BSI – Technische Richtlinie

Bezeichnung: Kryptographische Verfahren: Empfehlungen und Schlüssellängen

Kürzel: BSI TR-02102-1

Version: 2018-01

Stand: 22. Januar 2018
<table>
<thead>
<tr>
<th>Version</th>
<th>Datum</th>
<th>Änderungen</th>
</tr>
</thead>
</table>
# Inhaltsverzeichnis

1 Einleitung
   1.1 Sicherheitsziele und Auswahlkriterien ........................................... 7
   1.2 Allgemeine Hinweise ................................................................. 9
   1.3 Kryptographische Hinweise ....................................................... 11
   1.4 Umgang mit Legacy-Algorithmen ................................................ 12
   1.5 Wichtige in dieser Richtlinie nicht behandelte Themen ...................... 13

2 Symmetrische Verschlüsselungsverfahren ........................................... 16
   2.1 Blockchiffren ................................................................. 16
      2.1.1 Betriebsarten ................................................................. 17
      2.1.2 Betriebsbedingungen ....................................................... 18
      2.1.3 Paddingverfahren .......................................................... 19
   2.2 Stromchiffren ................................................................. 19

3 Asymmetrische Verschlüsselungsverfahren .......................................... 23
   3.1 Vorbemerkung zu asymmetrischen Schlüssellängen ............................ 23
      3.1.1 Allgemeine Vorbemerkungen ............................................. 23
         3.1.1.1 Sicherheit asymmetrischer Verfahren .............................. 23
         3.1.1.2 Äquivalente Schlüssellängen für asymmetrische und symmetrische kryptographische Verfahren 24
      3.1.2 Schlüssellängen bei langfristig schützenswerten Informationen und in Systemen mit langer vorgesehener Einsatzdauer 25
   3.2 Sonstige Bemerkungen ........................................................... 27
      3.2.1 Seitenkanalangriffe und Fault-Attacken .................................. 27
      3.2.2 Public-Key-Infrastrukturen .............................................. 27
   3.3 ECIES-Verschlüsselungsverfahren .............................................. 29
   3.4 DLIES-Verschlüsselungsverfahren .............................................. 30
   3.5 RSA ................................................................. 32

4 Hashfunktionen ................................................................. 33

5 Datenauthentisierung ................................................................. 35
   5.1 Vorbemerkungen ................................................................. 35
   5.2 Sicherheitsziele ................................................................. 35
   5.3 Message Authentication Code (MAC) .......................................... 36
   5.4 Signaturverfahren ................................................................. 37
      5.4.1 RSA ................................................................. 37
      5.4.2 Digital Signature Algorithm (DSA) ..................................... 38
      5.4.3 DSA-Varianten basierend auf elliptischen Kurven ................. 39
      5.4.4 Merklesignaturen ........................................................... 40
      5.4.5 Langfristige Beweiswerterhaltung für digitale Signaturen ........ 41
# Tabellenverzeichnis

1.1 Beispiele für Schlüssellängen für ein Sicherheitsniveau von mindestens 100 beziehungsweise 120 Bit .......................... 9
1.2 Empfohlene Schlüssellängen ................................................................. 9
2.1 Empfohlene Blockchiffren ................................................................. 16
2.2 Empfohlene Betriebsarten für Blockchiffren ........................................... 17
2.3 Empfohlene Paddingverfahren für Blockchiffren ........................................ 19
3.1 Schlüssellängen .................................................................................. 22
3.2 Ungefährender Rechenaufwand $R$ (in Vielfachen des Rechenaufwandes für eine einfache kryptographische Operation, z.B. einmalige Auswertung einer Blockchiffre auf einem Block) für die Berechnung diskreter Logarithmen in elliptischen Kurven (ECDLP) beziehungsweise Faktorisierung allgemeiner zusammengesetzter Zahlen mit den angegebenen Bitlängen .......................................................... 26
3.3 Empfohlenes Formatierungsverfahren für den RSA-Verschlüsselungsalgorithmus 32
4.1 Empfohlene Hashfunktionen ................................................................. 34
5.1 Empfohlene MAC-Verfahren ................................................................. 37
5.2 Parameter für empfohlene MAC-Verfahren ............................................. 37
5.3 Empfohlene Signaturverfahren .............................................................. 38
5.4 Empfohlene Formatierungsverfahren für den RSA-Signaturalgorithmus ................ 39
5.5 Empfohlene Signaturverfahren basierend auf elliptischen Kurven .................. 41
6.1 Schematische Darstellung eines Challenge-Response-Verfahren zur Instanzauthentisierung ................................................................. 43
6.2 Empfohlene Passwortlängen und empfohlene Anzahl der Zugriffsversuche für den Zugriffsschutz kryptographischer Komponenten ................................................................. 44
6.3 Empfohlenes passwortbasiertes Verfahren für den Zugriffsschutz auf kontaktlose Chipkarten ................................................................. 46
7.1 Empfohlene asymmetrische Schlüsselleinigungsverfahren ........................ 49
8.1 Berechnung der Teilgeheimnisse im Shamir Secret-Sharing-Verfahren ............. 51
8.2 Zusammensetzen der Teilgeheimnisse im Shamir Secret-Sharing-Verfahren ........ 52
9.1 Empfohlenes Verfahren zur Seedgenerierung unter GNU/Linux ................. 59
A.1 Empfohlenes symmetrisches Verfahren zur authentisierten Schlüsselvereinbarung 64
A.2 Empfohlene asymmetrische Verfahren zur Schlüsselvereinbarung mit Instanzauthentisierung ................................................................. 64
B.1 Empfohlenes Verfahren zur Schlüsselableitung ....................................... 66
B.2 Empfohlene Verfahren zur Erzeugung unvorherseagbarer Initialisierungsvektoren 67
| B.3 | Empfohlene EC-Systemparameter für asymmetrische Verfahren, die auf elliptischen Kurven basieren | 68  |
| B.4 | Berechnung von Zufallswerten auf \( \{0, \ldots, q-1\} \) | 68  |
| B.5 | Empfohlenes Verfahren 1: Erzeugung von Primzahlen durch die Verwerfungsmethode | 70  |
| B.6 | Empfohlenes Verfahren 2: Erzeugung von Primzahlen durch eine effizienzoptimierte Verwerfungsmethode | 70  |
| B.7 | Legacy-Verfahren: Erzeugung von Primzahlen durch inkrementelle Suche | 71  |
| B.8 | Empfohlener probabilistischer Primzahltest | 71  |
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 2018 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, 17, 27, 28, 29]. 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 [49].

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.


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.


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

### 1.1. Sicherheitsziele und Auswahlkriterien


Insgesamt erreichen alle in dieser Technischen Richtlinie angegebenen kryptographischen Verfahren mit den in den einzelnen Abschnitten geforderten Parametern ein Sicherheitsniveau von mindestens 100 Bit. Für den Vorhersagezeitraum nach Ende 2022 wird die Verwendung von Verfahren empfohlen, die ein Sicherheitsniveau von mindestens 120 Bit erreichen.

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 ein Sicherheitsniveau von 100 beziehungsweise 120 Bit nach dem gegenwärtigen Kenntnisstand gerade erreicht wird.

<table>
<thead>
<tr>
<th>Symmetrische Verfahren</th>
<th>asymmetrische Verfahren</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ideale Blockchiffre</td>
<td>Idealer MAC</td>
</tr>
<tr>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>120</td>
<td>120</td>
</tr>
</tbody>
</table>

Tabelle 1.1.: Beispiele für Schlüssellängen für ein Sicherheitsniveau von mindestens 100 beziehungsweise 120 Bit

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

Tabelle 1.2.: Empfohlene Schlüssellängen für verschiedene kryptographische Verfahren

<table>
<thead>
<tr>
<th>Blockchiffre</th>
<th>MAC</th>
<th>RSA</th>
<th>DH $F_p$</th>
<th>DH (elliptische Kurve)</th>
<th>ECDSA</th>
</tr>
</thead>
<tbody>
<tr>
<td>128</td>
<td>128</td>
<td>2000 $^a$</td>
<td>2000 $^a$</td>
<td>250</td>
<td>250</td>
</tr>
</tbody>
</table>


Im Fall von Message Authentication Codes ist die Länge des Digest-Outputs neben der Schlüssellänge ein wichtiger Sicherheitsparameter. Idealerweise sollte ein MAC nicht praktisch von einem Angreifer von einer Zufallsfunktion mit entsprechender Digest-Länge unterschieden werden können. Solange dieses Kriterium erfüllt ist, bleibt dem Angreifer nur die Möglichkeit, gefälschte Nachrichten durch Raten zu erzeugen, und er hat dabei pro Versuch eine Erfolgswahrscheinlichkeit von $2^{-n}$, wenn $n$ die Tag-Länge ist. In vielen Anwendungen wird in einer solchen Situation $n = 64$ akzeptabel sein, also eine Taglänge von 64 Bit.


