfunktionen findet sich in [11].

Als Hashfunktionen sind alle in Tabelle 4.1 empfohlenen Hashverfahren geeignet. Die benötigte pseudozufällige Funktionenfamilie kann durch die HMAC-Konstruktion aus der verwendeten Hashfunktion konstruiert werden.

Für eine genaue Beschreibung des Verfahrens siehe [27].

Die generell geringen komplexitätstheoretischen Annahmen, die der Sicherheit von Merkle-Signaturen zugrundeliegen, lassen Merkle-Signaturen als eine gute Methode für die Erstellung langfristig sicherer Signaturen erscheinen. Dies gilt auch unter der Annahme, dass Angriffe durch Quantencomputer über den Zeitraum hinweg, in dem die Signatur gültig bleiben soll, keine Anwendung finden.

Anders als in den anderen in der vorliegenden Technischen Richtlinie beschriebenen Signaturverfahren kann bei Verwendung von Merkle-Signaturen mit einem gegebenen öffentlichen Schlüssel allerdings jeweils nur eine endliche Anzahl von Nachrichten authentifiziert werden. Außerdem ist die Rechenzeit zur Erzeugung des öffentlichen Schlüssels proportional zu dieser Anzahl zu authentisierender Nachrichten und damit vergleichsweise lang, wenn eine große Anzahl von Nachrichten ohne zwischenzeitliche Erzeugung und authentisierte Verteilung eines neuen öffentlichen Schlüssels signiert werden soll. Ergebnisse praktischer Experimente zur Effizienz aller Teilschritte (Schlüsselgenerierung, Signaturerzeugung, Signaturverifikation) des in [27] beschriebenen Verfahrens und zu den auftretenden Schlüssellängen und Signaturgrößen finden sich in Abschnitt 6 von [27].

5.2.5. Langfristige Beweiswerterhaltung für digitale Signaturen

Unabhängig von den vorliegenden Empfehlungen zu Verfahren und Schlüssellängen für digitale Signaturen und unabhängig von den entsprechenden Vorgaben zur qualifizierten elektronischen Signatur in [15] wird dazu geraten, die Möglichkeit künftiger Umstellungen der Systeme auf neue Signaturverfahren oder längere Signaturschlüssel schon bei der Entwicklung vorzusehen, 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 03125 (TR-ESOR) [26].
Ausdrücklich *nicht* empfohlen wird ein Einsatz von MAC-Verfahren zur langfristigen Datenauthentisierung, da dabei die prüfende Stelle Kenntnis des geheimen MAC-Schlüssels benötigt und das langfristige Risiko einer Kompromittierung des geheimen Schlüssels daher deutlich größer ist als bei digitalen Signaturen.

6. Instanzauthentisierung

Unter Instanzauthentisierung werden in dieser Technischen Richtlinie kryptographische Protokolle verstanden, in denen ein Beweisender einem Prüfer den Besitz eines Geheimnisses nachweist. Bei symmetrischen Verfahren ist dies ein symmetrischer Schlüssel, der vorab ausgetauscht werden muss. In asymmetrischen Verfahren zeigt der Beweisende, dass er im Besitz eines geheimen Schlüssels ist. Hier wird in der Regel eine PKI benötigt, damit der Prüfer den zugehörigen öffentlichen Schlüssel dem Beweisenden 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. Unter einer PIN (*Personal Identification Number*) wird hier ein nur aus den Ziffern 0-9 bestehendes Passwort verstanden.

Die Authentisierung sollte - wo das sinnvoll und möglich ist - gegenseitig erfolgen und kann mit einer Schlüsseleinigung einhergehen, um die Vertraulichkeit und Integrität einer anschließenden Kommunikation zu gewährleisten, siehe Kapitel 7 für empfohlene Schlüsselaustausch- und Schlüsseleinigungsverfahren und Abschnitt A.2 für empfohlene Protokolle, die beide Verfahren kombinieren.

Deshalb werden in diesem Kapitel 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 auch nur dort Empfehlungen für Schlüssellängen usw. angegeben.

6.1. Symmetrische Verfahren

Für den Nachweis des Beweisenden (B) gegenüber einem Prüfer (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 von 100 Bit erreicht, sollte r 100 Bit Min-Entropie besitzen. B berechnet dann mittels des gemeinsamen geheimen Schlüssels K eine Prüfsumme von r und schickt diese zurück an P. P prüft dann diese Prüfsumme. Solche Verfahren heißen auch *Challenge-Response-Verfahren*, siehe Tabelle 6.1 für eine schematische Darstellung.

Beweisender (B)		**Prüfer (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. Grund-

sätzlich können alle in Kapitel 2 empfohlenen Verschlüsselungsverfahren und alle in Abschnitt 5.1 empfohlenen MAC-Verfahren eingesetzt werden. Für empfohlene Bitlängen und Nebenbedingungen an die benutzten Zufallswerte siehe Abschnitt A.2.

6.2. Asymmetrische Verfahren

Wie schon im letzten Abschnitt werden auch für asymmetrische Verfahren Challenge-Response-Protokolle zur Instanzauthentisierung eingesetzt. Hier berechnet der Beweisende eine Prüfsumme zu einem vom Prüfer gesendeten Zufallswert r mit seinem geheimen Schlüssel. Der Prüfer verifiziert dann die Prüfsumme mit Hilfe des zugehörigen öffentlichen Schlüssels. Grundsätzlich können hierfür alle in Abschnitt 5.2 empfohlenen Verfahren eingesetzt werden. Für empfohlene Bitlängen und Nebenbedingungen an die benutzten Zufallswerte siehe ebenfalls Abschnitt A.2.

Bemerkung 14 Auch wenn die in Abschnitt 5.2 empfohlenen Signaturverfahren zur Datenauthentisierung auch zur Instanzauthentisierung genutzt werden können, sollte darauf geachtet werden, dass die eingesetzten Schlüssel verschieden sind. Das bedeutet, dass ein Schlüssel zur Erzeugung von Signaturen nicht zur Instanzauthentisierung eingesetzt werden sollte. Dies muss in den entsprechenden Zertifikaten für die öffentlichen Schlüssel kenntlich gemacht werden.

6.3. Passwortbasierte Verfahren

Passwörter zum Freischalten der auf kryptographischen Komponenten, wie zum Beispiel Signaturkarten, zur Verfügung gestellten kryptographischen Schlüssel sind meist kurz, damit sich der Inhaber der Komponente das Passwort auch merken kann. In vielen Situationen ist zudem der erlaubte Zeichensatz begrenzt auf die Ziffern 0-9. 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 Nebenbedingungen werden empfohlen:

1. Es wird grundsätzlich empfohlen, Passwörter mit einer Entropie von mindestens $log_2(10^6)$ Bits zu nutzen. Dies kann erreicht werden zum Beispiel durch eine ideal zufällige Vergabe sechsstelliger PINs (vergleiche auch [36], Abschnitt 4.3.3).
2. Die Anzahl der aufeinanderfolgenden erfolglosen Zugriffsversuche muss eng beschränkt werden. Bei einer Passwortentropie von $log_2(10^6)$ Bit wird eine Beschränkung auf drei Versuche empfohlen.

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

Bemerkung 15 Werden Zugriffs-Passwörter für kryptographische Komponenten nicht wenigstens annähernd ideal zufällig durch einen technischen Prozess erzeugt sondern durch den Nutzer gesetzt, dann wird eine Sensibilisierung des Nutzers bezüglich der Auswahl sicherer Passwörter dringend empfohlen. Es wird empfohlen, in diesem Fall von der Verwendung rein numerischer Passwörter (PINs) abzusehen. Für Passwörter, die über einem Alphabet gebildet werden, das mindestens die Buchstaben A-Z, a-z und 0-9 enthält, wird eine Länge von acht Zeichen empfohlen. Es wird weiter empfohlen, Maßnahmen zu treffen, die sehr naheliegende Passwörter (zum

Beispiel beliebige einzelne Wörter der Landessprache oder einer wichtigen Fremdsprache sowie Datumsangaben in naheliegenden Formaten) ausschließen. Für eine Einschätzung des Sicherheitsniveaus nutzergenerierter PINs und Passwörter verweisen wir auf [73], Tabelle A.1.

Bemerkung 16 In manchen Anwendungen kann nach Abwägung aller 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 eng beschränkt werden, die genauen Beschränkungen sind abhängig von der Anwendung. Die Restrisiken sollten gründlich dokumentiert werden. Es wird empfohlen, in Situationen, in denen dies anwendbar ist, den berechtigten Nutzer über erfolgte unberechtigte Zugriffsversuche zu informieren, auch wenn die Komponente in deren Folge nicht gesperrt wurde.

Bemerkung 17 (*i*) Um Denial-of-Service Attacken oder eine versehentliche Sperrung der Komponente zu verhindern, muss ein Mechanismus zum Aufheben der Sperrung vorgesehen sein. Die Entropie des Schlüssels zur Entsperrung (englisch *Personal Unblocking Key*, kurz PUK) sollte mindestens 100 Bit betragen, wenn Offline-Attacken denkbar sind.

(*ii*) Wenn keine Offline-Angriffe auf die PUK möglich sind, wird empfohlen, eine PUK mit mindestens 32 Bit 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 oben ausgesprochene allgemeine Empfehlung von mindestens etwa 20 Bit Entropie für das in einem passwortbasierten Authentisierungsverfahren verwendete Passwort gilt natürlich 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 sensitiver Information verschafft), wird empfohlen, Passwörter über ein Verfahren auszuwählen, das mindestens 100 Bit Entropie liefert. Es wird für den Zugang zu Daten oder für die Authentisierung von Transaktionen mit hohem Schutzbedarf grundsätzlich von einer Ein-Faktor-Authentisierung abgeraten. Empfohlen wird in dieser Situation eine Zwei-Faktor-Authentisierung durch Wissen (Kenntnis eines Passwortes) und Besitz (einer sicheren Hardwarekomponente).

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

Für kontaktbehaftete Chipkarten ist das Verfahren sehr einfach. Das Passwort wird auf dem Pinpad des Kartenlesers eingegeben und ohne kryptographische Absicherung zur Chipkarte übertragen. Obwohl hier also auf Seiten des Kartenlesers keine kryptographischen Mechanismen zum Einsatz kommen, sollte dabei ein zertifizierter Kartenleser eingesetzt werden, um Angriffe durch Manipulationen im Kartenleser selbst zu verhindern.

Bei kontaktlosen Chipkarten kann die Kommunikation zwischen Kartenleser und Chipkarte auch noch aus einiger Entfernung mitgelesen werden. Hier 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 [22].

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 18 Auch hier muss die Anzahl der Versuche beschränkt sein. Empfohlen wird, nach drei erfolglosen Versuchen die Chipkarte zu sperren. Die sonstigen Bemerkungen aus Abschnitt 6.3.1 gelten entsprechend.

