Bundesamt
für Sicherheit in der
Informationstechnik

BSI - Technische Richtlinie

Bezeichnung:	**Kryptographische Verfahren: Empfehlungen und Schlüssellängen**
Kürzel:	BSI TR-02102
Version:	1.0
Stand:	20.06.2008

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

Inhaltsverzeichnis

Tabellenverzeichnis

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.

Die vorliegende Technische Richtlinie richtet sich in erster Linie an Entwickler, die ab 2008 die Einführung neuer kryptographischer Infrastrukturen 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.

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. Datenauthentisierung,
4. Instanzauthentisierung,
5. Schlüsselaustausch und Schlüsseleinigung,
6. Secret Sharing und
7. 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 übertragenen Daten müssen also zusätzlich mit einem geeigneten Verfahren authentisiert werden.

Ein weiteres Beispiel sind Schlüsseleinigungsverfahren. Hier ist es wichtig zu wissen, mit wem das Schlüsseleinigungsverfahren durchgeführt wird, um sogenannte Man-in-the-Middle-Attacken ausschließen zu können. Dies geschieht durch Verfahren, die Schlüsseleinigung und Instanzauthentisierung kombinieren.

Deshalb werden in Anhang A für diese beiden Einsatzszenarien entsprechende Verfahren angeben, 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 häufig verwendete Algorithmen empfohlen, die zum Beispiel zur Erzeugung

von Primzahlen und andere Systemparameter für asymmetrischen Verfahren, zur Schlüsselableitung für symmetrische Verfahren usw. benötigt werden.

Es ist vorgesehen, die in dieser Technischen Richtlinie getroffenen Empfehlungen jährlich zu überprüfen und gegebenenfalls anzupassen.

1.1 Sicherheitsziele und Auswahlkriterien

Die Sicherheit kryptographischer Verfahren hängt primär von der Stärke der zugrunde liegenden Algorithmen ab. Aus diesem Grund werden hier nur Verfahren empfohlen, die aufgrund der heute vorliegenden Ergebnisse langjähriger Diskussionen und Analysen entsprechend eingeschätzt werden können. Weitere Faktoren für die Sicherheit sind natürlich die konkreten Implementierungen der Algorithmen und die Zuverlässigkeit eventueller Hintergrundsysteme, wie zum Beispiel benötigte Public Key Infrastrukturen für den sicheren Austausch von Zertifikaten. Die Umsetzung konkreter Implementierungen werden hier aber genauso wenig betrachtet wie eventuell auftretende patentrechtliche Probleme. **Zwar wurde bei der Auswahl der Verfahren darauf geachtet, dass die Algorithmen weitestgehend frei von Patenten sind, garantieren kann das BSI dies aber nicht.**

Insgesamt erreichen alle in dieser Technischen Richtlinie empfohlenen kryptographischen Verfahren mit den in den einzelnen Abschnitten geforderten Schlüssellängen ein Sicherheitsniveau von 100 Bit, sind also grundsätzlich geeignet in Systemen, die Daten mit einem Schutzbedarf bis einschließlich hoch verarbeiten, siehe [13] für eine Definition.

Tabelle 1.1 zeigt die Schlüssellängen ausgewählter Algorithmen für das in diesem Dokument geforderte Sicherheitsniveau von 100 Bit.

Symmetrische Verfahren		**asymmetrische Verfahren**		
Verschlüsselung	MAC	RSA	DSA	ECDSA
100	100	2048	224/2048	200

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

1.2 Allgemeine Hinweise

Bei der Festlegung der Größe von Systemparametern (wie zum Beispiel Schlüssellänge, Größe des Bildraums für Hashfunktionen usw.) 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 [38].

Unter der Annahme, dass es bei der Entwicklung von Quantencomputern in den nächsten 10 Jahren zu keiner nennenswerten Anwendung kommt, lässt sich die Leistungsfähigkeit von Rechnern über einen Zeitraum von 10 Jahren relativ gut vorhersagen. Dies gilt leider nicht für kryptoanalytische Verfahren. Jede Vorhersage über einen Zeitraum von 6 Jahren hinaus ist schwierig, insbesondere bei asymmetrischen Verfahren. **Außerdem können sich auch die Prognosen für den erwähnten Zeitraum von 6 Jahren aufgrund unvorhersehbarer dramatischer Entwicklungen als falsch erweisen.**

Die Empfehlungen dieser Technischen Richtlinie werden daher also nur beschränkt auf einen Zeitraum von 6 Jahren ausgesprochen. Da ein Angreifer über das Internet übertragene Daten speichern kann, bleibt ein grundsätzliches Risiko für den langfristigen Schutz der Vertraulichkeit. Als unmittelbare Konsequenzen ergeben sich:

- Die Übertragung vertraulicher Daten sollte auf das notwendige Maß beschränkt werden.