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 [63, 64].


Die Angaben dieser Technischen Richtlinie werden daher also nur beschränkt auf einen Zeitraum bis Ende 2024 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 nicht-flüchtigen 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.


- 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**  

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 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.


In einigen Fällen beschränkt sich diese Technische Richtlinie auf eine informative Beschreibung der kryptographischen Primitive. Die kryptographische Sicherheit ist aber nicht 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 [80]: 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 [60]. 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 [65]. Insgesamt wird hier empfohlen, Triple-DES nicht in neuen Systemen zu verwenden, es sei denn es wäre aus Gründen der Rücksichtskompatibilität 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. Systeme, die MD5 für kryptographische Zwecke verwenden, sind daher mit der vorliegenden Technischen Richtlinie nicht konform.

3. HMAC-SHA1: SHA-1 ist keine kollisionsresistente Hashfunktion; die Erzeugung von SHA-1-Kollisionen ist zwar mit einigem Aufwand verbunden, aber praktisch machbar [94]. 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 oder der SHA-3-Familie einzusetzen. Prinziell ist eine Verwendung von SHA-1 in der HMAC-Konstruktion oder in anderen kryptographischen Mechanismen mit vergleichbaren kryptographischen Anforderungen an die genutzte Hashfunktion (zum Beispiel im Rahmen eines Pseudozufallsgenerators oder als Teil der Mask Generation Function in RSA-OAEP) bis 2018 konform zu der vorliegenden Technischen Richtlinie.

Triple-DES mit Keying Option 2 nach [80] zeigt insgesamt deutlich ernsthaftere Schwächen gegenüber chosen-Plaintext und known-Plaintext-Angriffen im Single-Key-Setting als mit Keying Option 1 [69, 70]. 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 [80]) zu migrieren. Soweit Triple-DES noch verwendet wird, sind alle Vorgaben zur Verwendung aus [80] 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:


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.


Speichermediums ist bei Verwendung einer SSD damit potentiell wertvoller für einen Angreifer als ein einzelnes Abbild einer Festplatte.


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


2. Symmetrische Verschlüsselungsverfahren


In diesem Zusammenhang sollte betont werden, dass auch in Fällen, in denen auf den ersten Blick der Schutz der Vertraulichkeit übermittelten 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 mancher Typen aktiver Seitenkanalangriffe gegenüber können auf solche Weise entstehen, siehe etwa [96] 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


Folgende Blockchiffren werden zur Verwendung in neuen kryptographischen Systemen empfohlen:

<table>
<thead>
<tr>
<th>AES-128, AES-192, AES-256, siehe [43].</th>
</tr>
</thead>
</table>

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 Chiffretexte unter verschiedenen Schlüsseln hat, die zueinander in einer dem Angreifer


Die besten bekannten Angriffe gegen AES, die keine related-Keys benötigen, erzielen nur einen geringen Vorteil gegenüber generischen Angriffen [18].

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 ein 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 demselben 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 eine Rekonstruktion von Teilen des Klartextes durch Häufigkeitsanalyse 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 [76],
2. Cipher-Block Chaining (CBC), siehe [73], und
3. Counter Mode (CTR), siehe [73].

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. Idealerweise sollte für nicht authentisierte verschlüsselte Daten keine Entschlüsselung oder sonstige weitere Verarbeitung erfolgen. Wenn nicht authentisierte verschlüsselte Daten entschlüsselt und weiter verarbeitet werden, dann ergeben sich erhöhte Restrisiken im Hinblick auf die Ausnutzung von Fehlerorakeln.

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:
   - Initialisierungsvektoren dürfen sich innerhalb einer Schlüsselwechselperiode nicht wiederholen. Genauer dürfen in dem gesamten Mechanismus keine zwei AES-Verschlüsselungen (d.h. Anwendungen der zugrundeliegenden AES-Blockchiffre) mit gleichen Eingabewerten (Schlüssel, Nachricht) durchgeführt werden. Nichtbeachtung dieser Bedingung führt bei einer Wiederholung im Rahmen der Erzeugung des GCM-Schlüsselstroms zu einem potentiell vollständigen Verlust der Vertraulichkeit für die betroffenen Klartextblöcke!
   - 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 [76] 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 unvorhersehbare Initialisierungsvektoren zu verwenden, siehe auch Abschnitt B.2.

\(^1\)In [59] 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.
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 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 [57], padding method 2 und [73], Appendix A.
2. Padding gemäß [86], Abschnitt 6.3.
3. ESP-Padding, siehe [85] 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 [96]. Allgemeiner gilt, dass bei Verschlüsselungsverfahren, in denen ein Angreifer Änderungen am Chiffrauto durchführen kann, dass kontrollierte Änderungen am Klartext resultieren, einem Angreifer keine Seitenkanalinformation zur Verfügung stehen darf, die Aufschluss darüber liefert, ob ein gegebenes Chiffrauto 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 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


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.


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 des Schlüssels 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 \[ \text{Keysize} \]) 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.

<table>
<thead>
<tr>
<th>Verfahren</th>
<th>ECIES</th>
<th>DLIES</th>
<th>RSA</th>
</tr>
</thead>
<tbody>
<tr>
<td>$l$</td>
<td>250</td>
<td>2000</td>
<td>2000</td>
</tr>
<tr>
<td>Referenz</td>
<td>[1, 50, 66]</td>
<td>[1, 50]</td>
<td>[88]</td>
</tr>
<tr>
<td>Näheres in</td>
<td>Abschnitt 3.3</td>
<td>Abschnitt 3.4</td>
<td>Abschnitt 3.5</td>
</tr>
</tbody>
</table>


**Bemerkung 4** Für Verfahren basierend auf dem Diffie-Hellman-Problem/der Berechnung diskreter Logarithmen in elliptischen Kurven enthalten die vorliegenden empfohlenen Schlüssellängen für den Zeitraum bis 2022 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:


den Jahren die Anforderungen der vorliegenden Technischen Richtlinie in diesem Bereich als
grundsätzliche Vorsichtsmaßnahme erhöht werden.

Es wird als grundsätzliche Sicherheitsmaßnahme empfohlen, in EC-Verfahren Kurvenparameter
to 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 [38].

**Bemerkung 5**

Bei Zugrundelegung der heute bekannten Faktorisierungsverfahren und unter
der Annahme, dass es nicht zu einer Nutzung von Quantencomputern für solche Angriffe kommt,
is nicht damit zu rechnen, dass RSA-Moduln einer Länge von 2000 Bit in näherer Zukunft faktorisiert werden können. Allerdings ist der Sicherheitsspielraum kryptographischer Mechanismen mit einem angenommenen Sicherheitsniveau von etwa 100 Bit auch nicht mehr groß, sobald zusätzlich zu der Entwicklung der rechentechnischen Möglichkeiten irgendein Fortschritt auf dem Gebiet der Kryptoanalyse angenommen wird. In den letzten Jahren ist die Verwendung von

**Bemerkung 6**

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 [50, 66] 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 [50]).

Grundsätzlich wird 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 grundsätzlich zu dokumentieren und sollten in der Dokumentation unter Sicherheitsgesichtspunkten begründet werden.

Beider Auswahl der empfohlenen asymmetrischen Verschlüsselungsverfahren wurde darauf geachtet, dass lediglich probabilistische Algorithmen1 zum Einsatz kommen. Hier wird also bei jeder Berechnung eines Chiffretextes ein neuer Zufallswert benötigt. Die Anforderungen an diese Zufallszahlen sind teilweise nicht direkt durch die Erzeugung gleichverteilter Werte von fester Bitlänge zu erfüllen. Näheres zu diesen Zufallszahlen wird, wo notwendig, in den Abschnitten zu den entsprechenden Verfahren angegeben.

---

1Der RSA-Algorithmus selbst ist nicht probabilistisch, dafür aber das hier empfohlene Paddingverfahren zu RSA.
3.1. Vorbemerkung zu asymmetrischen Schlüssellängen

3.1.1. Allgemeine Vorbemerkungen


3.1.1.1. Sicherheit asymmetrischer Verfahren

Die Sicherheit von asymmetrischen kryptographischen Verfahren beruht, was die Mehrzahl der in dieser Richtlinie behandelten Verfahren betrifft (Merkle-Signaturen sind die einzige Ausnahme), 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 \( e \) 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.


- 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 Chiffren, 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 erfolgt ist. Im Gegensatz dazu gibt es in der Regel keine nachträglichen Maßnahmen zum Schutz der Vertraulichkeit eines Klartextes zu einem gegebenen Chiffra.t.

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 \( \text{ord}(U) \) 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 alle empfohlenen Schlüssellängen oberhalb des in dieser Richtlinie minimal angestrebten 100-Bit-Sicherheitslevels. Für den Zeitraum ab 2023 wird konsistent ein Sicherheitsniveau von mindestens 120 Bit angestrebt, wobei auch hier bei den Verfahren basierend auf elliptischen Kurven ein gewisser Sicherheitsspielraum verbleibt.