7. Schlüsseleinigungsverfahren, Schlüsseltransportverfahren und Key-Update

Schlüsseleinigungsverfahren dienen dem Austausch eines Verschlüsselungsschlüssels über einen unsicheren Kanal. Diese Verfahren müssen unbedingt mit Instanzauthentisierungsverfahren kombiniert werden. Ansonsten besteht keine Möglichkeit zu entscheiden, mit welcher Partei die Schlüsseleinigung durchgeführt wird (Datenauthentisierung allein genügt hier nicht, da ein Angreifer eine in der Vergangenheit durchgeführte Kommunikation mitgeschnitten haben könnte um die aufgezeichneten Daten für einen Angriff zu nutzen). Aus diesem Grund geben wir, wie schon in Kapitel 6, in diesem Kapitel nur ganz allgemeine Ideen für Schlüsseleinigungsverfahren an und verweisen für konkrete Verfahren, d.h. für Schlüsseleinigungsverfahren, die auch eine Instanzauthentisierung beinhalten, auf Abschnitt A.2.

Nach erfolgreicher Schlüsseleinigung befinden sich beide Parteien im Besitz eines gemeinsamen Geheimnisses. Für empfohlene Verfahren zur Generierung symmetrischer Schlüssel aus diesem Geheimnis siehe Abschnitt B.1. Im Wesentlichen wird hier die Verwendung einer Schlüsselableitungsfunktion empfohlen.

In manchen Situationen kann es hier sinnvoll sein, in die Schlüsselableitungsfunktion ein vorverteiltes Geheimnis eingehen zu lassen. Damit kann zum Beispiel eine Separierung verschiedener Benutzergruppen erreicht werden, auch eine zusätzliche Verteidigungslinie gegen Angriffe auf das Schlüsseleinigungsverfahren kann auf diese Weise eingezogen werden. Hinsichtlich einer Separierung verschiedener Benutzergruppen kann es überdies sinnvoll sein, weitere öffentliche Daten, die spezifisch für beide Kommunikationspartner sind, in die Schlüsselableitung eingehen zu lassen.

Es wird empfohlen, nur Schlüsseleinigungsverfahren zu verwenden, in denen beide Kommunikationspartner gleiche Anteile für den neuen Schlüssel bereitstellen. Beide Seiten sollten dabei mindestens 100 Bit Entropie beitragen. Bei der Auswahl eines Schlüsseleinigungsverfahrens für eine bestimmte Anwendung sollte auch in Betracht gezogen werden, ob in dem gewählten Protokoll eine Seite eine größere Kontrolle über das Schlüsselmaterial hat als die andere und ob eine solche Asymmetrie in der gegebenen Anwendung sicherheitsrelevante Auswirkungen hat.

Neben Schlüsseleinigungsverfahren sind auch Schlüsseltransportverfahren von praktischer Bedeutung. Dabei werden geheime Schlüsseldaten von einer Stelle erzeugt und gesichert zu einem oder mehreren Empfängern transportiert. Die Empfänger haben hier keine Kontrolle über die verteilten Sitzungsschlüssel. Die erzeugende Stelle kann eine vertrauenswürdige Drittpartei sein oder einer der Kommunikationsteilnehmer. Im letzteren Fall wird empfohlen, dass alle Teilnehmer nur jeweils selbst erzeugte Schlüssel zur Übertragung eigener sensibler Daten nutzen.

Ebenfalls behandelt werden sollen in diesem Abschnitt Key-Update-Verfahren. Hier teilen zwei Parteien bereits ein gemeinsames Geheimnis und leiten daraus am Ende einer Schlüsselperiode einen neuen Schlüssel ab. Dies kann erreicht werden durch Ableitung neuer Sitzungsschlüssel aus einem dauerhaften Masterschlüssel oder auch durch eine Update-Transformation, die aus dem aktuellen Schlüssel und gegebenenfalls weiteren Daten einen neuen Schlüssel generiert.

Vorbemerkung: Asymmetrische versus symmetrische Schlüsseleinigungsverfahren
Mit asymmetrischen Schlüsseleinigungsverfahren sind Sicherheitseigenschaften erreichbar, die allein unter Verwendung symmetrischer Kryptographie nicht realisierbar sind. Zum Beispiel ha-

ben beide empfohlenen asymmetrischen Schlüsseleinigungsverfahren die Eigenschaft der *(Perfect) Forward Secrecy*. Dies bedeutet, dass ein Angreifer, der alle gegebenenfalls vorhandenen langfristigen Geheimnisse beider Kommunikationsteilnehmer kennt[1], dennoch nicht den während einer unkompromittierten Protokollausführung ausgehandelten Schlüssel ermitteln kann, falls er das dem verwendeten asymmetrischen Verfahren zugrundeliegende mathematische Problem (in den hier vorgestellten Verfahren das Diffie-Hellman-Problem) nicht effizient lösen kann. Im Vergleich kann in symmetrischen Schlüsseleinigungsverfahren höchstens erreicht werden, dass ein Angreifer, der alle Langzeitgeheimnisse beider Teilnehmer kennt, die Ergebnisse *vergangener* korrekt durchgeführter Schlüsseleinigungen nicht ermitteln kann.[2]

7.1. Symmetrische Verfahren

Schlüsseltransport Grundsätzlich können alle der in Kapitel 2 empfohlenen symmetrischen Verschlüsselungsverfahren zum Transport von Sitzungsschlüsseln verwendet werden. Es wird empfohlen, ein in Kapitel 2 empfohlenes Verschlüsselungsverfahren mit einem MAC aus Abschnitt 5.1 zu kombinieren (im Encrypt-then-MAC-Modus), um eine manipulationssichere Übertragung des Schlüsselmaterials zu erreichen.

Schlüsseleinigung Auch Schlüsseleinigungsverfahren lassen sich rein auf Basis symmetrischer Verfahren realisieren, wenn die Existenz eines gemeinsamen langfristigen Geheimnisses vorausgesetzt werden kann. Key Establishment Mechanism 5 aus [46] 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 der in Abschnitt B.1 empfohlene Mechanismus verwendet werden.

Key Update In manchen Situationen kann es nützlich sein, die in einem kryptographischen System genutzten Schlüssel bei allen Beteiligten synchron auszutauschen, ohne dass weitere Kommunikation stattfindet. In diesem Fall können Key-Update-Mechanismen zum Einsatz kommen. Wir gehen im Folgenden davon aus, dass der Masterschlüssel K_t eines Kryptosystems zum Zeitpunkt t über ein solches Verfahren ersetzt werden soll. Für allgemeine Anwendungen empfehlen wir das folgende Verfahren:

1. $K_{t+1} := \mathrm{KDF}(s, \mathrm{Label}, \mathrm{Context}, L, K_t)$.
2. Hier ist KDF eine kryptographische Schlüsselableitungsfunktion nach [72]. s ist der dabei im Extraktionsschritt genutzte Salt-Wert. Label und Context gehen in dem in [72] vorgesehenen Schlüsselexpansionsschritt nach [75] 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 geht ebenfalls im Expansionsschritt ein.

Es 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} nach dem beschriebenen Verfahren. Es wird empfohlen, dies durch Verwendung geeigneter Label-Werte zu erzwingen. Es wird weiter empfohlen, in Label oder Context mindestens auch die

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

[2] *Merkle Puzzles* sind hier insofern ausgenommen, als es sich dabei um ein Schlüsseleinigungsverfahren mit öffentlichen Schlüsseln unter ausschliesslicher Nutzung symmetrischer Primitive handelt [63]. Dieses Verfahren hat aber nur akademische Bedeutung.

Kryptoperiode t zu codieren. Als zusätzliche Maßnahme kann es 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 [72, 75] verwiesen.

7.2. Asymmetrische Verfahren

Grundsätzlich können alle der in Kapitel 3 empfohlenen asymmetrischen Verschlüsselungsverfahren zum Transport neuer Sitzungsschlüssel verwendet werden.

Als asymmetrische Schlüsselaushandlungsverfahren werden empfohlen:

1. Diffie-Hellman, siehe [66],
2. EC Diffie-Hellman (ECKA-DH), siehe [23].

Tabelle 7.1.: Empfohlene asymmetrische Schlüsseleinigungsverfahren

Für die Spezifizierung sind folgende Algorithmen festzulegen:

1. Ein Algorithmus zum Festlegen der Systemparameter.
2. Ein Algorithmus zur Schlüsseleinigung.

7.2.1. Diffie-Hellman

Die Sicherheit dieses Verfahrens beruht auf der vermuteten Schwierigkeit des Diffie-Hellman-Problems in Gruppen $\mathbb{F}_p$, p eine Primzahl.

Systemparameter

1. Wähle eine Primzahl p.
2. Wähle ein Element $g \in \mathbb{F}_p^*$ mit $\mathrm{ord}(g)$ prim und $q := \mathrm{ord}(g) \geq 2^{224}$ (es wird $\mathrm{ord}(g) \geq 2^{250}$ empfohlen im Falle eines Einsatzzeitraums nach 2015).

Das Paar (p, g, q) muss vorab **authentisch** zwischen den Kommunikationspartnern ausgetauscht werden.

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. Das ausgehandelte Geheimnis ist dann g^{xy}. Ein Mechanismus für eine nachfolgende Schlüsselableitung aus dem gemeinsamen Geheimnis wird in Abschnitt B.1 empfohlen.

Schlüssellänge Die Länge von p sollte mindestens 2000 Bit betragen. Fußnote b) zu Tabelle 3.1 und Bemerkung 4 aus Kapitel 3 gelten entsprechend.

Bemerkungen zur Implementierung Bei der Implementierung des Diffie-Hellman-Protokolls ist eine Reihe von Implementierungsfehlern weit verbreitet. Auf einige dieser Implementierungsprobleme wird in [77] eingegangen. Es wird empfohlen, insbesondere Abschnitt 7 von [77] zu beachten.

7.2.2. EC Diffie-Hellman

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

Systemparameter Wähle kryptographisch starke EC-Systemparameter (p, a, b, P, q, i), siehe Abschnitt B.3. Die damit definierte elliptische Kurve bezeichnen wir mit C, die durch P erzeugte zyklische Untergruppe wird in diesem Abschnitt mit $\mathcal{G}$ bezeichnet.

Die Systemparameter 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. Das ausgehandelte Geheimnis ist dann $xy \cdot P$.

Ein Mechanismus für eine nachfolgende Schlüsselableitung aus dem gemeinsamen Geheimnis wird in Abschnitt B.1 empfohlen.

Soweit das möglich ist, wird empfohlen, auf beiden Seiten der Schlüsselvereinbarung zu testen, ob die Punkte Q_A und Q_B protokollgerecht gewählt worden sind und das Protokoll bei negativem Testergebnis abzubrechen. Bei korrekter Ausführung des oben wiedergegebenen 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 [23].

Schlüssellänge Die Länge von q sollte mindestens 224 Bit betragen. Für einen Einsatzzeitraum nach 2015 werden 250 Bit empfohlen.

Bemerkungen zur Implementierung Bei der Implementierung des Diffie-Hellman-Protokolls ist eine Reihe von Implementierungsfehlern weit verbreitet. Auf einige dieser Implementierungsprobleme wird in [77] eingegangen. Es wird empfohlen, insbesondere Abschnitt 7 von [77] zu beachten. Auch die Hinweise in Abschnitt 4.3 von [23] und die AIS46 [4] sind zu berücksichtigen.

8. Secret Sharing

Häufig müssen kryptographische Schlüssel über einen langen Zeitraum gespeichert werden. Dies erfordert insbesondere, dass Kopien dieser Schlüssel angelegt werden müssen, um einen Verlust der Schlüssel zu verhindern. Je mehr Kopien allerdings generiert werden, um so größer ist die Wahrscheinlichkeit dafür, dass das zu schützende Geheimnis kompromitiert wird.