- Die Infrastruktur muss so ausgelegt sein, dass der Übergang zu größeren Schlüssellängen und stärkeren kryptographischen Verfahren möglich ist.
- Für Daten, deren Vertraulichkeit langfristig gesichert bleiben soll, ist zu empfehlen, von vornherein für die Verschlüsselung der Übertragung über allgemein zugängliche Kanäle, wie das Internet, aus den in dieser Technischen Richtlinie empfohlenen Verfahren möglichst starke zu wählen, also z.B. AES-256 statt AES-128.

Die Sicherheitsbewertung der in diesem Dokument empfohlenen kryptographischen Verfahren erfolgt ohne Berücksichtigung konkreter Einsatzbedingungen. Für konkrete Szenarien können sich andere Sicherheitsanforderungen ergeben. Ziel ist lediglich, mit Hilfe dieses Dokumentes die Evaluierung und Zertifizierung der technischen Komponenten zu unterstützen. **Es wird empfohlen, schon während der Planung kryptographischer Infrastrukturen die Zusammenarbeit mit entsprechenden Experten auf diesem Gebiet zu suchen.**

Die in dieser Technischen Richtlinie empfohlenen kryptographischen Verfahren müssen in vertrauenswürdigen technischen Komponenten implementiert werden, um das geforderte Sicherheitsniveau zu erreichen. Weiter sind die Implementierungen der kryptographischen Verfahren und Protokolle selbst in die Sicherheitsanalyse mit einzubeziehen, um z.B. Seitenkanalangriffe zu verhindern. Die Sicherheit von technischen Komponenten und Implementierungen muss jeweils dem vorgesehenen Sicherheitsniveau entsprechend durch Common Criteria Zertifikate oder ähnliche Verfahren des BSI, wie zum Beispiel im Zuge einer Zulassung, nachgewiesen werden.

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 muss darauf geachtet werden, dass für die unterschiedlichen Anwendungen jeweils verschiedene Schlüssel eingesetzt werden.

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

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

2 Symmetrische Verschlüsselungsverfahren

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

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 Schlüsselaustauschverfahren eingesetzt werden, siehe Kapitel 3.

2.1 Blockchiffren

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

Nach dem derzeitigen Wissensstand gelten die folgenden Blockchiffren als sicher:

1. AES-128, AES-192, AES-256, siehe [24],
2. SERPENT-128, SERPENT-192, SERPENT-256, siehe [3], und
3. Twofish-128, Twofish-192, Twofish-256, siehe [51].

Tabelle 2.1: Empfohlene Blockchiffren

2.1.1 Betriebsarten

Wie in Abschnitt 2.1 bereits beschrieben, werden für Klartexte, deren Bitlänge die Blockgröße der eingesetzten Blockchiffre übersteigen, sogenannte Betriebsarten verwendet. Dazu wird zunächst der Klartext in Blöcke partitioniert, deren Bitlänge der Blockgröße der eingesetzten Blockchiffre entsprechen. Der letzte Block hat eventuell eine kleinerer Bitgröße und muss entsprechend aufgefüllt werden, siehe Abschnitt 2.1.3 für geeignete Verfahren.

Die einfachste Möglichkeit, den Klartext zu verschlüsseln, ist nun, jeden Klartextblock mit dem selben Schlüssel zu verschlüsseln (diese Betriebsart heißt auch Electronic Code Book (ECB)). Dies führt aber dazu, dass gleiche Klartextblöcke zu gleichen Chiffretextblöcken verschlüsselt werden. Der Chiffretext liefert damit zumindest Informationen über die Struktur des Klartextes. 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. Cipher-Block Chaining (CBC), siehe [43],
2. Galois-Counter-Mode (GCM), siehe [46], und
3. Counter Mode (CTR), siehe [43].

Tabelle 2.2: Empfohlene Betriebsarten für Blockchiffren

Bemerkung 1 Der Galois-Counter-Mode (GCM) liefert zusätzlich eine Datenauthentisierung und wird entsprechend genauer in Abschnitt A.1 behandelt.