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 deutlich beschleunigen, siehe etwa Angriffe unter Nutzung eines Static Diffie Hellman Orakels aus [22, 33].


\[^2\text{Algorithmen, die auf einer generischen Darstellung einer Gruppe operieren, haben auf Elemente und Gruppenoperationen nur Black-Box-Zugriff. Intuitiv kann man sich etwa ein Orakel vorstellen, das verschlüsselte Gruppenelemente annimmt und das Ergebnis von Gruppenoperationen verschlüsselt ausgibt.}\]
Technische Richtlinie – Kryptographische Algorithmen und Schlüssellängen

\[
\begin{array}{|c|c|c|}
\hline
\log_2(R) & \text{ECDLP} & \text{Faktorisierung/DLP in } \mathbb{F}_p^* \\ \hline
60 & 120 & 700 \\ \hline
70 & 140 & 1000 \\ \hline
100 & 200 & 1900 \\ \hline
128 & 256 & 3200 \\ \hline
192 & 384 & 7900 \\ \hline
256 & 512 & 15500 \\ \hline
\end{array}
\]

Tabelle 3.2.: Ungefährer Rechenaufwand \( R \) (in Vielfachen des Rechenaufwandes für eine einfache kryptographische Operation, z.B. einmalige Auswertung einer Blockchiffre auf einem Block) für die Berechnung diskreter Logarithmen in elliptischen Kurven (ECDLP) beziehungsweise Faktorisierung allgemeiner zusammengesetzter Zahlen mit den angegebenen Bitlängen.

\( 2^{50} \) 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 [39], 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 \textit{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 2024 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:


- Es 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 \( \geq 128 \) 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 Robustheit gegen Quantencomputer-Angriffe angenommen wird. Die vorliegende Richtlinie enthält keine Empfehlungen zu Quantencomputer-resistenten Verschlüsselungsverfahren, da zumindest innerhalb des Vorhersagezeitraums dieser Technischen Richtlinie nicht mit einer Anwendung von Quantencomputern auf die empfohlenen Verfahren gerechnet wird. Auf lange Sicht ist eine Prognose schwierig, weil verschiedene technologische Grundlagenfragen derzeit noch offen sind. 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 quantencomputerresistente Verfahren derzeit der Fall ist. Die Verwendung hybrider Verfahren, die also eine klassische und eine quantencomputerresistente Public-Key-Verschlüsselungsmethode miteinander kombinieren, eröffnet prinzipiell einen Weg, die Sicherheitsgarantien bewährter klassischer Verfahren gegen klassische Kryptoanalyse mit den Vorteilen quantencomputerresistenter Verfahren zu verbinden. Für die Auswahl und Implementierung geeigneter Post-Quantum-Verfahren sowie eine geeignete Kombination mit klassischen Verfahren und für die Festlegung geeigneter Schlüssellängen sollte unbedingt ein Experte herangezogen werden.


Für eine ausführlichere Diskussion über langfristig sichere Schlüssellängen in asymmetrischen kryptographischen Verfahren verweisen wir auf [39, 64].
3.2. Sonstige Bemerkungen

3.2.1. Seitenkanalangriffe und Fault-Attacken


3.2.2. Public-Key-Infrastrukturen


Solche Angriffe können nur dann zuverlässig abgewehrt werden, wenn eine authentische Ver teilung 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.


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.

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

### 3.3. ECIES-Verschlüsselungsverfahren

**Allgemeine Beschreibung**  


**Komponenten**  
ECIES benötigt folgende Komponenten:


- Einen Message Authentication Code $\text{MAC}_{KM}$. Es können die in Abschnitt 5.3 empfohlenen Verfahren verwendet werden.


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 $\{1, \ldots, q-1\}$.


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 $M \in \{0,1\}^\lambda$ 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 $T := \text{MAC}_{KM}(C)$ 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 $T' := \text{MAC}_{KM}(C)$ 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 $P$ soll mindestens $q \geq 250$ gelten.

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 7** 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


**Komponenten** DLIES benötigt folgende Komponenten:

- Einen Message Authentication Code $\text{MAC}_{KM}$.
- 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 zufällig eine Primzahl $q$ von geeigneter Bitlänge (siehe Unterabschnitt zu Schlüssellängen), so dass $q$ prim ist.
3. Wähle nun ein $x \in \mathbb{Z}_p^*$ so, dass $x^k \neq 1$. Setze $g := x^k$. Dann ist $g$ ein Element der Ordnung $q$ in $\mathbb{Z}_p^*$.
4. Wähle zufällig eine natürliche Zahl $a$ mit $2 \leq a < q$ und setze $A := g^a$. Dann ist $(p, g, A, q)$ der öffentliche Schlüssel und $a$ der geheime Schlüssel.
Verschlüsselung  Sei gegeben eine Nachricht \( M \in \{0,1\}^* \) und ein öffentlicher Schlüssel \((p,g,A,q)\), der auf zuverlässige Weise dem berechtigten Empfänger \( E \) der Nachricht zugeordnet werden kann. Zur Verschlüsselung wählt der Sender \( S \) dann eine zufällige Zahl \( b \in \{1,\ldots,q-1\} \) 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 Chiffretext \( C := E_K(M) \) sowie ein MAC \( T := MAC_{KM}(C) \) 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 \( T' := MAC_{KM}(C) \) und prüft, ob \( T = T' \) ist. Ist dies nicht der Fall, bricht der Entschlüsselungsvorgang ab. Ist dagegen \( T = T' \), dann erhält \( E \) durch \( M = E^{-1}_K(C) \) die Nachricht zurück.

Schlüssellänge  Die Länge der Primzahl \( p \) sollte für einen Einsatzzeitraum bis 2022 mindestens 2000 Bit betragen, danach mindestens 3000 Bit. Die Länge der Primzahl \( q \) sollte in beiden Fällen mindestens 250 Bit betragen. Fußnote a) zu Tabelle 3.1 und Bemerkung 4 sowie Bemerkung 5 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 Größe von \( p \) und \( q \) ist das nach gegenwärtigem Kenntnisstand der Fall. Allerdings kann die Schwierigkeit des Problems, diskrete Logarithmen in \( \mathbb{F}_p^* \) zu bestimmen, durch Vorberechnungen erheblich reduziert werden, die nur von \( p \) abhängen und nicht zum Beispiel von der gewählten Untergruppe oder ihrem Erzeuger. Deshalb ist es als grundsätzliche Vorsichtsmaßnahme empfehlenswert (aber nicht für eine Konformität zu der vorliegenden Technischen Richtlinie strikt notwendig), in Fällen, in denen eine große Anzahl von Nutzern einen gemeinsamen DH-Modul nutzt, schon vor 2023 Schlüssellängen ab 3000 Bit einzusetzen anstelle der minimal notwendigen 2000 Bits.

Bemerkung 8  Das DLIES-Verfahren ist ein probabilistischer Algorithmus, d.h. für die Berechnung des Chiffretextes wird eine Zufallszahl \( k \) benötigt. Hier ist \( k \in \{1,\ldots,q-1\} \) und sollte bezüglich der Gleichverteilung auf \{1,\ldots,q−1\} gewählt werden. In Abschnitt B.4 werden drei Algorithmen zur Erzeugung von \( k \) besprochen.

Bemerkung 9  Man kann die Effizienz des am Anfang dieses Abschnitts beschriebenen Verfahrens zur Schlüsselerzeugung erhöhen, wenn mehrere Nutzer sich die Werte \((p,q,g)\) teilen, so dass sie einmalig vorberechnet werden können. Alternativ ist es auch möglich, veröffentlichte Parameter zu verwenden. Die vorliegende Technische Richtlinie empfiehlt in diesem Fall eine Verwendung der MODP-Gruppen aus [84] oder der ffdhe-Gruppen aus [11], jeweils verbunden mit einer Wahl geeigneter Schlüssellängen (MODP-1536 ist also zum Beispiel unabhängig vom vorgesehenen Einsatzzeitraum nicht geeignet). In den genannten Gruppen ist jeweils \( q = (p-1)/2 \) und \( g = 2 \). Die Verwendung eines gemeinsamen \( p \) durch mehrere Nutzer wird nur dann empfohlen, wenn \( \log_2(p) \geq 3000 \), da die Berechnung diskreter Logarithmen durch Vorberechnungsattacken vereinfacht werden kann, die nur von dem Parameter \( p \) abhängen.