Wir geben deshalb in diesem Kapitel ein Verfahren an, welches erlaubt, ein Geheimnis, wie zum Beispiel 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, siehe [84]. Wir nehmen im Folgenden an, dass das zu verteilende Geheimnis ein Schlüssel K der Bitlänge r ist: $K = (k_0, \ldots, k_{r-1}) \in \{0,1\}^r$.

Zur Berechnung der verteilten Geheimnisse auf n Benutzer, so dass t Benutzer das Geheimnis K wieder rekonstruieren können, geht man wie folgt vor:

1. Wähle eine Primzahl $p \geq \max(2^r, n+1)$ und setze $a_0 := \sum_{i=0}^{r-1} k_i \cdot 2^i$.
2. Wähle unabhängig voneinander $t-1$ zufällige Werte $a_1, \ldots, a_{t-1} \in \{0, 1, \ldots, p-1\}$. Die Werte $a_0, a_1, \ldots, a_{t-1}$ definieren dann ein zufälliges Polynom
$$f(x) = \sum_{j=0}^{t-1} a_j x^j$$
über $\mathbb{F}_p$, für das $f(0) = a_0 = \sum_{i=0}^{r-1} k_i \cdot 2^i$ gilt.
3. Berechne die Werte $K_i := f(i)$ für alle $i \in \{1, \ldots, n\}$.

Tabelle 8.1.: Berechnung der Teilgeheimnisse im Shamir Secret-Sharing-Verfahren

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

Bemerkung 19 Die Koeffizienten $a_0, \ldots, a_{t-1}$ eines unbekannten Polynoms f vom Grad $t-1$ können mit Hilfe der sogenannten *Lagrange-Interpolations-Formel* aus t Punkten $(x_i, f(x_i))$ wie folgt gefunden werden:

$$f(x) = \sum_{i=1}^{t} f(x_i) \prod_{1 \leq j \leq t, i \neq j} \frac{x - x_j}{x_i - x_j}.$$

Insbesondere lässt sich auf diese Weise $a_0 = f(0)$ (und damit K) aus t gegebenen Punkten berechnen. Dies ist die Grundlage für das obige Verfahren.

Um nun aus t Teilgeheimnissen $K_{j_1}, \ldots, K_{j_t}$ (mit paarweise verschiedenen j_l) das Geheimnis K zu rekonstruieren, berechnet man $a_0 = \sum_{i=0}^{r-1} k_i \cdot 2^i$ wie folgt (man beachte, dass in den Tabellen 8.1 und 8.2 jeweils in $\mathbb{F}_p$, d.h. modulo p gerechnet wird):

1. Berechne für alle $j \in \{j_1, \ldots, j_t\}$ den Wert $c_j = \prod_{1 \le l \le t, j_l \neq j} \frac{j_l}{j_l - j}$.
2. Berechne $K = \sum_{l=1}^{t} c_{j_l} K_{j_l}$.

Tabelle 8.2.: Zusammensetzen der Teilgeheimnisse im Shamir Secret-Sharing-Verfahren

Bemerkung 20 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 Schemas hängt daher über die angegebene Bedingung hinaus nicht von weiteren Sicherheitsparametern ab. Allerdings muss durch organisatorische und technische Maßnahmen sichergestellt werden, dass ein Angreifer nicht von t Teilgeheimnissen Kenntnis 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.

9. Zufallszahlengeneratoren

Eine große Zahl kryptographischer Anwendungen benötigen Zufallszahlen, etwa zur Erzeugung kryptographischer Langzeitschlüssel, Ephemeralschlüssel, Systemparameter oder zur Instanzauthentisierung. Dies betrifft symmetrische und asymmetrische Verschlüsselungsverfahren ebenso wie Signatur- und Authentisierungsverfahren und Paddingverfahren.

Für spezielle Anwendungen enthält Anhang B Algorithmen, mit denen man aus den Ausgabewerten eines Zufallszahlengenerators Zufallswerte mit gewünschten Eigenschaften (z.B. gleichverteilt auf $\{0, \ldots, q-1\}$) berechnen kann.

Für die meisten kryptographischen Anwendungen sind Unvorhersagbarkeit und Geheimhaltung der Zufallszahlen bzw. der aus ihnen abgeleiteten Werte unverzichtbar. Selbst wenn ein Angreifer lange Teilfolgen von Zufallszahlen kennt, soll ihn dies nicht in die Lage versetzen, Vorgänger oder Nachfolger zu bestimmen. Ungeeignete Zufallszahlengeneratoren können grundsätzlich starke kryptographische Mechanismen entscheidend schwächen. Daher müssen für kryptographische Anwendungen geeignete Zufallszahlengeneratoren eingesetzt werden.

Im deutschen Zertifizierungschema sind die AIS 20 [2] (für deterministische Zufallszahlengeneratoren) und AIS 31 [3] (für physikalische Zufallszahlengeneratoren) verbindlich. Von zentraler Bedeutung ist die gemeinsame mathematisch-technische Anlage [57]. Referenz [57] löst die Vorgängerdokumente [83] und [56] ab.

Die mathematisch-technische Anlage [57] definiert Funktionalitätsklassen für physikalische Zufallszahlengeneratoren (PTG.1 – PTG.3), für deterministische Zufallszahlengeneratoren (DRG.1 – DRG.4) und für nicht-physikalische nicht-deterministische Zufallszahlengeneratoren (NTG.1). Die alten Funktionalitätsklassen K2, K3, K4, P1 und P2 aus [83] und [56] wurden (unter neuem Namen) im Wesentlichen (mit partiell höheren Anforderungen) beibehalten, und es sind neue Funktionalitätsklassen hinzugekommen: hybride deterministische Zufallszahlengeneratoren (Funktionalitätsklasse DRG.4), hybride physikalische Zufallszahlengeneratoren (Funktionalitätsklasse PTG.3) und nicht-physikalische nicht-deterministische Zufallszahlengeneratoren (Funktionalitätsklasse NTG.1). Darüber hinaus erläutert [57] den mathematischen Hintergrund und illustriert die Konzepte an zahlreichen Beispielen.

Folgend werden in Stichpunkten die in den einzelnen Abschnitten dieses Kapitels enthaltenen Empfehlungen zum Einsatz von Zufallsgeneratoren in allgemeinen kryptographischen Anwendungen zusammengefasst:

- Bei Einsatz eines physikalischen Zufallsgenerators wird grundsätzlich empfohlen, einen PTG.3-Generator einzusetzen. Dies gilt besonders für die Erzeugung von Ephemeralschlüsseln bei der Erzeugung digitaler Signaturen und bei Diffie-Hellman-basierter Schlüsselaushandlung. In Kontexten, in denen für die Zufallszahlenerzeugung der Einsatz einer zertifizierten Kryptokomponente erforderlich ist, gilt diese Empfehlung natürlich nur bei Verfügbarkeit entsprechend zertifizierter Komponenten. In anderen Kontexten 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 bestimmte kryptographische Anwendungen können auch PTG.2-Generatoren verwendet werden. Dies ist dann der Fall, wenn der Vorteil, der für einen Angreifer aus der Existenz geringfügiger Schiefen oder Abhängigkeiten in der Verteilung der erzeugten Zufallswerte entsteht, beweisbar gering ist, zum Beispiel bei der Erzeugung symmetrischer

Sitzungsschlüssel.

- Grundsätzlich haben PTG.3-Generatoren und DRG.4-Generatoren verglichen mit PTG.2-Generatoren und DRG.3-Generatoren den Vorteil einer verbesserten Resistenz gegen Seitenkanalattacken und Fault-Angriffe. Im Fall eines PTG.3-Generators zum Beispiel bedeutet der dauernde 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 erschwert zusammenführen kann.

- Bei Einsatz eines deterministischen Zufallsgenerators wird empfohlen, einen DRG.3-Generator oder einen DRG.4-Generator einzusetzen. Es wird empfohlen, den Seed aus einer physikalischen Zufallsquelle der Klasse PTG.2 oder PTG.3 zu generieren. Ist keine solche Zufallsquelle vorhanden, dann kann unter Umständen auch die Verwendung eines nicht-physikalischen, nicht-deterministischen Zufallsgenerators in Erwägung gezogen werden, Details siehe Abschnitt 9.3 und Abschnitt 9.5. In jedem Fall ist darauf zu achten, dass der Seed mindestens 100 Bit Min-Entropie oder mindestens 125 Bit Shannon-Entropie aufweist.

- Diese Anforderungen an die Min-Entropie des Seeds eines deterministischen Zufallsgenerators erhöhen sich natürlich entsprechend, falls für ein kryptographisches System insgesamt ein Sicherheitsniveau von mehr als 100 Bit angestrebt wird. Im allgemeinen Fall wird für eine Systemsicherheit von n Bit eine Min-Entropie des RNG-Seeds von n Bit verlangt.

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

9.1. Physikalische Zufallszahlengeneratoren

Physikalische Zufallszahlengeneratoren nutzen dedizierte Hardware (üblicherweise eine elektronische Schaltung) oder physikalische Experimente, um hieraus 'echten' Zufall, d.h. unvorhersagbare Zufallszahlen, zu erzeugen. Physikalische Zufallszahlengeneratoren nutzen Effekte aus, die z.B. auf elektromagnetischen, elektromechanischen oder quantenmechanischen Effekten beruhen. Ein Angreifer sollte auch bei Kenntnis von Zufallszahlenteilfolgen (und genauer Kenntnis des Zufallszahlengenerators) in Bezug auf Vorgänger und Nachfolger nur einen vernachlässigbaren (im Idealfall gar keinen) Vorteil gegenüber blindem Raten der Zufallszahlen besitzen.

Häufig ist eine deterministische Nachbearbeitung der 'Rauschrohdaten' (üblicherweise digitalisierte Rauschsignale) notwendig, um etwaig vorhandene Schiefen oder Abhängigkeiten zu beseitigen.

Wird ein physikalischer Zufallsgenerator eingesetzt, wird grundsätzlich empfohlen, einen PTG.3-Generator im Sinne der AIS 31 einzusetzen (vgl. [57], Kap. 4). Dies trifft insbesondere auf Anwendungen zu, bei denen ein Angreifer zumindest prinzipiell Informationen über verschiedene Zufallszahlen zusammenführen kann. Für bestimmte Anwendungen genügt auch ein Zufallsgenerator der Klasse PTG.2. Sofern eine Implementierung des Zufallsgenerators in einer zertifizierten Kryptokomponente erforderlich ist, gilt die Empfehlung einer Verwendung eines PTG.3-Generators natürlich nur, soweit geeignete zertifizierte Komponenten existieren.

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 [57]. Grob gesagt, muss die Nachbearbeitung einen DRG.3-kompatiblen deterministischen Zufallsgenerator implementieren und es muss dem

internen Zustand des Zufallsgenerators jederzeit mindestens soviel neue Entropie durch einen Zufallsgenerator der Klasse PTG.2 zugeführt werden, wie durch die kryptographische Anwendung verlangt wird.