2.1.2 Initialisierungsvektoren

Für die unter 2.1.1 aufgeführten Betriebsarten werden Initialisierungsvektoren benötigt. Diese Werte müssen zwar nicht vertraulich behandelt werden, unterliegen aber bestimmten Nebenbedingungen, die in der folgenden Tabelle zusammengefasst sind.

1. Für CBC: Es sind nur unvorhersagbare Initialisierungsvektoren zu verwenden, siehe auch Abschnitt B.2.
2. Für GCM: Die Zählerstände im Zähleranteil des Initialisierungvektors dürfen sich bei gleichem Schlüssel nicht wiederholen.
3. Für CTR: Die Zählerstände dürfen sich bei gleichem Schlüssel nicht wiederholen.

Tabelle 2.3: Empfohlene Verfahren für die Erzeugung von Initialisierungsvektoren

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

2.1.3 Paddingverfahren

Wie bereits in Abschnitt 2.1.1 erläutert, kann es bei der Partitionierung eines zu verschlüsselnden Klartextes geschehen, dass der letzte Klartextblock kleiner als die Blockgröße der eingesetzten Chiffre ist. Das Auffüllen dieses letzten Blocks zu der geforderten Größe heißt auch *Padding*.

Folgende Paddingverfahren werden empfohlen:

1. ISO-Padding, siehe [40] und [46],
2. Padding gemäß [48], und
3. ESP-Padding, siehe [49].

Tabelle 2.4: Empfohlene Paddingverfahren für Blockchiffren

2.2 Stromchiffren

In Stromchiffren wird aus einem Schlüssel und einem Initialisierungsvektor zunächst ein sogenannter Schlüsselstrom generiert, das heißt eine pseudozufällige Folge von Bits, die dann auf die zu verschlüsselnde Nachricht Modulo 2 addiert wird.

Zur Zeit werden keine Stromchiffren empfohlen.

3 Asymmetrische Verschlüsselungsverfahren

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

Für die Spezifizierung von asymmetrischen Verschlüsselungsverfahren sind folgende Algorithmen festzulegen:

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

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

Zusätzlich geben wir Empfehlungen für minimale Schlüssellängen an.

Tabelle 3.1 gibt einen Überblick über die im Folgenden empfohlenen asymmetrischen Verschlüsselungsverfahren.

1. RSA, siehe [50],
2. DLIES, siehe [26], und
3. ECIES, siehe [26].

Tabelle 3.1: Empfohlene asymmetrische Verschlüsselungsverfahren

Bemerkung 2 Bei der Auswahl der empfohlenen asymmetrischen Verschlüsselungsverfahren wurde darauf geachtet, dass lediglich sogenannte probabilistische Algorithmen[1] zum Einsatz kommen. Hier wird also bei jeder Berechnung eines Chiffretextes ein **neuer** Zufallswert benötigt. Anforderungen an diese Zufallswerte werden in den entsprechenden Abschnitten angegeben.

3.1 RSA-Verschlüsselungsverfahren

Die Sicherheit dieses Verfahrens beruht auf der vermuteten Schwierigkeit des Faktorisierungsproblems ganzer Zahlen.

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

Schlüsselgenerierung

1. Wähle zwei Primzahlen p und q zufällig und unabhängig voneinander unter der Nebenbedingung
$$0.1 < |\log_2 p - \log_2 q| < 30.$$
2. Wähle den öffentlichen Exponenten $e \in \mathbb{N}$ unter den Nebenbedingungen
$$\mathrm{ggT}(e, (p-1)\cdot(q-1)) = 1 \text{ und } 2^{16} + 1 \leq e \leq 2^{1824} - 1.$$
3. Berechne den geheimen Exponenten $d \in \mathbb{N}$ in Abhängigkeit von e unter der Nebenbedingung
$$e \cdot d = 1 \bmod \mathrm{kgV}(p-1, q-1).$$

Mit $n = p \cdot q$ (dem sogenannten Modulus) ist dann (n, e) der öffentliche Schlüssel und d der geheime Schlüssel. Weiter müssen natürlich auch die beiden Primzahlen p und q geheim gehalten werden, da sonst jeder aus dem öffentlichen Schlüssel (n, e) wie unter Punkt 3. den geheimen Exponenten berechnen kann.

Bemerkung 3 (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 [11].

(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 [50]. 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 [50].

Tabelle 3.2: Empfohlenes Formatierungsverfahren für den RSA-Verschlüsselungsalgorithmus

Schlüssellänge Die Länge des Modulus n sollte mindestens 2048 Bit betragen.