3.5. RSA

Schlüsselgenerierung

1. Wähle zwei Primzahlen \( p \) und \( q \) zufällig und unabhängig voneinander aus. Nähere Hinweise zur Vorgehensweise bei der Primzahlerzeugung finden sich in Abschnitt B.5.
Die Zahlen $p$ und $q$ sollten von vergleichbarer Bitlänge sein und nicht zu nah beieinander liegen: werden $p$ und $q$ also zum Beispiel unabhängig voneinander aus einem zu engen Intervall gewählt, dann sind Angriffe basierend auf Kenntnis der führenden Bits von $p$ und $q$ denkbar. Bei einer Wahl von $p$ und $q$ entsprechend Abschnitt B.5 tritt hier keine Sicherheitslücke auf.

2. Bei der empfohlenen Schlüssellänge von 2000 Bit (s.u.), wähle den öffentlichen Exponenten $e \in \mathbb{N}$ unter den Nebenbedingungen

$$\gcd(e, (p-1) \cdot (q-1)) = 1 \text{ und } 2^{16} + 1 \leq e \leq 2^{1824} - 1.$$ 

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

$$e \cdot d = 1 \mod \gcd(p-1, q-1).$$

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 abzusehen, 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 10

(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 [20].

(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 [88]. 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 [88].

Tabelle 3.3.: Empfohlenes Formatierungsverfahren für den RSA-Verschlüsselungsalgorithmus


Schlüssellänge

Die Länge des Modulus $n$ sollte bei einem Einsatzzeitraum bis Ende 2022 mindestens 2000 Bit betragen. Danach verlangt die vorliegende Technische Richtlinie eine Schlüssellänge von mindestens 3000 Bit. Fußnote a) zu Tabelle 3.1 und Bemerkungen 4 sowie 5 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 \( H : \{0,1\}^* \rightarrow \{0,1\}^n \), 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 \( m' \in \{0,1\}^* \setminus \{m\} \) 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 120 Bit muss wegen des Geburtstagsparadoxons für eine Hashfunktion \( H : \{0,1\}^* \rightarrow \{0,1\}^n \) mindestens die Bedingung \( n \geq 240 \) gelten. Eine Fallunterscheidung je nach Einsatzzeitraum des Verfahrens ist an dieser Stelle nicht nötig, weil die in dieser Technischen Richtlinie empfohlenen Hashverfahren alle bereits eine Digest-Länge von \( \geq 256 \) Bit aufweisen.


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

---

\(^1\)Spezifikationen realer Hashfunktionen beinhalten in der Regel eine Längenbegrenzung, die aber so hoch liegt, dass sie von realen Eingabestrings nicht überschritten wird.
• SHA-256, SHA-512/256, SHA-384 und SHA-512; siehe [41].
• SHA3-256, SHA3-384, SHA3-512; siehe [44].

Tabelle 4.1.: Empfohlene Hashfunktionen

**Bemerkung 12** Die Hashfunktion SHA-224 ist nicht mehr in der Liste der empfohlenen Algorithmen enthalten. Andererseits sind zwei Familien von Hashfunktionen vertreten. Hierzu ergeben sich folgende Anmerkungen:


2. Sowohl die Hashfunktionen der SHA-2-Familie als auch die der SHA-3-Familie werden als kryptographisch stark eingeschätzt. Es gibt im Hinblick auf klassische Angriffe auf Kollisionsresistenz und Einwegeigenschaften keinen heute bekannten praktisch relevanten Unterschied zwischen den beiden Funktionenfamilien. In gewissen anderen Szenarien gibt es Unterschiede, zum Beispiel sind die Funktionen der SHA-3-Familie resistent gegen Length-Extension-Attacken.

**Bemerkung 13** (i) Für die Hashfunktion SHA-1 können Beispiele von Hashkollisionen mit einem Aufwand von ≈ 10000 CPU-Jahren Rechenzeit erzeugt werden [94]; die genauen Aufwände hängen von der gewählten Angriffsplattform und der Implementierung des Angriffes ab. Für nähere Angaben hierzu verweisen wir auf [94]. Zudem bewegt sich auch der generische Aufwand zur Erzeugung beliebiger SHA-1-Kollisionen, der für Problemstellungen noch relevant sein kann, in denen ein Angreifer ein sehr großes Maß an Kontrolle über den Inhalt kollidierender Nachrichten erreichen will, mit ≈ 2^{80} SHA-1-Auswertungen am oberen Ende einer möglichen praktischen Umsetzbarkeit. In Anwendungen, die eine kollisionsresistente Hashfunktion benötigen, sollte daher SHA-1 definitiv nicht mehr eingesetzt werden.

5. Datenauthentisierung

5.1. Vorbemerkungen

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 denselben 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.


5.2. Sicherheitsziele

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

- Sicherung der Integrität von Daten, die über ein Netzwerk übermittelt werden, auf dem Weg vom Sender zum Empfänger. Hier besitzen Sender und Empfänger in der Regel ein gemeinsames Geheimnis, und der Empfänger hat kein Interesse daran, gefälschte Übertragungen zu erzeugen. Es bietet sich daher in diesem Fall die Nutzung eines symmetrischen Verfahrens zur Datenauthentisierung an.

- Sicherung der Nichtabstreitbarkeit einer Nachricht. Hierbei soll sichergestellt werden, dass der Besitzer eines bestimmten Schlüssels zuverlässig als Urheber einer Nachricht identifiziert werden kann und dass auch der Urheber selbst eine signierte Nachricht nicht so erstellen kann, dass über die Validität der Signatur nachträglich Zweifel entstehen können. In einer solchen Situation dürfen die Prüfer einer Nachricht nicht über den entsprechenden Signaturschlüssel verfügen. Daher kommt in diesem Fall nur die Verwendung digitaler

- Absicherung eines asymmetrischen Schlüsseltausches gegen Man-in-the-Middle-Angriffe. In diesem Fall steht kein gemeinsames Geheimnis zur Verfügung, und es muss daher eine integritätsgeschützte Übermittlung der Key-Exchange-Nachrichten mittels digitaler Signaturen sichergestellt werden.

**Bemerkung 14** Spezielle Situationen können zu speziellen Anforderungen an die beteiligten Anwendungen führen. Zum Beispiel werden bei Codesignaturen die Sicherheitsziele der Integrität der übermittelten Anwendung sowie der Nichtabstellbarkeit möglicherweise enthaltener Schadfunktionen in der ausgelieferten Software verfolgt, obwohl die signierten Daten in der Regel weder beim Empfänger noch beim Ersteller sinnvoll angezeigt und mit vertretbarem Aufwand inhaltlich geprüft werden können. Die Sicherheitsfunktion der sicheren Anzeige auf Erstellerseite verlagert sich damit vollständig auf die Prozesse zur Qualitätssicherung beim Ersteller sowie auf die Sicherheit der von ihm eingesetzten technischen Komponenten.


### 5.3. Message Authentication Code (MAC)

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 [75],
- HMAC, siehe [8],
- GMAC, siehe [76].

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 ≥ 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 [45]. 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.


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:

<table>
<thead>
<tr>
<th>Verfahren</th>
<th>CMAC</th>
<th>HMAC</th>
<th>GMAC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Schlüssellänge</td>
<td>≥ 128</td>
<td>≥ 128</td>
<td>≥ 128</td>
</tr>
<tr>
<td>Taglänge empfohlen</td>
<td>≥ 96</td>
<td>≥ 96</td>
<td>≥ 96</td>
</tr>
</tbody>
</table>

Tabelle 5.2.: Parameter für empfohlene MAC-Verfahren

5.4. Signaturverfahren

In Signaturalgorithmen werden die zu signierenden Daten zunächst geasht und dann aus diesem Hashwert die Prüfsumme bzw. die Signatur mit dem geheimen Schlüssel des Beweisenden berechnet. Der Prüfer verifiziert dann die Signatur mit dem entsprechenden öffentlichen Schlüssel.
Technische Richtlinie – Kryptographische Algorithmen und Schlüssellängen

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.


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 gehaschten Daten und ein Algorithmus zum Verifizieren der Signatur.


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

| 1. RSA, siehe [53], |
| 2. DSA, siehe [54] und [42], |
| 3. DSA-Varianten auf elliptischen Kurven: |
| a) ECDSA, siehe [27], |
| b) ECKDSA, ECGDSA, siehe [27, 54], und |
| 4. Merkle-Signaturen, siehe [31]a |

*aMerkle-Signaturen unterscheiden sich in wesentlichen Aspekten von den anderen an dieser Stelle empfohlenen Signaturverfahren. Für eine genauere Beschreibung der wichtigsten Punkte wird auf Abschnitt 5.4.4 verwiesen.*

Tabelle 5.3.: Empfohlene Signaturverfahren

Bei einer geeigneten Wahl der Sicherheitsparameter erreichen nach heutigem Kenntnisstand alle hier empfohlenen Signaturverfahren ein vergleichbares Sicherheitsniveau, wenn die privaten Schlüssel zuverlässig geheim gehalten werden und also insbesondere nicht aufgrund von Implementierungsschwächen ermittelt werden können, etwa durch Seitenkanäle, Fault-Attacken oder gegen eine bestimmte Art der Schlüsselgenerierung ausgerichtete mathematische Angriffe. Für die Erstellung qualifizierter elektronischer Signaturen im Anwendungsbereich des Vertrauensdienstegesetzes können trotz der grundsätzlich für alle empfohlenen Verfahren gegebenen si-
cherheitstechnischen Eignung formal abweichende Regelungen greifen. Wir verweisen für dieses Thema auf den SOGIS-Leitfaden zur Eignung von kryptographischen Algorithmen [91].