Zufallszahlen aus PTG.2-konformen Zufallszahlengeneratoren besitzen hohe Entropie, können aber gewisse Schiefen und / oder Abhängigkeiten aufweisen. Ob in einer bestimmten Anwendung ein PTG.2-Generator ausreicht, sollte mit einem Experten geklärt werden.

Vereinfacht gesagt, müssen PTG.2- bzw. PTG.3-konforme Zufallszahlengeneratoren folgende Eigenschaften erfüllen:

1. Die stochastischen Eigenschaften der Zufallszahlen lassen sich hinreichend gut durch ein stochastisches Modell beschreiben. Auf der Basis dieses stochastischen Modells kann man die Entropie der Zufallszahlen zuverlässig abschätzen.

2. Der durchschnittliche Entropiezuwachs pro Zufallsbit liegt oberhalb einer gegebenen Mindestschranke (nahe bei 1).

3. Die digitalisierten Rauschsignale werden im laufenden Betrieb statistischen Tests unterzogen, die geeignet sind, nicht akzeptable statistische Defekte oder Verschlechterungen der statistischen Eigenschaften in angemessener Zeit zu erkennen.

4. Ein Totalausfall der Rauschquelle wird de facto sofort erkannt. Es dürfen keine Zufallszahlen ausgegeben werden, die nach einem Totalausfall der Rauschquelle erzeugt wurden.

5. Wird ein Totalausfall der Rauschquelle oder nicht akzeptable statistische Defekte der Zufallszahlen entdeckt, führt dies zu einem Rauschalarm. Auf einen Rauschalarm folgt eine definierte, geeignete Reaktion (z.B. Stilllegen der Rauschquelle).

6. (nur PTG.3-konforme Zufallszahlengeneratoren) Eine (ggf. zusätzliche) starke kryptographische Nachbearbeitung sorgt dafür, 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. Hybride physikalische Zufallszahlengeneratoren der Funktionalitätsklasse PTG.3 besitzen neben einer starken Rauschquelle eine starke kryptographische Nachbearbeitung mit Gedächtnis. Eine typische Realisierung besteht darin, dass Zufallszahlen eines PTG.2-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, sich bei Bedarf in diesem Zusammenhang frühzeitig an Experten auf diesem Gebiet zu wenden.**

9.2. Deterministische Zufallszahlengeneratoren

Deterministische bzw. Pseudozufallszahlengeneratoren können aus einem Zufallswert fester Länge, dem sogenannten *Seed*, eine pseudozufällige Bitfolge praktisch beliebiger Länge berechnen. Dazu wird der innere Zustand des Pseudozufallszahlengenerators zunächst mit dem Seed initialisiert. In die Berechnung können auch öffentlich bekannte Parameter eingehen. In jedem Schritt wird der innere Zustand dann 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

(reseed / seed update). Dabei können unterschiedliche Aufrufschemata zum Einsatz kommen (zum Beispiel regelmäßig oder auf Anfrage der Applikation).

Der innere Zustand eines deterministischen Zufallszahlengenerators muss zuverlässig gegen Auslesen und Manipulation geschützt werden.

Wird ein deterministischer Zufallsgenerator verwendet, dann wird empfohlen, einen DRG.3- oder DRG.4-konformen Zufallsgenerator gegen das Angriffspotential HOCH im Sinne der AIS 20 einzusetzen (vgl. [57]). Vereinfacht gesagt, bedeutet dies unter anderem:

1. Es ist einem Angreifer nicht praktisch möglich, zu einer bekannten Zufallszahlenfolge Vorgänger oder Nachfolger zu berechnen oder mit signifikant höherer Wahrscheinlichkeit zu erraten, als dies ohne Kenntnis dieser Teilfolge möglich wäre.

2. Es ist einem Angreifer nicht praktisch möglich, aus Kenntnis eines inneren Zustandes zuvor ausgegebene Zufallszahlen zu berechnen oder mit signifikant höherer Wahrscheinlichkeit zu erraten, als dies ohne Kenntnis des inneren Zustands möglich wäre.

3. (nur DRG.4-konforme Zufallszahlengeneratoren) Selbst wenn ein Angreifer den aktuellen inneren Zustand kennt, ist ihm nicht praktisch möglich, Zufallszahlen, die nach dem nächsten reseed / seed update 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 gewisse Vorteile auf gegenüber DRG.3-konformen Zufallsgeneratoren.

Die Entropie des Seed muss mindestens 100 Bit betragen (Min-Entropie). 125 Bit Shannon-Entropie sind ebenfalls hinreichend.

9.3. Nicht-physikalische nicht-deterministische Zufallszahlengeneratoren

Für viele kryptographische Anwendungen, etwa im E-Business oder E-Government, steht weder ein physikalischer noch ein deterministischer Zufallszahlengenerator zur Verfügung, da sie im Allgemeinen auf Computern ohne zusätzliche kryptographische Hardware ausgeführt werden. Stattdessen werden in aller Regel nicht-physikalische nicht-deterministische Zufallszahlengeneratoren (NPTRNG) verwendet.

Wie physikalische Zufallszahlengeneratoren erzeugen auch nicht-physikalische nicht-deterministische Zufallszahlengeneratoren 'echt zufällige' Zufallszahlen. Allerdings nutzen sie hierfür keine dedizierte Hardware, sondern Systemressourcen (Systemzeit, RAM-Inhalte usw.) und / oder Nutzerinteraktion (Tastatureingaben, Mausbewegung usw.). Nicht-physikalische nicht-deterministische Zufallszahlengeneratoren werden üblicherweise auf Computern eingesetzt, die nicht speziell für kryptographische Anwendungen entwickelt wurden, zum Beispiel allen verbreiteten Typen von PCs, Laptops oder Smartphones. Wie physikalische Zufallszahlengeneratoren setzen sie auf Sicherheit im informationstheoretischen Sinn durch hinreichend viel Entropie.

Ein typisches Vorgehen ist das Folgende: Es werden lange Bitstrings von 'zufälligen Daten' (oder genauer: von nicht-deterministischen Daten) erzeugt, wobei die Entropie pro Bit normalerweise eher gering ist. Dieser Bitstring wird mit einem inneren Zustand vermischt. Aus dem inneren Zustand werden Zufallszahlen errechnet und dann ausgegeben. Der bekannteste Vertreter der nicht-physikalischen nicht-deterministischen Zufallszahlengeneratoren ist /dev/random. In der mathematisch-technischen Anlage [57] wird eine Funktionalitätsklasse (NTG.1) definiert.

[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 100 Bit Min-Entropie erzeugt werden.

Vereinfacht gesagt, bedeutet dies unter anderem:

1. Die Entropie des inneren Zustands wird geschätzt. Wird eine Zufallszahl ausgegeben, wird der Entropiezähler entsprechend reduziert.
2. Zufallszahlen dürfen nur ausgegeben werden, wenn der Wert des Entropiezählers groß genug ist.
3. Es ist einem Angreifer nicht praktisch möglich, aus Kenntnis des inneren Zustandes und der zuvor für Seed-updates verwendete Zufalls-Bitstrings frühere Zufallszahlen zu berechnen oder mit signifikant höherer Wahrscheinlichkeit zu erraten, als dies ohne Kenntnis des inneren Zustands und der Bitstrings möglich wäre.

Es ist für NPTRNG von entscheidender Bedeutung, dass die durch den Zufallsgenerator 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 [81]. In diesem Fall kann der Output eines NPTRNG im Extremfall vollständig vorhersagbar werden, wenn das System zweimal aus dem gleichen Systemabbild heraus gestartet wird und alle Entropiequellen des virtuellen Systems vom Wirtsrechner kontrolliert werden.
Bei der geplanten Verwendung eines NPTRNG als alleiniger oder als wesentlichster Zufallsquelle für ein System, das für die Verarbeitung sensibler Daten bestimmt ist, sollte unbedingt ein Experte hinzugezogen werden.

9.4. 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 deterministischen Anteils, welcher der Klasse DRG.3 angehört. Während der Nutzung des Zufallszahlengenerators wird zudem immer wieder neuer Zufall hinzugefügt. Dies kann (z.B.) 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 potentieller Angreifer zumindest prinzipiell Informationen über viele Zufallszahlen kombinieren kann (z.B. Ephemeralschlüssel), sollte ein physikalischer Zufallszahlengenerator der Funktionalitätsklasse PTG.3 angehören.

Die Ableitung von Signaturschlüsseln, Ephemeralschlüsseln und Primzahlen (für RSA) o.ä. aus den erzeugten Zufallszahlen soll mit geeigneten Algorithmen erfolgen (zu elliptischen Kurven vgl. [4], Abschitte 5.2 und 5.5.1). Vereinfacht gesagt, sollte einem potentiellen 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 derselben Wahrscheinlichkeit auf, und verschiedene Zufallszahlen sollten zumindest keine praktisch ausnutzbaren Zusammenhänge aufweisen. Ebenso wie die Signaturalgorithmen kann auch die Erzeugung geheim zu haltender Signaturschlüssel, Ephemeralschlüssel und Primzahlen Ziel von Seitenkanalangriffen werden ([42, 4] etc.). Dieser Aspekt wird [57] explizit angesprochen.

Die für sicherheitskritische Anwendungen relevanten Funktionaliätsklassen aus [83] und [56] wurden in [57] unter neuem Namen und partiell gestiegenen Anforderungen im Wesentlichen

beibehalten (P2 → PTG.2, K3 → DRG.2, K4 → DRG.3). Außerdem sind neue Funktionalitätsklassen hinzugekommen (PTG.3, DRG.4, NTG.1).

9.5. Seedgenerierung für deterministische Zufallszahlengeneratoren

Für die Initialisierung eines deterministischen Zufallszahlengenerators wird ein Seed mit hinreichend hoher Entropie benötigt. Der 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 dieser Zufallsgenerator wurde zumindest nicht einer gründlichen herstellerunabhängigen Zertifizierung unterzogen. Hier bietet sich die Verwendung eines nicht-physikalischen nicht-deterministischen Zufallzahlengenerators an.

Hierfür geeignet sind NTG.1-konforme Zufallszahlengeneratoren (hohes Angriffspotential). Zur Zeit gibt es noch keine NTG.1-zertifizierte Zufallszahlengeneratoren. Daher geben wir unten für die zwei wichtigsten PC-Betriebssysteme geeignete Verfahren zur Seedgenerierung an.

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 der Benutzerin bzw. 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 also die Existenz von z.B. Viren oder Trojanern auf diesem Rechner aus. Benutzerinnen und Benutzer über diese Risiken aufgeklärt werden.

9.5.1. GNU/Linux

Folgendes Verfahren wird zur Seedgenerierung unter dem Betriebsystem GNU/Linux empfohlen.

/dev/random (in der Kernelversion 2.6.21.5 sowie 3.2, 3.5, 3.6 und 3.7)

Tabelle 9.1.: Empfohlenes Verfahren zur Seedgenerierung unter GNU/Linux

Bemerkung 21 Die Funktion /dev/random wurde bisher nur in der Kernelversion 2.6.21.5 sowie 3.2, 3.5, 3.6 und 3.7 vom BSI untersucht und bei einer Anwendung in typischen PC-ähnlichen Systemen als geeignet bewertet. Solange bei den darauffolgenden Kernelversionen keine Schwachstellen bekannt sind, darf die Funktion /dev/random auch für diese aktuelleren Kernel verwendet werden. Für weitere, aktualisierte Erkenntnisse betreffend die kryptographische Einschätzung von /dev/random in verschiedenen Versionen des Linux-Kernels verweisen wir auf [91].