3.2 DLIES-Verschlüsselungsverfahren

Die Sicherheit dieses Verfahrens beruht auf der vermuteten Schwierigkeit des Diskreten Logarithmenproblems in Körpern $\mathbb{F}_p$ mit Primordnung p.

Schlüsselgenerierung

1. Wähle eine Primzahl $p \in \mathbb{N}$.
2. Wähle einen Erzeuger g der multiplikativen Gruppe $\mathbb{F}_p^*$.
3. Wähle eine zufällige Zahl $a \in \{1, \ldots, p-2\}$ und setze $A := g^a$.

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

Verschlüsselung und Entschlüsselung Für die Ver- und Entschlüsselung siehe [26].

Schlüssellänge Die Länge der Primzahl p sollte mindestens 2048 Bit betragen.

Bemerkung 4 Das DLIES-Verfahren ist ein sogenannter probabilistischer Algorithmus, d.h. für die Berechnung des Chiffretextes wird eine Zufallszahl k benötigt. Hier ist $k \in \{1, \ldots, p-2\}$ und sollte bezüglich der Gleichverteilung auf $\{1, \ldots, p-2\}$ gewählt werden. Siehe Abschnitt B.4 für einen empfohlenen Algorithmus zur Berechnung des Zufallswertes k.

3.3 ECIES-Verschlüsselungsverfahren

Die Sicherheit dieses Verfahrens beruht auf der vermuteten Schwierigkeit des Diskreten Logarithmenproblems auf elliptischen Kurven.

Schlüsselgenerierung

1. Erzeuge kryptographisch starke EC-Systemparameter (p, a, b, P, q, i), siehe Abschnitt B.3.
2. Wähle d zufällig und gleichverteilt in $\{1, \ldots, q-1\}$.
3. Setze $G := d \cdot P$.

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

Verschlüsselung und Entschlüsselung Für die Ver- und Entschlüsselung siehe [26].

Schlüssellänge Für die Ordnung q des Basispunktes P muss $q \geq 2^{224}$ gelten.

Bemerkung 5 Wie DLIES ist das hier vorgestellte Verfahren ein probabilistischer Algorithmus. Hier muss ebenfalls ein Zufallswert $k \in \{1, \ldots, q-1\}$ gemäß der Gleichverteilung gewählt werden. Siehe Abschnitt B.4 für einen empfohlenen Algorithmus zur Berechnung des Zufallswertes k.

4 Hashfunktionen

Hashfunktionen bilden einen Bitstring $m \in \{0,1\}^*$ beliebiger Länge auf einen Bitstring $h \in \{0,1\}^n$ fester Länge $n \in \mathbb{N}$ ab. Diese Funktionen spielen in vielen kryptographischen Verfahren eine große Rolle, so zum Beispiel bei der Ableitung kryptographischer Schlüssel oder bei der Datenauthentisierung.

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

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

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

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

Eine Hashfunktion H, die alle obigen Bedingungen erfüllt, heißt *kryptographisch stark*. Dabei impliziert die Kollisionsresistenz die 2nd-Preimage-Eigenschaft.

Der Begriff „praktisch unmöglich“ bedeutet hier, dass es keinen Algorithmus A gibt, der eine effiziente Laufzeit hat und den Wert bzw. die Werte berechnet. Für das in dieser Technischen Richtlinie geforderte Sicherheitsniveau von 100 Bit muss wegen des Geburtstagsparadoxon für eine Hashfunktion $H : \{0,1\}^* \longrightarrow \{0,1\}^n$ mindestens die Bedingung $n \geq 200$ gelten.

Bemerkung 6 Nicht für alle der in dieser Technischen Richtlinie empfohlenen kryptographischen Protokolle, in denen Hashfunktionen eingesetzt werden, ist es erforderlich, dass alle drei Forderungen erfüllt sind. So braucht zum Beispiel bei deterministischen Zufallszahlengeneratoren, die auf Hashfunktionen basieren, die 2nd-Preimage-Eigenschaft und damit auch die Kollisionsresistenz nicht unbedingt erfüllt sein. Wir wollen aber im Folgenden die verschiedenen Anforderungen nicht unterscheiden und betrachten deshalb nur Hashfunktionen, die alle drei obigen Eigenschaften haben.