**Bemerkung 17** Mit Ausnahme des DS 3 (vergl. Tabelle 5.4) sind die empfohlenen asymmetrischen Signaturverfahren probabilistische Algorithmen. Hier wird also bei jeder Berechnung einer Signatur ein neuer Zufallswert benötigt. Anforderungen an diese Zufallswerte werden in den entsprechenden Abschnitten angegeben.

**Bemerkung 18** Merkle-Signaturen gelten im Gegensatz zu allen anderen hier aufgeführten Signaturverfahren als sicher gegen Angriffe unter Nutzung von Quantencomputern [31]. Zudem sind sie als einziges der hier genannten Verfahren forward secure im Sinne von [9], siehe auch [58] für weitere Informationen zum Thema Forward Security.

5.4.1. 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, $e$ invertierbar mod $\varphi(n), 2^{16} < e < 2^{256}$) und der Signaturschlüssel ist $d := e^{-1} \pmod{\varphi(n)}$.

**Signaturerzeugung und Signaturverifikation** Für die Signaturerzeugung bzw. -verifikation siehe [53]. 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 [34]. Die folgenden Verfahren werden empfohlen:

1. EMSA-PSS, siehe [88].
2. Digital Signature Scheme (DS) 2 und 3, siehe [56].

Tabelle 5.4.: Empfohlene Formatierungsverfahren für den RSA-Signaturalgorithmus

**Schlüssellänge** Die Länge des Moduls $n$ sollte (Einsatzzeitraum bis 2022) mindestens 2000 Bit betragen und bei Verwendung ab 2023 mindestens 3000 Bit. Fußnote a) zu Tabelle 3.1 und Bemerkungen 4 sowie 5 aus Kapitel 3 gelten entsprechend.

5.4.2. Digital Signature Algorithm (DSA)

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

---

1Der RSA-Algorithmus selbst ist deterministisch, nicht aber die hier empfohlenen Paddingverfahren zu RSA außer DS 3.
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 $g := x^{(p-1)/q} \mod p$.

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.


Schlüssellängen Die Länge der Primzahl $p$ sollte bei einem Einsatzzeitraum bis einschließlich 2022 mindestens 2000 betragen. Für Signaturen, die ohne weitere Maßnahmen (z. B. Übersignatur) länger als bis Ende 2022 gültig bleiben sollen, wird eine Schlüssellänge $\geq 3000$ Bit empfohlen.

Bemerkung 19 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 [82]. In Abschnitt B.4 werden zwei Algorithmen zur Berechnung von $k$ besprochen.


5.4.3. 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 $\{1, \ldots, q-1\}$.


Dann bilden die EC-Systemparameter $(p, a, b, P, q, i)$ zusammen mit $G$ den öffentlichen Schlüssel und $d$ den geheime Schlüssel.
**Signaturerzeugung und Signaturverifikation**  
Folgende Algorithmen sind grundsätzlich geeignet:

1. ECDSA, siehe [27].
2. ECKDSA, ECGDSA, siehe [27, 54].

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 5.4.2 gelten entsprechend.

**Schlüssellängen**  
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, $q \geq 2^{250}$ zu wählen.

**Bemerkung 21**  
Wie das DSA-Verfahren sind alle in diesem Abschnitt empfohlenen Signaturverfahren probabilistische Algorithmen. Hier muss ebenfalls ein Zufallszahlen $k \in \{1, \ldots, q-1\}$ gemäß der Gleichverteilung gewählt werden, da andernfalls Angriffe existieren, vergleiche [82].

**5.4.4. Merklesignaturen**

Im Gegensatz zu den bisher beschriebenen Signaturverfahren beruht die Sicherheit des in [31] beschriebenen Algorithmus nur auf der kryptographischen Stärke einer Hashfunktion und einer pseudzufälligen Funktionenfamilie. Insbesondere werden keine Annahmen zur Abwesenheit effizienter Lösungsverfahren 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

Als Hashfunktionen sind alle in Tabelle 4.1 empfohlenen Hashverfahren geeignet. Die benötigte pseudzufällige Funktionenfamilie kann durch die HMAC-Konstruktion aus der verwendeten Hashfunktion konstruiert werden. Für eine genaue Beschreibung des Verfahrens siehe [31].

Die generell geringen komplexitätstheoretischen Annahmen, die der Sicherheit von Merkle-Signaturen zugrunde liegen, 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

²Eine Diskussion der Quantencomputer-Sicherheit der Kollisionsresistenz von Hashfunktionen findet sich in [12].

5.4.5. Langfristige Beweiswerterhaltung für digitale Signaturen


Nähere Informationen zu diesem Thema finden sich in der Technischen Richtlinie 03125 (TR-ESOR) [30].
6. Instanzenauthentisierung


Die Authentisierung sollte - wo das sinnvoll und möglich ist - gegenseitig erfolgen und kann mit einer Schlüsselangabe einhergehen, um die Vertraulichkeit und Integrität einer anschließenden Kommunikation zu gewährleisten, siehe Kapitel 7 für empfohlene Schlüsselaustausch- und Schlüsselangabeverfahren 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 Instanzenauthentisierung 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


<table>
<thead>
<tr>
<th>Beweisender (B)</th>
<th>Prüfer (P)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wähle Zufallswert $r$</td>
<td></td>
</tr>
<tr>
<td>Berechne Prüfsumme $c$</td>
<td></td>
</tr>
<tr>
<td>Verifiziere Prüfsumme</td>
<td></td>
</tr>
</tbody>
</table>

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

6.2. Asymmetrische Verfahren


Bemerkung 22 Auch wenn die in Abschnitt 5.4 empfohlenen Signaturverfahren zur Datenauthentisierung auch zur Instanizauf Natashaütisierung genutzt werden können, sollte darauf geachtet werden, dass die eingesetzten Schlüssel verschieden sind. Das bedeutet, dass ein Schlüssel zur Erzeugung von Signaturen nicht zur Instanzauf Natashaütisierung 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 [40], 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 23 Werden Zugriffs-Passwörter für kryptographische Komponenten nicht wenigs tens 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 [79], Tabelle A.1.


Bemerkung 25 (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 Entsperre (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 kryptografisches 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 hoher 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


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 [26].

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

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

**Bemerkung 26** 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


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 Schlüssleinigungsverfahren 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.


Vorbemerkung: Asymmetrische versus symmetrische Schlüssleinigungsverfahren

Mit asymmetrischen Schlüssleinigungsverfahren sind Sicherheitseigenschaften erreichbar, die allein unter Verwendung symmetrischer Kryptographie nicht realisierbar sind. Zum Beispiel ha-
ben beide empfohlenen asymmetrischen Schlüsselteilungsverfahren 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 Schlüsselteilungsverfahren höchstens erreicht werden, dass ein Angreifer, der alle Langzeitgeheimnisse beider Teilnehmer kennt, die Ergebnisse *vergangener* korrekt durchgeführter Schlüsselteilungen 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 5.3 zu kombinieren (im Encrypt-then-MAC-Modus), um eine manipulationssichere Übertragung des Schlüsselmaterials zu erreichen.

**Schlüsselteilung** Auch Schlüsselteilungsverfahren lassen sich rein auf Basis symmetrischer Verfahren realisieren, wenn die Existenz eines gemeinsamen langfristigen Geheimnisses vorausgesetzt werden kann. Key Establishment Mechanism 5 aus [51] 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. \(K_{t+1} := \text{KDF}(s, \text{Label}, \text{Context}, L, K_t)\).
2. Hier ist \(\text{KDF}\) eine kryptographische Schlüsselableitungsfunktion nach [78]. \(s\) ist der dabei im Extraktionsschritt genutzte Salt-Wert. Label und Context gehen in dem in [78] vorgesehenen Schlüsselexpansionsschritt nach [81] 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 Schlüsselteilungsverfahren mit öffentlichen Schlüsseln unter ausschließlicher Nutzung symmetrischer Primitive handelt [68]. 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 [78, 81] 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 [71],
2. EC Diffie-Hellman (ECKA-DH), siehe [27].

Tabelle 7.1.: Empfohlene asymmetrische Schlüsselenigungsverfahren

Es sind festzulegen:

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

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

Das Tripel $(p, g, q)$ muss vorab authentisch zwischen den Kommunikationspartnern ausgetauscht werden. Prinzipiell können gleiche Systemparameter durch viele Nutzer verwendet werden. Zur Erzeugung geeigneter Systemparameter siehe Bemerkung 9.