Bemerkung 22 Prinzipiell sollte das empfohlene Verfahren auch für Linux-Systeme funktionieren, die über eine relativ geringe Zahl von Entropiequellen verfügen, zum Beispiel eingebettete Systeme. Da /dev/random blockiert, wenn der interne Entropiepool ausgeschöpft ist, ist es in diesem Fall allerdings möglich, dass die Zufallszahlenerzeugung sehr langsam wird.

Bemerkung 23 Die Nutzung von /dev/urandom kann in manchen Kontexten problematisch sein [43]. Generell wird dringend davon abgeraten, nicht-deterministische nicht-physikalische Zufallsgeneratoren unter Bedingungen einzusetzen, in denen nicht klar ist, ob die zu einem sicheren Betrieb notwendigen Annahmen zum Vorhandensein echter Entropiequellen zutreffen.

9.5.2. Windows

Im Gegensatz zum System GNU/Linux gibt es für das Betriebssystem Windows derzeit keine vom BSI untersuchte Funktion, die hinreichend große Entropie gewährleistet. Zur Erzeugung sicherer Seeds sollten daher mehrere Systemaufrufe in geeigneter Weise kombiniert werden. Dazu eignen sich die beiden Funktionen

1. ReadTimeStampCounter():
   gibt die Anzahl der seit dem Systemstart durchgelaufenen Prozessorzyklen an. Bei einem Prozessor mit einer Taktfrequenz von mindestens 1 GHz gibt es pro Sekunde also mindestens 2^{30} verschiedene Werte.

2. KeQuerySystemTime():
   gibt die aktuelle Systemzeit an und hat eine Auflösung von 100 ns, und nimmt somit pro Sekunde mindestens 2^{23} verschiedene Werten an.

Kombiniert man diese beiden Funktionen in geeigneter Weise, so dass die Zeitpunkte, an denen die Funktionen aufgerufen werden, voneinander (weitestgehend) unabhängig sind, lässt sich so ein Seed mit hinreichender Entropie erzeugen.

Beispielhaft sei folgendes Verfahren angegeben. Dabei muss aber gewährleistet sein, dass sämtliche Schritte (inklusive das Starten des Rechners und das Anstoßen der Abfragen der Zeitpunkte A,B,C,D) von einem menschlichem Nutzer ausgeführt werden und nicht automatisiert sind.

1. Start des Rechners
2. Start des Programms
   a) A:=ReadTimeStampCounter(),
   b) B:=KeQuerySystemTime(),
3. Verifizieren eines Logins
   a) C:=ReadTimeStampCounter(),
4. Erzeugen des Seeds
   a) D:=ReadTimeStampCounter(),
   b) SEED:=A||B||C||D,

Geht man davon aus, dass ein Angreifer die obigen Zeitpunkte nur auf höchstens eine Sekunde genau schätzen kann, so hat der Wert SEED eine Entropie von mindestens 113 Bit. Dass ein Angreifer die Zeitpunkte nicht genauer in Erfahrung bringen kann, muss durch entsprechende operationelle Maßnahmen sichergestellt werden.

Für die Erzeugung von Seed-Werten kann es zudem empfehlenswert sein, auch Zufallsquellen einzubeziehen, deren Verhalten nicht gründlich untersucht wurde. Im obigen Beispiel könnte der Seed-Wert erweitert werden um einen Wert E, der aus einem im System vorhandenen Hardware-Zufallsgenerator stammt, auch wenn dieser nicht durch eine unabhängige Stelle untersucht wurde.

Der auf solche oder ähnliche Weise gewonnene Seed-Wert ist nicht selbst als kryptographischer Schlüssel geeignet, sondern nur als Seed-Wert für einen geeigneten Pseudozufallsgenerator. Auch hierfür muss je nach Konstruktion des Pseudozufallsgenerators noch eine kryptographische Nachbearbeitung des Seed-Wertes erfolgen, zum Beispiel durch Anwendung einer Hashfunktion.

Anhang A.

Anwendung kryptographischer Verfahren

Die in den vorangegangenen Kapiteln erläuterten Verfahren müssen häufig miteinander kombiniert werden, um den Schutz sensitiver Daten zu gewährleisten. Insbesondere sollten zu übertragende sensitive Daten nicht nur verschlüsselt, sondern zusätzlich authentisiert werden, damit eine etwaige Veränderung vom Empfänger erkannt wird.

Weiter muss eine Schlüsseleinigung immer mit einer Instanzauthentisierung und einer Authentisierung aller während der Schlüsseleinigung übertragenen Nachrichten einher gehen, damit sich beide Parteien sicher sind, mit wem sie kommunizieren. Andernfalls kann durch eine sogenannte Man-in-the-Middle-Attacke die Kommunikation kompromittiert werden. Je nach Anwendung können neben der Man-in-the-Middle-Attacke auch andere Arten von Angriffen auf die Authentizität der Nachrichtenübermittlung die Sicherheit eines informationsverarbeitenden Systems ohne Instanzauthentisierung oder ohne Datenauthentisierung gefährden (z.B. Replay-Attacken).

Sowohl für Verschlüsselung mit Datenauthentisierung als auch zur authentisierten Schlüsselvereinbarung geben wir in diesem Kapitel geeignete Verfahren an.

A.1. Verschlüsselungsverfahren mit Datenauthentisierung (Secure Messaging)

Grundsätzlich können bei der Kombination von Verschlüsselung und Datenauthentisierung alle in Kapitel 2 bzw. Abschnitt 5.1 empfohlenen Verfahren eingesetzt werden.

Allerdings müssen die folgenden beiden Nebenbedingungen eingehalten werden:

1. Authentisiert werden ausschließlich die verschlüsselten Daten sowie gegebenenfalls nicht vertrauliche Daten, die unverschlüsselt übertragen werden und nur authentisiert werden sollen.

2. Verschlüsselungs- und Authentisierungsschlüssel sollen verschieden und sollen nicht voneinander ableitbar sein.

Bemerkung 24 Es besteht die Möglichkeit, Verschlüsselungs- und Authentisierungsschlüssel aus einem gemeinsamen Schlüssel abzuleiten. Empfohlene Verfahren sind in Abschnitt B.1 zusammengefasst.

A.2. Authentisierte Schlüsselvereinbarung

Wie bereits in der Einleitung zu diesem Kapitel erwähnt, muss eine Schlüsselvereinbarung immer mit einer Instanzauthentisierung kombiniert werden. Wir geben nach einigen allgemeinen Vorbemerkungen sowohl Verfahren an, die sich auf rein symmetrische Algorithmen stützen, als auch solche, die sich auf rein asymmetrische Algorithmen stützen.

A.2.1. Vorbemerkungen

Ziele Ziel eines Verfahrens zum Schlüsseltausch mit Instanzauthentisierung ist es, dass die beteiligten Parteien ein gemeinsames Geheimnis teilen und dass sie am Ende der Protokollausführung sicher sind, mit wem sie es teilen.

Für die Berechnung symmetrischer Schlüssel für Verschlüsselungs- und Datenauthentisierungsverfahren aus diesem Geheimnis siehe Abschnitt B.1.

Voraussetzungen an die Umgebung Symmetrische Verfahren zum authentisierten Schlüsseltausch nehmen stets die Existenz vorverteilter Geheimnisse an. Bei asymmetrischen Verfahren wird in der Regel die Existenz einer Public-Key-Infrastruktur angenommen, 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 auf zuverlässigem Wege allen Teilnehmern bekanntgemacht worden sind und dass alle Teilnehmer zu einer korrekten Prüfung aller relevanten Zertifikate auf Gültigkeit zu jedem Zeitpunkt in der Lage sind.

Hinweise zur Umsetzung Bei der konkreten Umsetzung der vorgestellten Verfahren müssen die folgenden beiden Bedingungen erfüllt werden.

1. Die für die Authentisierung benutzten Zufallswerte müssen bei jeder Durchführung des Protokolls mit großer Wahrscheinlichkeit verschieden sein. Dies kann zum Beispiel dadurch erreicht werden, dass jedes Mal ein Zufallswert der Länge mindestens 100 Bit bezüglich der Gleichverteilung aus $\{0,1\}^{100}$ gewählt wird.

2. 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 100 Bit Min-Entropie zu dem auszuhandelnden Schlüssel beitragen.

A.2.2. Symmetrische Verfahren

Grundsätzlich kann jedes Verfahren aus Abschnitt 6.1 zur Instanzauthentisierung mit jedem Verfahren aus Abschnitt 7.1 zum Schlüsselaustausch miteinander kombiniert werden. Die Kombination muss dabei so erfolgen, dass die ausgetauschten Schlüssel tatsächlich authentisiert sind, Man-in-the-Middle-Attacken also ausgeschlossen werden können.

Folgendes Verfahren wird für diese Anwendung empfohlen:

Key Establishment Mechanism 5 aus [46].

Tabelle A.1.: Empfohlenes symmetrisches Verfahren zur authentisierten Schlüsselvereinbarung

Bemerkung 25 Als Verschlüsselungsverfahren können in Key Establishment Mechanism 5 aus [46] alle in dieser Technischen Richtlinie empfohlenen authentisierten Verschlüsselungsverfahren verwendet werden (siehe Abschnitt A.1).

[1] Hier wird davon ausgegangen, dass nur symmetrische Schlüssel ausgehandelt werden.

A.2.3. Asymmetrische Verfahren

Wie schon für symmetrische Verfahren kann auch hier grundsätzlich jedes Verfahren aus Abschnitt 6.2 zur Instanzauthentisierung mit jedem Verfahren aus Abschnitt 7.2 zur Schlüsselvereinbarung miteinander kombiniert werden.

Um allerdings 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 benötigen als Vorbedingung einen Mechanismus zur manipulationssicheren Verteilung öffentlicher Schlüssel. Dieser Mechanismus muss folgende Eigenschaften aufweisen:

- Der durch einen Nutzer erzeugte öffentliche Schlüssel muss zuverlässig an dessen Identität gebunden werden.
- Ebenso sollte der zugehörige private Schlüssel 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 Wege, dies zu erreichen. Die manipulationssichere Schlüsselverteilung kann durch eine PKI erreicht 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 [71], Abschnitt 6.6.2 beschriebene Verfahren empfohlen. In dem zweiten empfohlenen Verfahren (KAS2-bilateral-confirmation nach [71]) ist dieser Schritt bereits enthalten.

1. Elliptic Curve Key Agreement of ElGamal Type (ECKA-EG), siehe [23].
2. Instanzauthentisierung mit RSA und Schlüsselvereinbarung mit RSA, siehe KAS2-bilateral-confirmation nach [71], Abschnitt 8.3.3.4.
3. MTI(A0), siehe [47], Annex C.6.

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

Bemerkung 26 Um konform mit der vorliegenden Technischen Richtlinie zu sein, muss bei der konkreten Umsetzung der Protokolle darauf geachtet werden, dass lediglich die in diesem Dokument empfohlenen kryptographischen Komponenten zur Anwendung kommen.