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

1. SHA-224, SHA-256, SHA-384, SHA-512, siehe [22].

Tabelle 4.1: Empfohlene Hashfunktionen

Bemerkung 7 (*i*) Die Kollisionsangriffe der Arbeitsgruppe um die chinesische Kryptologin X. Wang (siehe [53]) auf die Hashfunktion SHA-1 (spezifiziert in [22]) haben eine dynamische Entwicklung bei der Analyse von Hashfunktionen ausgelöst. Zusätzlich bildet SHA-1 (wie auch die in [27] spezifizierte Funktion RIPEMD-160) Bitstrings beliebiger Länge auf Bitstrings der Länge 160 ab. Schon allein aus diesem Grund erfüllen diese Funktionen das in dieser Technischen Richtlinie geforderte Sicherheitsniveau von 100 Bit nicht.

(*ii*) Man beachte, dass schon eine einzige Kollision einer Hashfunktion zu einer Unsicherheit bei Signaturverfahren führen kann, vergleiche [17] und [25].

5 Datenauthentisierung

Unter Datenauthentisierung verstehen wir in dieser Technischen Richtlinie kryptographische Verfahren, die garantieren, dass übersandte oder gespeicherte Daten nicht verändert wurden. Genauer benutzt ein Beweisender (üblicherweise der Sender der Daten) einen kryptographischen Schlüssel zur Berechnung der Prüfsumme der zu authentisierenden Daten. Ein Prüfer (üblicherweise der Empfänger der Daten) prüft dann, ob die empfangene Prüfsumme der zu authentisierenden Daten mit der übereinstimmt, die er bei Unverfälschtheit der Daten und Verwendung des richtigen Schlüssels erwarten würde.

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

5.1 Message Authentication Code (MAC)

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

Grundsätzlich gelten die folgenden Verfahren als sicher, wenn unter 1. eine aus Tabelle 2.1 aufgeführte Blockchiffre einsetzt wird, bzw. unter 2. eine aus Tabelle 4.1 aufgeführte Hashfunktion eingesetzt wird und die Länge des Schlüssels für beide Verfahren mindestens 16 Byte beträgt:

1. CMAC, siehe [45],
2. HMAC, siehe [9].

Tabelle 5.1: Empfohlene MAC-Verfahren

5.2 Signaturverfahren

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

Für die Zuordnung des öffentlichen Schlüssels zum Beweisenden wird üblicherweise eine Public Key Infrastruktur benötigt.

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 eine Bitlänge fester Länge abbildet, und
3. Ein Algorithmus zum Signieren der gehashten Daten und ein Algorithmus zum Verifizieren der Signatur.

Zusätzlich geben wir Empfehlungen für minimale Schlüssellängen an.

Für die Berechnung des Hashwertes sind grundsätzlich alle der in Tabelle 4.1 aufgelisteten Hashfunktionen geeignet. Wir müssen also in den folgenden vier Unterabschnitten jeweils nur noch die unter Punkt 1. und 3. aufgeführten Algorithmen und Schlüssellängen angeben. Im übrigen können alle empfohlenen Verfahren sowohl zur Signierung von Daten, als auch zum Ausstellen von Zertifikaten genutzt werden.

Tabelle 5.2 gibt einen Überblick über die im Folgenden empfohlenen Signaturverfahren.

1. RSA, siehe [30], 2. DSA, siehe [30] und [23], 3. DSA-Varianten auf elliptischen Kurven: a) ECDSA, siehe [8], b) ECKDSA, ECGDSA, siehe [32], und c) Nyberg-Rueppel-Signaturen, siehe [34]. 4. Merkle-Signaturen, siehe [14].

Tabelle 5.2: Empfohlene Signaturverfahren

Vergleiche auch den sogenannten Algorithmenkatalog zum deutschen Signaturgesetz, siehe [10].

Bemerkung 8 Mit Ausnahme des DS 3 (vergl. Tabelle 5.3) sind die empfohlenen asymmetrischen Signaturverfahren probabilistische Algorithmen[1]. Hier wird also bei jeder Berechnung einer Signatur ein **neuer** Zufallswert benötigt. Anforderungen an diese Zufallswerte werden in den entsprechenden Abschnitten angegeben.

5.2.1 RSA

Die Sicherheit dieses Verfahrens beruht auf der vermuteten Schwierigkeit des Faktorisierungsproblems ganzer Zahlen.