Schlüsselenbarung

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 Schlüsselenbarung 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.


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 $G$ 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 $Q_A \in G$, $Q_B \in G$, $Q_A \neq O$ und $Q_B \neq O$ gelten. Im Rahmen der Prüfung $Q_A, Q_B \in G$ sollte explizit auch überprüft werden, ob $Q_A, Q_B \in C$.

Weitere Hinweise finden sich in Abschnitt 4.3.2.1 von [27].

Schlüssellänge  Die Länge von $q$ sollte mindestens 250 Bit betragen.

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 kompromittiert 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 [90].

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. Wählen wir eine Primzahl $p \geq \max(2^r, n+1)$ und setzen $a_0 := \sum_{i=0}^{r-1} k_i \cdot 2^i$.
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 27** 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 $j_i$) 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 $c_j = \prod_{1 \leq l \leq t, j_l \neq j} \frac{j_l}{j_l - j}$.
2. Berechne $K = \sum_{l=1}^{t} c_j K_{j_l}$.

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

**Bemerkung 28** 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 authentifiziert stattfinden, soweit es einem Angreifer physikalisch möglich ist, diese Kommunikation aufzuzeichnen oder zu manipulieren.

Zudem ist die informationstheoretische Sicherheit nur gegeben, wenn die $a_i$ für $i > 0$ echt zufällig und entsprechend der Gleichverteilung aus $\mathbb{F}_p$ gewählt werden. Um mindestens eine komplexitätstheoretische Sicherheit zu garantieren, sollte daher zur Erzeugung der $a_i$ ein physikalischer Zufallsgenerator der Funktionalitätsklasse PTG.2 oder PTG.3 genutzt werden oder ein deterministischer Zufallsgenerator der Funktionalitätsklasse DRG.3 oder DRG.4. Die Ausgabewerte dieses Zufallsgenerators müssen so nachbearbeitet werden, dass sie der Gleichverteilung auf $\mathbb{F}_p$ entsprechen. Geeignete Verfahren hierzu werden in Abschnitt B.4 angegeben.
9. Zufallszahlengeneratoren


Üblicherweise wird bei der Erzeugung von Zufallszahlen das Ziel verfolgt, Ausgabewerte gleichverteilt auf \{0, 1\}^n zu erzeugen. Dies entspricht aber nicht immer der in einer Anwendung benötigten Verteilung. 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.


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


- 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, nachweislich 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.

- 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.


- In Anwendungen, in denen es plausibel ist, dass ein Angreifer, der geringfügige Schiefen in der Verteilung der erzeugten Zufallszahlen nutzen könnte, daraus auch nachträglich einen Vorteil ziehen kann, sollten PTG.2-Zufallsgeneratoren wo möglich durch Zufallsgeneratoren der Funktionalitätsklassen PTG.3 oder DRG.4 ersetzt werden, falls aktuell erzeugte Zufallszahlen voraussichtlich nach 2020 noch Ziel von Angriffen sein können. In diesen Fällen ist also auch der passive Einsatzzeitraum eines Zufallsgenerators für aktuelle Planungen relevant.

- 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. unvorhersehbare Zufallszahlen, zu erzeugen. In der Regel wird hierbei unvorhersehbares Verhalten einfacher

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


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

1. Die statistischen 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 Mindestschanke (nahe bei 1).


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).


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


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. [62]). 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.

1Unter einer signifikant höheren Wahrscheinlichkeit wird hier eine Wahrscheinlichkeit verstanden, die mindestens über der Wahrscheinlichkeit liegt, die für das seed update erzeugten echten Zufallszahlen zu erraten. Für jedes seed update müssen mindestens 100 Bit Min-Entropie erzeugt werden.
9.3. Nicht-physikalische nicht-deterministische Zufallszahlengeneratoren


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.


In der mathematisch-technischen Anlage [62] wird eine Funktionalitätsklasse für solche Zufallsgeneratoren (NTG.1) definiert. Für NTG.1-Zufallsgeneratoren wird grob gesagt verlangt, dass die Menge der gesammelten Entropie im laufenden Betrieb zuverlässig geschätzt wird und dass die Ausgabedaten eine Shannon-Entropie von > 0.997 Bit pro Ausgabebit aufweisen.

Vereinfacht gesagt, bedeutet dies unter anderem:


2. Zufallszahlen dürfen nur ausgegeben werden, wenn der Wert des Entropiezählers groß genug ist.


9.4. Verschiedene Aspekte


9.5. Seedgenerierung für deterministische Zufallszahlengeneratoren


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 sollten
über diese Risiken aufgeklärt werden.

9.5.1. GNU/Linux

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

Auslesen von Daten aus der Gerätedatei /dev/random

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


Bemerkung 32 Prinzipiell kann /dev/random natürlich nicht nur zum Seeding eines Pseudo- zufallsgenerators verwendet werden, sondern auch zur direkten Erzeugung kryptographischer Schlüssel.
9.5.2. Windows

Im Gegensatz zum System GNU/Linux gibt es für die Betriebssysteme der Windows-Familie derzeit keine vom BSI untersuchte Funktion, die hinreichend große Entropie gewährleistet. Zur Erzeugung sicherer Seeds sollten daher mehrere Entropiequellen in geeigneter Weise kombiniert werden. Beispielhaft wäre in Windows 10 zur Erzeugung eines Seedwertes von 128 Bit Entropie etwa die folgende Methode denkbar:

1. Einlesen von 128 Bit Zufallsdaten in einen 128-Bit-Puffer $S_1$ aus der Windows-API-Funktion `CryptGenRandom`.

2. Beziehen eines Bitstrings $S_2$ mit mindestens 100 Bit Entropie aus einer anderen Quelle. Hierbei kommen etwa in Frage:

   • Auswertung der Zeitabstände zwischen aufeinanderfolgenden Tastendrücken des Benutzers: Wenn diese mit einer Genauigkeit von einer Millisekunde erfasst werden können (was zu überprüfen ist!), dann können hierbei konservativ etwa drei Bit an Entropie pro Tastenanschlag angesetzt werden. Hierbei ist im Hinblick auf eine Einschätzung der zeitlichen Auflösung der gemessenen Zeitabstände die gesamte Verarbeitungskette auf die gesammelte Entropie limitierende Faktoren zu untersuchen. Zum Beispiel ist es denkbar, dass die Genauigkeit der internen Uhr eine Auflösungsgrenze angibt, die Abtastfrequenz der Tastatur eine andere, und der Zeitabstand, innerhalb dessen genutzte Systemtimer aktualisiert wird, wieder eine andere. Es wird empfohlen, hierbei auch in praktischen Tests die Verteilung der Tastaturanschlagszeiten zu messen und auf Auffälligkeiten zu untersuchen. Die Sequenz der zeitlichen Abstände einer hinreichend großen Anzahl von Tastaturereignissen kann dann in einen Binärstring $B$ codiert werden. Man setzt dann $S_2 := \text{SHA256}(B)$ und löscht die aufgenommenen Daten zu den Tastaturanschlagszeiten (und andere Daten, die in dem Prozess erhoben wurden) durch Überschreiben mit Nullen aus dem Arbeitsspeicher.

   • Durch den Benutzer ausgelöste Ereignisse: Die Zeitpunkte $T_1, T_2, T_3, T_4, T_5$ von fünf durch den Benutzer ausgelösten Ereignissen werden über die Windows-API mittels der Funktionen `QueryPerformanceCounter()` oder `GetSystemTimePreciseAsFileTime()` aufgenommen. Diese haben in der Regel eine Genauigkeit in der Größenordnung einer Mikrosekunde. Man kann dann unter Voraussetzungen, die im Folgenden beschrieben werden und im Einzelfall auf ihre Plausibilität hin geprüft werden müssen, annehmen, dass der Bitstring $T := T_1 || T_2 || T_3 || T_4 || T_5$ aus Angreifersicht etwa 100 Bit Entropie enthält. Die eben erwähnten Voraussetzungen hierfür sind:

   a) Jedes $T_i$ darf selbst dann nicht genauer als auf eine Sekunde geschätzt werden können, wenn für alle $j \neq i$ dem Angreifer $T_j$ bekannt ist.

   b) In dieser Situation darf der Wert von $T_i$ nicht durch andere Erwägungen (z.B. zur Polling-Frequenz der Tastatur) auf weniger als $2^{20}$ Möglichkeiten beschränkt werden können, wenn irgendein Intervall von einer Sekunde Länge angegeben wird, das $T_i$ enthält.

Man setzt wie im letzten Beispiel $S_2 := \text{SHA256}(T)$ und löscht $T$ aus dem Arbeitsspeicher.