Bemerkung 27 Bei dem Verfahren ECKA-EG findet keine gegenseitige Authentisierung statt. Hier beweist nur eine Partei der anderen im Besitz eines privaten Schlüssels zu sein, und auch dies geschieht nur im Anschluss an die Ausführung des Protokolls implizit durch 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 zum Beispiel Systemparameter zu generieren, oder aus den Ergebnissen von Zufallszahlengeneratoren oder Schlüsseleinigungsverfahren symmetrische Schlüssel zu erzeugen. Diese Funktionen und Algorithmen sind sorgfältig zu wählen, um das in dieser Technischen Richtlinie geforderte Sicherheitsniveau zu erreichen und kryptoanalytische Angriffe zu verhindern.

B.1. Schlüsselableitung

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, abgeleitet werden. Daneben können durch Verwendung einer Schlüsselableitungsfunktion auch folgende Ziele erreicht werden:

1. Bindung von Schlüsselmaterial an Protokolldaten (zum Beispiel Sendername, Empfängername...) durch Verwendung der Protokolldaten in der Schlüsselableitungsfunktion.

2. Ableitung von Sitzungsschlüsseln oder Schlüsseln für verschiedene Zwecke aus einem Masterschlüssel auch in rein symmetrischen Kryptosystemen.

3. Nachbearbeitung von Zufallsdaten zur Beseitigung statistischer Schiefen bei der Herstellung kryptographischer Schlüssel.

Folgendes Verfahren wird für alle Anwendungen von Schlüsselableitungsfunktionen empfohlen:

Key Derivation through Extraction-then-Expansion nach [72].

Tabelle B.1.: Empfohlenes Verfahren zur Schlüsselableitung

Es wird empfohlen, als MAC-Funktion in dem angegebenen Verfahren einen der in Abschnitt 5.1 empfohlenen MACs einzusetzen. Falls keine der dort angegebenen MAC-Funktionen verwendet werden kann, ist auch die Nutzung eines HMAC-SHA1 möglich.

B.2. Erzeugung unvorhersagbarer Initialisierungsvektoren

Wie bereits in Abschnitt 2.1.2 beschrieben, müssen Initialisierungsvektoren für symmetrische Verschlüsselungsverfahren, die die Betriebsart Cipherblock Chaining Mode (CBC) einsetzen, unvorhersagbar sein. Dies bedeutet nicht, dass die Initialisierungsvektoren vertraulich behandelt werden müssen, sondern lediglich, dass ein möglicher Angreifer nicht in der Lage sein darf,

zukünftig eingesetzte Initialisierungsvektoren mit einer Erfolgswahrscheinlichkeit $> 2^{-95}$ zu erraten. Darüber hinaus darf der Angreifer auch nicht in der Lage sein, die Wahl der Initialisierungsvektoren zu beeinflussen.

Zwei Verfahren werden zur Erzeugung unvorhersagbarer Initialisierungsvektoren empfohlen (sei dazu n die Blockgröße der eingesetzten Blockchiffre):

1. **Zufällige Initialisierungsvektoren:** Erzeuge eine zufällige Bitfolge der Länge n und nutze diese als Initialisierungsvektor. Die Entropie der zufälligen Bitfolge muss dabei mindestens 95 Bit betragen.
2. **Verschlüsselte Initialisierungsvektoren:** Nutze ein deterministisches Verfahren zur Erzeugung sogenannter Prä-Initialisierungsvektoren (z.B. einen Zähler). Verschlüssele den Prä-Initialisierungsvektor mit der einzusetzenden Blockchiffre und dem einzusetzenden Schlüssel und nutze den Chiffretext 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 vermuteten Schwierigkeit des Diskreten Logarithmenproblems in diesen Gruppen.

Zur Festlegung der EC-Systemparameter werden benötigt:

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

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

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

Die Werte (p, a, b, P, q, i) bilden dann die *EC-Systemparameter*, wobei $q := \mathrm{ord}(P)$ die Ordnung des Basispunktes P in $E(\mathbb{F}_p)$ bezeichne und $i := \#E(\mathbb{F}_p)/q$ der sogenannten *Kofaktor* ist.

Nicht alle EC-Systemparameter sind für die in diesem Dokument empfohlenen asymmetrischen Verfahren basierend auf elliptischen Kurven geeignet, d.h. dass in den von diesen elliptischen Kurven generierten Gruppen das diskrete Logarithmenproblem effizient lösbar ist. Neben einer ausreichenden Bitlänge von q müssen zusätzlich die folgenden Bedingungen gelten:

1. Die Ordnung $q = \mathrm{ord}(P)$ des Basispunktes P ist eine von p verschiedene Primzahl,
2. $p^r \neq 1 \bmod q$ für alle $1 \leq r \leq 10^4$, und
3. die Klassenzahl der Hauptordnung, die zum Endomorphismenring von E gehört, ist mindestens 200.

Siehe [34] für eine Erläuterung.

EC-Systemparameter, die die obigen Bedingungen erfüllen, heißen auch *kryptographisch stark*.

Bemerkung 28 Es wird empfohlen, die unter Punkt 1. zu generierende 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:

1. brainpoolP224r1, siehe [34],
2. brainpoolP256r1, siehe [34],
3. brainpoolP320r1, siehe [34],
4. brainpoolP384r1, siehe [34],
5. brainpoolP512r1, siehe [34].

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 angesprochen, die Zufallszahlen $k \in \{1, \ldots, q-1\}$ (z.B. als Ephemeralschlüssel) benötigen, wobei q in aller Regel keine 2er-Potenz ist. Es wurde bereits in den Bemerkungen 7, 6, 12 und 13 darauf hingewiesen, dass k nach Möglichkeit (zumindest nahezu) gleichverteilt gewählt werden sollte.

Die in Kapitel 9 behandelten Zufallszahlengeneratoren erzeugen jedoch gleichverteilte Zufallszahlen auf $\{0, 1, \ldots, 2^n - 1\}$ ('zufällige n-Bitstrings'). Die Aufgabe besteht also darin, hieraus (wenigstens nahezu) gleichverteilte Zufallszahlen auf $\{0, 1, \ldots, q\}$ abzuleiten.

In Tabelle B.4 werden zwei Verfahren beschrieben, mit denen dies bewerkstelligt werden kann. Dabei ist $n \in \mathbb{N}$ so gewählt, dass $2^{n-1} \leq q < 2^n - 1$ (Mit anderen Worten: q hat die Bitlänge n).

Verfahren 1.

1. Wähle $k \in \{0, 1, \ldots, 2^n - 1\}$ gleichverteilt.
2. `if` $k < q$ `return` k
3. `else goto` 1.

Verfahren 2.

1. Wähle $k \in \{0, 1, \ldots, 2^{n+64} - 1\}$ gleichverteilt.
2. $k = k' \bmod q$ (d.h. $0 \leq k < q$).
3. `return` k.

Tabelle B.4.: Berechnung von Zufallswerten auf $\{0, \ldots, q-1\}$

Bemerkung 29 (*i*) Das Verfahren 1 überführt eine Gleichverteilung auf $\{0,\ldots,2^n-1\}$ in eine Gleichverteilung auf $\{0,\ldots,q-1\}$, und zwar liefert Verfahren 1 die bedingte Verteilung auf $\{0,\ldots,q-1\} \subset \{0,\ldots,2^n-1\}$. Dagegen erzeugt Verfahren 2 selbst für ideale Zufallszahlengeneratoren mit Werten in $\{0,\ldots,2^n-1\}$ 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 jedoch den Vorteil, dass etwaig vorhandene Schiefen auf $\{0,\ldots,2^n-1\}$ in aller Regel reduziert werden dürften. Für PTG.2-konforme Zufallszahlengeneratoren wird daher dieses Verfahren empfohlen.

(*iii*) Das Verfahren 1 besitzt außerdem 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. (Es sei angemerkt, dass die Wahrscheinlichkeit, dass eine auf $k \in \{0,1,\ldots,2^n-1\}$ gleichverteilte Zufallszahl kleiner als q ist, $q/2^n \geq 2^{n-1}/2^n = 1/2$ ist.)

B.5. Probabilistische Primzahltests

Bei der Festlegung der Systemparameter für RSA-basierte asymmetrische Verfahren müssen zwei Primzahlen p, q gewählt werden. Für die Sicherheit dieser Verfahren ist es zusätzlich nötig, die Primzahlen geheim zu halten, dies setzt insbesondere voraus, dass p und q zufällig gewählt werden müssen.

Folgendes Verfahren wird zur Erzeugung zufälliger Primzahlen empfohlen. Dabei sei b die gewünschte Bitlänge der Primzahl (hier $b \geq 1024$).

1. Wähle eine zufällige ungerade Zahl p der Bitlänge b.
2. Falls p keine Primzahl ist, gehe zurück zu 1.
3. Gib p aus.

Tabelle B.5.: Empfohlenes Verfahren zur Erzeugung von Primzahlen

Für den zweiten Schritt des obigen Algorithmus werden aus Effizienzgründen meist probabilistische Primzahltests eingesetzt. Der folgende Algorithmus wird empfohlen:

Miller-Rabin, siehe [66], Algorithmus 4.24.

Tabelle B.6.: Empfohlener probabilistischer Primzahltest

Bemerkung 30 (*i*) Der Miller-Rabin-Algorithmus benötigt neben der zu untersuchenden Zahl p einen Zufallswert $a \in \{2,3,\ldots,p-2\}$, die sogenannte Basis. Ist a bezüglich der Gleichverteilung auf $\{2,3,\ldots,p-2\}$ gewählt, so ist die Wahrscheinlichkeit dafür, dass p zusammengesetzt ist, obwohl der Miller-Rabin-Algorithmus ausgibt, dass p eine Primzahl ist, höchstens $1/4$.

(*ii*) (Worst Case) Um die Wahrscheinlichkeit dafür, dass eine feste Zahl p mittels des Miller-Rabin-Algorithmus als Primzahl erkannt wird, obwohl sie zusammengesetzt ist, auf $1/2^{100}$ zu beschränken, muss der Algorithmus 50-mal mit jeweils unabhängig voneinander bezüglich der Gleichverteilung gewählten Basen $a_1,\ldots,a_{50} \in \{2,3,\ldots,p-2\}$ aufgerufen werden, siehe

Abschnitt B.4 für empfohlene Verfahren zur Berechnung gleichverteilter Zufallszahlen aus $\{2, 3, \ldots, p-2\}$.

(*iii*) (Average Case) Um eine 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 wiedergegebene Abschätzung nahelegen würde, vergleiche [31], [38], Appendix F und [50], Annex A. So sind für $b = 1024$ lediglich 4 Iterationen notwendig, um bei einer verbleibenden Fehlerwahrscheinlichkeit von 2^{-109} auszuschließen, dass p zusammengesetzt ist, obwohl der Miller-Rabin-Algorithmus p als Primzahl erkennt [50]. Auch hier müssen die Basen unabhängig und 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 Zahlen, für die die Abschätzungen des Worst-Case-Falles zutreffen, in ihrer Dichte mit steigender Größe der Zahlen deutlich abnehmen.

(*iv*) (Optimierungen) Zur Optimierung der Laufzeit etwa von Algorithmus B.5 kann es nützlich sein, vor Anwendung des probabilistischen Primzahltests zusammengesetzte Zahlen mit sehr kleinen Faktoren durch Probedivision zu eliminieren. Ein solcher Vortest hat nur geringfügige Auswirkungen auf die Wahrscheinlichkeit, dass vom Test als wahrscheinliche 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 unverändert.

(*v*) (Sonstige Anmerkungen) Entsprechend [38], Appendix F.2 wird auch in dieser Richtlinie empfohlen, bei der Erzeugung von Primzahlen, die in besonders sicherheitskritischen Funktionen in einem Kryptosystem Verwendung finden sollen oder deren Erzeugung wenig zeitkritisch ist, eine Verifikation der Primzahleigenschaft mit 50 Runden des Miller-Rabin-Tests durchzuführen. 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 genutzt werden.

Anhang C.

Protokolle für spezielle kryptographische Anwendungen

Wir behandeln hier einige Protokolle, die als Bausteine kryptographischer Lösungen benutzt werden können. 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 den Entwickler nur schwer zu entdecken sind. Es wird daher empfohlen, dort, wo es möglich ist, allgemein zugängliche und vielfach evaluierte Protokolle einer eigenen Protokollentwicklung vorzuziehen.

C.1. SRTP

SRTP ist ein Protokoll, das das Audio- und Videoprotokoll RTP um Funktionen zur Sicherstellung von Vertraulichkeit und Integrität der übertragenen Nachrichten ergänzt. Es wird in RFC 3711 [7] definiert. SRTP muss mit einem Protokoll zum Schlüsselmanagement kombiniert werden, da es keine eigenen Mechanismen zur Aushandlung eines Kryptokontextes vorsieht.

Wir empfehlen folgende Verwendung von SRTP:

- Als symmetrische Verschlüsselungsverfahren werden sowohl AES im Counter-Modus als auch im f8-Modus wie in [7] empfohlen.
- Zum Integritätsschutz sollte ein auf SHA-2 (bevorzugt) oder SHA-1 basierender HMAC verwendet werden. Dieser HMAC darf im Kontext dieses Protokolls auf 80 Bit gekürzt werden.
- Als Schlüsselmanagementsystem sollte MIKEY [5] verwendet werden. Dabei werden die folgenden Schlüsselmanagementverfahren aus [5] empfohlen: DH-Schlüsseltausch mit Authentisierung über PKI, RSA mit PKI, und 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 [7] 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 zum Beispiel durch die Datenübertragungsrate, die zeitliche Abfolge verschiedener Signale oder eine sonstige Verkehrsanalyse zu minimieren. Ansonsten werden Angriffe wie in [6] möglich.

C.2. SSH

SSH ist ein Protokoll zur Herstellung einer sicheren Verbindung zu einem entfernten Terminal, das zum Beispiel bei der Wartung von Rechnersystemen über ein unsicheres Netzwerk hinweg zum Einsatz kommt. Die kryptographische Sicherung der Verbindung soll in dieser Situation sowohl eine Manipulation der übermittelten Daten verhindern (zum Beispiel also von Befehlen an eine Unix-Shell) als auch ihre Vertraulichkeit schützen. Die Grundarchitektur des Protokolls wird in [90] beschrieben, auf weitere relevante Dokumente wird dort verwiesen. Hinsichtlich einer Nutzung von SSH werden folgende Empfehlungen ausgesprochen:

- Es sollte keine niedrigere SSH-Version als SSH 2.0 verwendet werden.
- Es sollte eine quelloffene, weit verbreitete Implementierung von SSH 2.0 verwendet werden.
- Wenn die Implementierung das zulässt, sollte als MAC ein SHA2-HMAC verwendet werden.
- Zur Datenauthentisierung sollte, wenn ein SHA2-HMAC nicht zur Verfügung steht, ein SHA1-HMAC eingesetzt werden.
- Als symmetrische Verschlüsselungsfunktion sollte AES im Counter-Modus verwendet werden.
- Eine Neuaushandlung von Schlüsseln muss in den in [9] angegebenen Abständen erfolgen.
- Die Schlüssellängen für asymmetrische SSH-Komponenten sollten den Empfehlungen der vorliegenden Technischen Richtlinie entsprechen.
- Bei Verwendung von EC-Kryptographie wird empfohlen, die in Tabelle B.3 gelisteten Parameter zu nutzen. Ist dies nicht möglich, sollte auf die in [86], Abschnitt 10.1 gelisteten NIST-Kurven mit Bitlänge mindestens 224 Bit zurückgegriffen werden.
- SSH ist anfällig für bestimmte Typen von Keystroke-Timing-Attacken [85]. Die Relevanz dieser Angriffe in einem gegebenen Anwendungsszenario sollte durch einen Experten eingeschätzt werden, bevor SSH in kritischen Systemen eingesetzt wird. Abhängig von dieser Einschätzung der Bedrohungslage sollte die Implementierung geeignete Gegenmaßnahmen enthalten.
- Zur Authentisierung des SSH-Benutzers/Clients sollte eine zwei-Faktor-Authentisierung über ein starkes Passwort und ein auf dem Client-Rechner oder (besser) auf einem geschützten Authentisierungstoken gespeichertes Zertifikat genutzt werden.

Literaturverzeichnis

[1] M. Abdalla, M. Bellare, P. Rogaway, *DHIES: An encryption scheme based on the Diffie-Hellman Problem*, verfügbar unter http://charlotte.ucsd.edu/~mihir/papers/dhaes.pdf

[2] AIS 20: *Funktionalitätsklassen und Evaluationsmethodologie für deterministische Zufallszahlengeneratoren*, Version 2.1, 18.09.2011, Bundesamt für Sicherheit in der Informationstechnik. Verfügbar unter https://www.bsi.bund.de/cae/servlet/contentblob/478150/publicationFile/30276/ais20pdf.pdf.

[3] AIS 31: *Funktionalitätsklassen und Evaluationsmethodologie für physikalische Zufallszahlengeneratoren*, Version 2.1, 18.09.2011, Bundesamt für Sicherheit in der Informationstechnik. Verfügbar unter https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Zertifizierung/Interpretationen/ais31_pdf.pdf.

[4] W. Killmann, T. Lange, M. Lochter, W. Thumser, G. Wicke, Anhang zu AIS 46: *Minimum Requirements for Evaluating Side-Channel Attack Resistance of Elliptic Curve Implementations* Bundesamt für Sicherheit in der Informationstechnik Verfügbar unter https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Zertifierung/Interpretation/ECCGuide.pdf

[5] J. Arkko, E. Carrara, F. Lindholm, M. Naslund, K. Norrman, *MIKEY: Multimedia Internet KEYing*, RFC 3830, verfügbar http://tools.ietf.org/html/rfc3830

[6] Ballard, Coulls, Monrose, Masson, *Spot me if you can: recovering spoken phrases in encrypted VoIP conversations*, IEEE Symposium on Security and Privacy, 2008

[7] M. Baugher, D. McGrew, M. Naslund, E. Carrara, K. Norrman *The Secure Real-time Transport Protocol (SRTP)* RFC 3711, 2004

[8] M. Bellare, R. Canetti und H. Krawczyk. *HMAC: Keyed-Hashing for Message Authentication.* RFC 2104, 1997.

[9] M. Bellare, T. Kohno, C. Namprempre *The Secure Shell (SSH) Transport Layer Encryption Modes* RFC 4344, 2006

[10] S. Kent, K. Seo, *Security Architecture for the Internet Protocol* RFC 4301, 2005

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

[12] A. Biryukov und D. Khovratovich, *Related-Key Cryptanalysis of the Full AES-192 and AES-256*, Asiacrypt 2009, LNCS 5912/2009, 1-18

[13] A. Biryukov, O. Dunkelman, N. Keller, D. Khovratovich, A. Shamir, *Key Recovery Attacks of Practical Complexity on AES Variants With Up to 10 Rounds*, Eurocrypt 2010, LNCS 6110/2010, 299-319

[14] S. Blake-Wilson, A. Menezes, *Unknown Key-Share Attacks on the Station-to-Station (STS) Protocol*, Public Key Cryptography 1999, LNCS 1560/1999, 154-170

[15] BNetzA, Bekanntmachung zur elektronischen Signatur nach dem Signaturgesetz und der Signaturverordnung (Übersicht über geeignete Algorithmen), veröffentlicht auf den Internetseiten des Bundesanzeigers unter BAnz AT 27.03.2013 B4

[16] A. Bogdanov, D. Khovratovich, C. Rechberger, *Biclique Cryptanalysis of the Full AES*, Asiacrypt 2011, LNCS 7073/2011, 344-371

[17] D. Boneh und G. Durfee. *Cryptanalysis of RSA with private key d less then* $N^{0.292}$. Eurocrypt 1999, LNCS 1592/1999, 1-11.

[18] D. Brown, *Generic Groups, Collision Resistance, and ECDSA*, Designs, Codes and Cryptography 04/2005, Vol. 35, Issue 1, S. 119-152

[19] T. Brown, R. Gallant, *The Static Diffie-Hellman Problem*, IACR e-print report 306/2004

[20] BSI, TR 02102-2, *Kryptographische Verfahren: Empfehlungen und Schlüssellängen, Teil 2 - Verwendung von Transport Layer Security (TLS), Version 2014-01*

[21] BSI, TR 02102-3, *Kryptographische Verfahren: Empfehlungen und Schlüssellängen, Teil 3 - Verwendung von IPsec, Version 2014-01*

[22] BSI, TR 3110-2, *Advanced Security Mechanisms for Machine Readable Travel Documents*, Version 2.10, 2012, https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/TechGuidelines/TR03110/TR-03110_v2.1_P2pdf.pdf

[23] BSI, *Elliptic Curve Cryptography*, BSI Technical Guideline TR-03111, Version 2.0, 2012.

[24] BSI, Technische Richtlinie TR-03116, *Technische Richtlinie für eCard-Projekte der Bundesregierung*, Version 3.17, 2013

[25] BSI, Technische Richtlinie TR-03116-2, *eCard-Projekte der Bundesregierung, Teil 2 – Hoheitliche Ausweisdokumente*, Stand 2013

[26] BSI, BSI Technische Richtlinie TR-03125, *Beweiswerterhaltung kryptographisch signierter Dokumente*, 2011, https://www.bsi.bund.de/ContentBSI/Publikationen/TechnischeRichtlinien/tr03125/index_htm.html

[27] J. Buchmann, E. Dahmen, A. Hülsing *XMSS–A Practical Forward Secure Signature Scheme Based on Minimal Security Assumptions* Fourth International Conference in Post-Quantum Cryptography, LNCS 7071/2011, 117-129

[28] S. Chen, R. Wang, X. Wang, K. Zhang, *Side-Channel Leaks in Web Applications: A Reality Today, a Challenge Tomorrow*, 2010 IEEE Symposium on Security and Privacy, verfügbar unter http://research.microsoft.com/pubs/119060/WebAppSideChannel-final.pdf

[29] J. H. Cheon, *Security Analysis of the Strong Diffie-Hellman Problem*, Advances in Cryptology - Eurocrypt 2006, LNCS 4004/2006, S. 1-11

[30] J.-S. Coron, D. Naccache und J. Stern. *On the Security of RSA-Padding.* Crypto 99, LNCS 1666/1999, 1-18.