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

- Auf ähnliche Weise können auch Mausbewegungen des Benutzers zur Gewinnung von Entropie genutzt werden. Die in Mausbewegungen enthaltene Entropie ist nicht leicht genau zu bestimmen, aber es erscheint sehr implausibel, dass die Kette von Zeigerpositionen, die zum Beispiel 60 Sekunden willkürlicher Mausbewegungen entspricht, sich verlustlos auf einen Datensatz von unter 100 Bit Größe komprimieren läßt. Man definiert dann $S_2$ wieder durch einen SHA-2-Hash über die Mausereignisse.

3. In allen diesen Fällen kann dann ein Seed-Wert $S$ für einen geeigneten Pseudozufallsgenerator abgeleitet werden, indem $S := \text{SHA256}(S_1 || S_2)$ gesetzt wird.

Anhang A.

Anwendung kryptographischer Verfahren


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 5.3 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. Sonstige Daten, die in der gleichen Datenübertragung übermittelt werden, sind nicht authentisiert.

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

Bemerkung 34 Es besteht die Möglichkeit, Verschlüsselungs- und Authentisierungsschlüssel aus einem gemeinsamen Schlüssel abzuleiten. Empfohlene Verfahren sind in Abschnitt B.1 zusammengefasst.

Bemerkung 35 Bei der authentisierten Übertragung verschlüsselter Daten wird die Nutzung eines MACs im Encrypt-then-MAC-Modus empfohlen.

Bemerkung 36 Werden Daten verschlüsselt übertragen, für die zusätzlich eine Nichtabtreibbarkeit des Klartextes ein Sicherheitsziel ist, dann sollte der Klartext durch eine digitale Signatur gesichert werden. In diesem Fall würde also zunächst der Klartext signiert, dann verschlüsselt, und schließlich die verschlüsselte Übertragung durch einen MAC vor Veränderung auf dem Übertragungsweg geschützt. Eine Signatur über das Chiffat kann sinnvoll sein, wenn darüber hinaus die chiffrierte Nachricht unabhärtbar sein soll beziehungsweise nur der Absender (und nicht
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 Ableitung symmetrischer Schlüssel für Verschlüsselungs- und Datenauthentisierungsverfahren aus diesem Geheimnis siehe Abschnitt B.1.

Voraussetzungen an die Umgebung

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.


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:

Hier wird davon ausgegangen, dass nur symmetrische Schlüssel ausgehandelt werden.
Key Establishment Mechanism 5 aus [51].

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

Bemerkung 37 Als Verschlüsselungsverfahren können in Key Establishment Mechanism 5 aus [51] alle in dieser Technischen Richtlinie empfohlenen authentisierten Verschlüsselungsverfahren verwendet werden (siehe Abschnitt A.1).

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 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 [77], Abschnitt 5.6.2 beschriebene Verfahren empfohlen. In dem zweiten empfohlenen Verfahren (KAS2-bilateral-confirmation nach [77]) ist dieser Schritt bereits enthalten.

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

1. Elliptic Curve Key Agreement of ElGamal Type (ECKA-EG), siehe [27].
2. Instanzaauthentisierung mit RSA und Schlüsselvereinbarung mit RSA, siehe KAS2-bilateral-confirmation nach [77], Abschnitt 8.3.3.4.
3. MTI(A0), siehe [52], Annex C.6.
**Bemerkung 38** 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 39** 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 von Zufallsgeneratoren oder Schlüsselleinigungsverfahren gelieferten Werten 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 Schlüsselleinigungsverfahren 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:


Folgendes Verfahren wird für alle Anwendungen von Schlüsselableitungsfunktionen empfohlen:

Key Derivation through Extraction-then-Expansion nach [78].

Tabelle B.1.: Empfohlenes Verfahren zur Schlüsselableitung

Es wird empfohlen, als MAC-Funktion in dem angegebenen Verfahren einen der in Abschnitt 5.3 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 praktisch nicht in der Lage sein darf, zukünftig eingesetzte Initialisierungsvektoren zu erraten. Darüber hinaus darf der Angreifer auch nicht in der Lage sein, die Wahl der Initialisierungsvektoren zu beeinflussen.
Zwei Verfahren werden zur Erzeugung unvorhersehbarer 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.


Tabelle B.2.: Empfohlene Verfahren zur Erzeugung unvorhersehbarer 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 \( q := \text{ord}(P) \) die Ordnung des Basispunktes \( P \) in \( E(\mathbb{F}_p) \) bezeichnet und \( i := \#E(\mathbb{F}_p)/q \) der sogenannte **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 \( q = \text{ord}(P) \) des Basispunktes \( P \) ist eine von \( p \) verschiedene Primzahl,
2. \( p^r \neq 1 \mod q \) für alle \( 1 \leq r \leq 10^4 \), und
3. die Klassenzahl der Hauptordnung, die zum Endomorphismenring von \( E \) gehört, ist mindestens 200.

Siehe [38] für eine Erläuterung.

EC-Systemparameter, die die obigen Bedingungen erfüllen, heißen auch **kryptographisch stark**.
**Bemerkung 40** Es wird empfohlen, die 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:

<table>
<thead>
<tr>
<th>Nummer</th>
<th>Systemparameter</th>
<th>Quelle</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>brainpoolP256r1</td>
<td>[38]</td>
</tr>
<tr>
<td>2</td>
<td>brainpoolP320r1</td>
<td>[38]</td>
</tr>
<tr>
<td>3</td>
<td>brainpoolP384r1</td>
<td>[38]</td>
</tr>
<tr>
<td>4</td>
<td>brainpoolP512r1</td>
<td>[38]</td>
</tr>
</tbody>
</table>

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 8, 7, 19 und 21 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. \( k = k' \mod q \) (d.h. \( 0 \leq k < q \)).
3. return \( k \).

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

**Bemerkung 41** (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. Erzeugung von Primzahlen

B.5.1. Vorbemerkungen

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. Andererseits ist es im Hinblick auf die Benutzerfreundlichkeit der Anwendung auch wichtig, die Primzahlerzeugung effizient durchzuführen. Hierbei können proprietäre Geschwindigkeitsoptimierungen in der Schlüsselerzeugung zu signifikanten kryptographischen Schwächen führen, siehe etwa [72]. Es wird daher als generelle Vorbemerkung dringend empfohlen, Verfahren einzusetzen, die öffentlich bekannt und hinsichtlich ihrer Sicherheit untersucht sind.


B.5.2. Konforme Verfahren

Drei Verfahren sind zur Erzeugung zufälliger Primzahlen gemäß der vorliegenden Technischen Richtlinie zulässig. Dabei handelt es sich kurz zusammengefasst um die folgenden:

1. Die gleichverteilte Erzeugung zufälliger Primzahlen durch die Verwerfungsmethode (engl. rejection sampling);

2. Die gleichverteilte Auswahl einer invertierbaren Restklasse \(r\) bezüglich \(B\#\), wo \(B\#\) die Primfakultät von \(B\) ist, also das Produkt aller Primzahlen kleiner \(B\), gefolgt von der Wahl einer Primzahl von geeigneter Größe mit Rest \(r\) mod \(B\#\) durch die Verwerfungsmethode;

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

Die ersten beiden Verfahren werden hierbei gleichermaßen empfohlen. Das dritte Verfahren erzeugt gewisse statistische Schiefen in der Verteilung der erzeugten Primzahlen, die grundsätzlich unerwünscht sind. Es ist aber weit verbreitet (siehe etwa Table 1 in [96]) und es gibt keine Hinweise darauf, dass sich die induzierten statistischen Schiefen für Angriffe nutzen lassen.
Dieses Verfahren wird daher als Legacy-Verfahren in der vorliegenden Technischen Richtlinie akzeptiert.

Die folgenden Tabellen geben einen Überblick über die drei durch diese Richtlinie unterstützten Verfahren:

### Tabelle B.5.: Empfohlenes Verfahren 1: Erzeugung von Primzahlen durch die Verwerfungsmethode

<table>
<thead>
<tr>
<th>Eingabeparameter: Ein Intervall $I := [a, b] \cap \mathbb{N}$, in dem eine Primzahl gefunden werden soll.</th>
</tr>
</thead>
<tbody>
<tr>
<td>2. Falls $p$ keine Primzahl ist, gehe zurück zu 1.</td>
</tr>
</tbody>
</table>

### Tabelle B.6.: Empfohlenes Verfahren 2: Erzeugung von Primzahlen durch eine effizienzoptimierte Verwerfungsmethode

<table>
<thead>
<tr>
<th>Eingabeparameter: Ein Intervall $I := [a, b] \cap \mathbb{N}$, in dem eine Primzahl gefunden werden soll, sowie eine kleine natürliche Zahl $B$, für die $\Pi := B# \ll b - a$ gilt. $\Pi$ kann hierbei ein fest vorgegebener Wert sein, sofern klar ist, dass es deutlich kleiner ist als $b - a$.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Wähle in $\mathbb{Z}/(\Pi)$ zufällig gemäß der Gleichverteilung ein invertierbares Element $r$. Dies ist äquivalent dazu, zufällig ein $r &lt; \Pi$ zu wählen, das zu $\Pi$ teilerfremd ist.</td>
</tr>
<tr>
<td>2. Wähle $k \in \mathbb{N}$ zufällig so aus, dass $p := k\Pi + r \in I$ ist. $k$ ist also gemäß der Gleichverteilung auf $[(a - r)/\Pi, (b - r)/\Pi]$ zu wählen.</td>
</tr>
<tr>
<td>3. Falls $p$ keine Primzahl ist, gehe zurück zu 2.</td>
</tr>
</tbody>
</table>

70 Bundesamt für Sicherheit in der Informationstechnik (BSI)
Eingabeparameter: Ein Intervall $I := [a, b] \cap \mathbb{N}$, in dem eine Primzahl gefunden werden soll, sowie eine kleine natürliche Zahl $B$, für die $\Pi := B\# \ll b - a$ gilt. II kann hierbei ein fest vorgegebener Wert sein, sofern klar ist, dass es deutlich kleiner ist als $b - a$.

1. Wähle in $\mathbb{Z}/(\Pi)$ zufällig gemäß der Gleichverteilung ein invertibles Element $r$. Dies ist äquivalent dazu, zufällig ein $r < \Pi$ zu wählen, das zu $\Pi$ teilerfremd ist.
2. Wähle $k \in \mathbb{N}$ zufällig so aus, dass $p := k\Pi + r \in I$ ist. $k$ ist also gemäß der Gleichverteilung auf $\lceil (a - r)/\Pi \rceil$, $\lfloor (b - r)/\Pi \rfloor$ zu wählen.
3. Teste, ob $p$ eine Primzahl ist. Wenn nicht, addiere $\Pi$ und wiederhole Schritt 3. Falls das Intervall $I$ verlassen wird, gehe zurück zu Schritt 1.

Tabelle B.7.: Legacy-Verfahren: Erzeugung von Primzahlen durch inkrementelle Suche

Als eigentlicher Primzahltest in den oben beschriebenen Algorithmen wird aus Effizienzgründen meist ein probabilistischer Primzahltest eingesetzt. Der folgende Algorithmus wird empfohlen:

Miller-Rabin, siehe [71], Algorithmus 4.24.

Tabelle B.8.: Empfohlener probabilistischer Primzahltest

**Bemerkung 42**

(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 $\{2, 3, \ldots, p-2\}$.


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


Es muss aber in diesem Zusammenhang beachtet werden, dass das Sicherheitsniveau der erzeugten RSA-Moduln in diesem Fall möglicherweise durch das Sicherheitsniveau der Zufallszahlenerzeugung beschränkt wird. Dies wäre etwa der Fall, wenn ein Zufallsgenerator mit 100 Bit Sicherheitsniveau zur Erzeugung von RSA-Schlüsseln einer Länge von 4096 Bit genutzt würde.


B.5.3. Erzeugung von Primzahlpaaren

Um die Sicherheit von Schlüsselpaaren zu gewährleisten, für die die zugrundeliegenden RSA-Moduln berechnet wurden durch Multiplikation zweier unabhängig voneinander mit einem der geeigneten Verfahren erzeugten Primzahlen, ist es wichtig, dass das Intervall \( I := [a, b] \cap \mathbb{N} \) nicht zu klein ist. Wenn Schlüsselpaare erzeugt werden sollen, deren Modulus \( N \) eine vorher festgelegte Bitlänge \( n \) aufweist, dann bietet es sich an, \( I = \left[ \left\lceil 2^{(n/2)} \right\rceil, \left\lfloor 2^{(n/2)} \right\rfloor \right] \cap \mathbb{N} \) zu wählen.

Eine andere Wahl von \( I \) ist zu dieser Technischen Richtlinie konform, wenn für \( p \) und \( q \) das gleiche Intervall \( I \) genutzt wird und \( Card(I) \geq 2^{-8b} \) ist.

B.5.4. Hinweise zur Sicherheit der empfohlenen Verfahren

Bezeichne im Folgenden \( \pi \) die Primzahlfunktion, also \( \pi(x) := Card(\{ n \in \mathbb{N} : n \leq x, n \text{ prim} \}) \). Nach dem Primzahlsatz ist \( \pi(x) \) asymptotisch äquivalent zu \( x/\ln(x) \), d.h. \( \frac{\ln(x) \cdot \pi(x)}{x} \) geht gegen 1, wenn \( x \to \infty \). Die Sicherheit der hier empfohlenen Verfahren zur Primzahlerzeugung stützt sich auf die folgenden Beobachtungen:

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


- Das im vorhergehenden Punkt angegebene Argument für die Sicherheit des Verfahrens B.6 liefert strenggenommen keine Garantie dafür, dass für ein konkretes II und ein konkretes Intervall I tatsächlich die Häufigkeit von Primzahlen während der Suche nicht von der gewählten Restklasse \(r \mod \Pi\) abhängig ist. In der Tat ist es klar, dass diese asymptotische Aussage nicht gültig sein wird, wenn II sich der Größenordnung von \(b - a\) annähert. Es wird aber erwartet, dass keine wesentlichen Unterschiede bezüglich der Primzahldichte zwischen den verschiedenen Restklassen existieren, wenn die Anzahl der Primzahlen in den einzelnen Restklassen groß ist. Das Intervall I enthält \(\pi(b) - \pi(a)\) Primzahlen, für jede Restklasse \(\mod \Pi\) werden daher \(\frac{\pi(b) - \pi(a)}{\varphi(\Pi)}\) Primzahlen erwartet. Für Zahlen der Großenordnung von etwa 1000 Bits kann dieser Erwartungswert mit einem geringen relativen Fehler auf \(\frac{b \ln(a) - a \ln(b)}{\ln(a) \ln(b) \varphi(\Pi)}\) geschätzt werden, solange \(\varphi(\Pi)\) klein ist im Vergleich zum Zähler dieses Bruches. Es wird empfohlen, II so zu wählen, dass \(\frac{b \ln(a) - a \ln(b)}{\ln(a) \ln(b) \varphi(\Pi)} \geq 2^{64}\).

- Die oben wiedergegebenen qualitativen Erwägungen sind ausreichend, um das Verfahren B.6 als geeignet einzuschätzen. In der Literatur gibt es aber genauere Untersuchungen zu eng verwandten Mechanismen zur Primzahlerzeugung, siehe etwa [46].

- Das Verfahren B.7 erzeugt Primzahlen, die nicht gleichverteilt sind. Die Wahrscheinlichkeit einer Primzahl \(p\) in dem Intervall I, durch dieses Verfahren ausgegeben zu werden, ist proportional zur Länge des primzahlfreien Abschnitts in der arithmetischen Folge \(p - k\Pi, p - (k - 1)\Pi, \ldots, p - \Pi, p\), der durch \(p\) beendet wird. Da die Primzahldichte in diesen arithmetischen Folgen für große II tendenziell zunimmt, wird erwartet, dass dieser Effekt für \(\Pi = 2\) am ausgeprägtesten ist. Auch hier bedeutet er aber in der Praxis nur einen sehr begrenzten Entropieverlust. Man kann die Verteilungsschiefe nach oben begrenzen, indem die Suche abgebrochen und mit einem neuen Startwert wieder aufgenommen wird, falls nach einer angemessenen Zahl \(k\) von Schritten keine Primzahl gefunden wurde: in diesem Fall werden alle Primzahlen, die einer Lücke der Länge \(\geq k\) folgen, mit gleicher Wahrscheinlichkeit ausgegeben. Die durch das Verfahren B.7 erzeugten Schiefen in der Verteilung der Primzahlen erscheinen im Hinblick auf eine Ausnutzbarkeit durch einen Angreifer nicht als besorgniserregend.
Anhang C.

Protokolle für spezielle kryptographische Anwendungen

Wir behandeln hier Protokolle, die als Bausteine kryptographischer Lösungen benutzt werden können. Zurzeit betrifft dies nur das Protokoll SRTP, da entsprechende Informationen für TLS, IPsec und SSH in die Teile 2-4 der vorliegenden Technischen Richtlinie ausgelagert wurden. 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


Wir empfehlen folgende Verwendung von SRTP:

- Zum Integritätschutz 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.
- 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 Integritätschutz in SRTP zu nutzen.

Literaturverzeichnis


[17] 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


Technische Richtlinie – Kryptographische Algorithmen und Schlüssellängen


[74] 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


[77] NIST Recommendation for Pair-Wise Key Establishment Schemes Using Integer Factorization Cryptography, Special Publication SP800-56B Rev. 1, National Institute of Standards and Technology, Technology Administration, U.S. Department of Commerce, 10/2014


[80] NIST Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher, Special Publication SP800-67 Rev.1, National Institute of Standards and Technology, Technology Administration, U.S. Department of Commerce, 01/2012.