[31] I. Damgard, P. Landrock und C. Pommerance. *Average Case Error Estimates for the Strong Provable Prime Test* Mathematics of Computation, Vol. 61, No. 203, 177-194, 1993.

[32] M. Daum und S. Lucks. *The Story of Alice and Bob,* "Rump Session"-Vortrag Eurocrypt 2005, http://www.cits.rub.de/imperia/md/content/magnus/rumpec05.pdf.

[33] W. Diffie, P. van Oorschot, M. Wiener, *Authentication and Authenticated Key Exchanges*, Designs, Codes and Cryptography, 1992 Vol. 2, Number 2, S. 107-125

[34] M. Lochter, J. Merkle, RFC 5639: *Elliptic Curve Cryptography ECC Brainpool Standard Curves and Curve Generation*, 2010, . http://tools.ietf.org/html/rfc5639,

[35] ECRYPT II, *Yearly Report on Algorithms and Key Sizes*, 2011

[36] Federal Information Processing Standards Publication 140-2 (FIPS PUB 140-2) *Security Requirements for Cryptographic Modules*, 2002

[37] Federal Information Processing Standards Publication 180-4 (FIPS PUB 180-4) *Secure Hash Standard*, 2012

[38] Federal Information Processing Standards Publication 186-4 (FIPS PUB 186-4) *Digital Signature Standard (DSS)*, 2013.

[39] Federal Information Processing Standards Publication 197 (FIPS PUB 197). *Advanced Encryption Standard (AES)*, 2001.

[40] N. Ferguson, *Authentication Weaknesses in GCM*, 2005, http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/comments/CWC-GCM/Ferguson2.pdf

[41] N. Ferguson, B. Schneier *A Cryptographic Evaluation of IPsec*, 2003, verfügbar unter http://www.schneier.com/paper-ipsec.html

[42] M. Gebhardt, G. Illies, W. Schindler. *A Note on the Practical Value of Single Hash Collisions for Special File Formats,* Sicherheit 2006, Köllen-Verlag, LNI P-77 (2006), 333-344. Erweiterte Version: NIST Cryptographic Hash Workshop 2005, http://csrc.nist.gov/groups/ST/hash/documents/Illies_NIST_05.pdf.

[43] N. Heninger, Z. Durumeric, E. Wustrow, J. Halderman, *Mining Your Ps and Qs: Detection of Widespread Weak Keys in Network Devices*, Proceedings of the 21st Usenix Symposium, 08/2012, https://www.factorable.net/weakkeys12.conference.pdf

[44] G. Illies, M. Lochter, O. Stein *Behördliche Vorgaben zu kryptographischen Algorithmen*, Datenschutz und Datensicherheit 11/2011, S. 807-811

[45] IEEE P1363. *Standard Specifications for Public Key Cryptography*, 2000.

[46] ISO/IEC 11770-2-2008 *Information technology – Security techniques – Key management - Part 2: Mechanisms using symmetric techniques*, 2008

[47] ISO/IEC 11770-3-2008. *Information technology – Security techniques – Key management – Part 3: Mechanisms using asymmetric techniques*, 2008.

[48] ISO/IEC 14888-2-2008 *Information technology – Security techniques – Digital signatures with appendix – Part 2: Integer factorization based mechanisms*, 2008

[49] ISO/IEC 14888-3-2006. *Information technology – Security techniques – Digital signatures with appendix – Part 3: Discrete logarithm based mechanisms*, 2006.

[50] ISO/IEC 18032. *IT security techniques – Prime number generation,* 2005.

[51] ISO/IEC 9796-2-2010. *Information technology – Security techniques – Digital Signature Schemes giving message recovery – Part 2: Integer Factorization based mechanisms*, 2010.

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

[53] G. Itkis, *Forward Security, adaptive cryptography: Time Evolution*, 2004, http://www.cs.bu.edu/~itkis/pap/forward-secure-survey.pdf

[54] T. Iwata, K. Ohashi, K. Minematsu, *Breaking and Repairing GCM Security Proofs*, Crypto 2012, LNCS 7417/2012, S. 31-49

[55] J. Kelsey, B. Schneier, D. Wagner, *Key-Schedule Cryptanalysis of IDEA, G-DES, GOST, SAFER, and Triple-DES*, Crypto 96, LNCS 1109/1996, S. 237-251

[56] W. Killmann, W. Schindler: *A proposal for: Functionality classes and evaluation methodology for true (physical) random number generators*, Version 3.1, 25.09.2001. Bundesamt für Sicherheit in der Informationstechnik, frühere mathematisch-technische Anlage zu [3].[1] Verfügbar unter https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Zertifierung/Interpretation/AIS31_Functionality_classes_evaluation_methodology_for_true_RNG.pdf

[57] W. Killmann, W. Schindler: *Functionality classes for random number generators.* Bundesamt für Sicherheit in der Informationstechnik, Version 2.0, 18.09.2011, mathematisch-technische Anlage zu [2] und [3]. Verfügbar unter https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Zertifierung/Interpretation/AIS31_Functionality_classes_for_random_number_generators.pdf

[58] A.K. Lenstra und E.R. Verheul. *Selecting Cryptographic Key Sizes.* J. Cryptology 39, 2001. 255-293

[59] A.K. Lenstra. *Key lengths*, in: *Handbook of Information Security*, John Wiley & Sons, 2006, 617-635

[60] S. Lucks, *Attacking Triple Encryption*, Fast Software Encryption 1998, LNCS Vol. 1372, S. 239-253

[61] ISO/IEC 18033-2. *Information Technology – Security techniques – Part 2: Asymmetric Ciphers*, 2006

[62] V. Gayoso Martínez, F. Hernández Álvarez, L. Hernández Encinas, C. Sánchez Ávila, *A Comparison of the Standardized Versions of ECIES*, Sixth International Conference on Information Assurance and Security, 2010

[63] R. Merkle, *Secure Communications over Insecure Channels*, Communications of the ACM, Vol. 21, No. 4, 1978, S. 294-299

[64] R. Merkle, M. Hellman, *On the security of multiple encryption*, Communications of the ACM, Vol. 24, No. 7, 1981, S. 465-467

[65] P. van Oorschot, M. Wiener, *A known-plaintext attack on two-key triple encryption*, Eurocrypt 1990, LNCS Vol. 473, S. 318-323

[66] A. Menezes, P. van Oorschot und O. Vanstone. *Handbook of Applied Cryptography.* CRC Press, 1996.

[67] NIST. Recommendation for Block Cipher Modes of Operation: Methods and Techniques, Special Publication SP800-38A, National Institute of Standards and Technology, Technology Administration, U.S. Department of Commerce, 2001.

[1] noch relevant für Aspekte, die durch andere Anlagen von [3] nicht geregelt werden.

[68] NIST Recommendation for Block Cipher Modes of Operation: The XTS-AES Mode for Confidentiality on Storage Devices, Special Publication SP800-38E, National Institute of Standards and Technology, Technology Administration, U.S. Department of Commerce, 2010

[69] NIST. Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication, Special Publication SP800-38B, National Institute of Standards and Technology, Technology Administration, U.S. Department of Commerce, 2005.

[70] NIST. Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) for Confidentiality and Authentication, Special Publication SP800-38D, National Institute of Standards and Technology, Technology Administration, U.S. Department of Commerce, November 2007.

[71] NIST Recommendation for Pair-Wise Key Establishment Schemes Using Integer Factorization Cryptography, Special Publication SP800-56B, National Institute of Standards and Technology, Technology Administration, U.S. Department of Commerce, 08.2009

[72] NIST Recommendation for Key Derivation through Extraction-then-Expansion, Special Publication SP800-56C, National Institute of Standards and Technology, Technology Administration, U.S. Department of Commerce, 11.2011

[73] NIST Electronic Authentication Guideline (Version 1.0.2), Special Publication SP800-63, National Institute of Standards and Technology, Technology Administration, U.S. Department of Commerce, 04.2006

[74] NIST Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher, Special Publication SP800-67, National Institute of Standards and Technology, Technology Administration, U.S. Department of Commerce, 05.2008

[75] NIST Recommendation for Key Derivation Using Pseudorandom Functions, Special Publication SP800-108, National Institute of Standards and Technology, Technology Administration, U.S. Department of Commerce, 10.2009

[76] P. Nguyen and I. Shparlinski. The insecurity of the elliptic curve signature algorithm with partially known nonces. *Designs, Codes and Cryptography*, Vol. 30, Number 2, 201-217, 2003.

[77] J.F. Raymond, A. Stiglic, *Security Issues in the Diffie-Hellman Key Agreement Protocol*, IEEE Transactions on Information Theory, 2000

[78] S. Kent IP Authentication Header, *RFC 4302*. 2005

[79] S. Kent. IP Encapsulating Security Payload (ESP). *RFC 4303*, 2005.

[80] R. Housley. Cryptographic Message Syntax (CMS). *RFC 5652,* 2009.

[81] T. Ristenpart, S. Yilek, *When Good Randomness Goes Bad: Virtual Machine Reset Vulnerabilities and Hedging Deployed Cryptography* Proceedings of the Network and Distributed Systems Security Symposium - NDSS 2010, 2010

[82] PKCS #1 v2.2: RSA Cryptographic Standard, 27.10.2012, https://www.emc.com/emc-plus/rsa-labs/pkcs/files/h11300-wp-pkcs-1v2-2-rsa-cryptography-standard.pdf

[83] W. Schindler: *Funktionalitätsklassen und Evaluationsmethodologie für deterministische Zufallszahlengeneratoren.* Bundesamt für Sicherheit in der Informationstechnik, Version 2.0, 02.12.1999, frühere mathematisch-technische Anlage zu [2].[2]. Verfügbar unter https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Zertifierung/Interpretation/AIS20_Functionality_Classes_Evaluation_Methodology_DRNG.pdf

[84] A. Shamir. *How to share a secret.* Communications of the ACM, Vol. 22 Issue 11 (1979), 612-613.

[85] D. X. Song, D. Wagner, X. Tian, *Timing Analysis of Keystrokes and Timing Attacks on SSH*, Proceedings of the 10th USENIX Security Symposium, 2001

[86] D. Stebila, J. Green, *Elliptic Curve Integration in the Secure Shell Transport Layer*, RFC 5656, 2009

[87] M. Stevens, *Attacks on Hash Functions and Applications*, Dissertation, Universität Leiden, 2012

[88] S. Vaudenay, *Security Flaws Induced by CBC Padding: Applications to SSL, IPSEC, WTLS...*, Eurocrypt 2002, LNCS 2332/2002, 534-545

[89] X. Wang, Y.L. Yin, and H. Yu. Finding Collisions in the Full SHA-1. Crypto 2005, LNCS 3621/2005, 17-36.

[90] T. Ylonen, C. Lonvick, *The Secure Shell (SSH) Protocol Architecture*, RFC 4251, 2006

[91] atsec, *Dokumentation und Analyse des Linux-Pseudozufallszahlengenerators*, Studie im Auftrag des BSI, 2013, https://www.bsi.bund.de/DE/Publikationen/Studien/LinuxRNG/index_htm.html

[2] noch relevant für Aspekte, die durch andere Anlagen von [2] nicht geregelt werden.