Schlüsselgenerierung

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

1. Wähle zwei Primzahlen p und q zufällig und unabhängig voneinander unter der Nebenbedingung
$$0,1 < |\log_2 p - \log_2 q| < 30.$$
2. Wähle den öffentliche Exponent $e \in \mathbb{N}$ unter den Nebenbedingungen
$$\mathrm{ggT}(e, (p-1)\cdot(q-1)) = 1 \text{ und } 2^{16} + 1 \leq e \leq 2^{1824} - 1.$$
3. Berechne den geheime Exponenten $d \in \mathbb{N}$ in Abhängigkeit von e unter der Nebenbedingung
$$e \cdot d = 1 \bmod \mathrm{kgV}(p-1, q-1).$$

Mit $n = p \cdot q$ (dem sogenannten Modulus) ist dann (n, e) der öffentliche Schlüssel und d der geheime Schlüssel. Weiter müssen natürlich auch die beiden Primzahlen p und q geheim gehalten werden, da sonst jeder aus dem öffentlichen Schlüssel (n, e) wie unter Punkt 3. den geheimen Exponenten berechnen kann.

Bemerkung 9 (*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 [11].

(*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.

Signaturerzeugung und Signaturverifikation Für die Signaturerzeugung bzw. -verifikation siehe [30]. 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 [15]. Die folgenden Verfahren werden empfohlen:

1. EMSA-PSS, siehe [50],
2. Digital Signature Scheme (DS) 2 und 3, siehe [36].

Tabelle 5.3: Empfohlene Formatierungsverfahren für den RSA-Signaturalgorithmus

Schlüssellänge Die Länge des Modulus n sollte mindestens 2048 Bit betragen.

5.2.2 Digital Signature Algorithm (DSA)

Die Sicherheit dieses Verfahrens beruht auf der vermuteten Schwierigkeit des Diskreten Logarithmenproblems in Körpern $\mathbb{F}_p$ mit Primordnung p.

Schlüsselgenerierung

1. Wähle zwei Primzahlen p und q so, dass
$$q \text{ teilt } p - 1$$
gilt.
2. Wähle x in $\mathbb{F}_p$ und berechne $g := x^{(p-1)/q} \bmod p$.
3. Falls $g = 1$, gehe zu 2.
4. Wähle eine Zahl $a \in \{1, \ldots, q-1\}$ und setze $A := g^a$.

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

Signaturerzeugung und Signaturverifikation Für die Signaturerzeugung bzw. -verifikation siehe [30] und [23].

Schlüssellänge Die Länge der Primzahl p sollte mindestens 2048 Bit und die Länge der Primzahl q mindestens 224 Bit betragen.

Bemerkung 10 Das DSA-Verfahren ist ein sogenannter probabilistischer Algorithmus, d.h. das für die Berechnung der Signatur eine Zufallszahl k benötigt wird. Hier ist $k \in \{1, \ldots, q-1\}$ und sollte bezüglich der Gleichverteilung auf $\{1, \ldots, q-1\}$ gewählt werden, da es ansonsten Angriffe gibt, vergleiche [23, Change Notice 1]. Siehe auch Abschnitt B.4 für einen empfohlenen Algorithmus zur Berechnung des Zufallswertes k.

5.2.3 DSA-Varianten basierend auf elliptischen Kurven

Die Sicherheit dieser Verfahren beruht auf der vermuteten Schwierigkeit des Diskreten Logarithmenproblems in elliptischen Kurven.

Schlüsselgenerierung

1. Erzeuge kryptographisch starke EC-Systemparameter (p, a, b, P, q, i), siehe Abschnitt B.3.
2. Wähle d zufällig und gleichverteilt in $\{1, \ldots, q-1\}$.
3. Setze $G := d \cdot P$.

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

Signaturerzeugung und Signaturverifikation Folgende Algorithmen sind grundsätzlich geeignet:

1. ECDSA, siehe [8],
2. ECKDSA, ECGDSA, siehe [32], und
3. Nyberg-Rueppel-Signaturen, siehe [34].

Tabelle 5.4: Empfohlene Signaturverfahren basierend auf elliptischen Kurven

Schlüssellänge Alle in Tabelle 5.4 aufgeführten Signaturverfahren garantieren ein Sicherheitsniveau von mindestens 100 Bit, wenn für die Ordnung q des Basispunktes P gilt $q \geq 2^{224}$.

Bemerkung 11 Wie das DSA-Verfahren sind alle in diesem Abschnitt empfohlenen Signaturverfahren probabilistische Algorithmen. Hier muss ebenfalls ein Zufallswert $k \in \{1, \ldots, q-1\}$ gemäß der Gleichverteilung gewählt werden, da es ansonsten Angriffe gibt, vergleiche [47]. Siehe auch Abschnitt B.4 für einen empfohlenen Algorithmus zur Berechnung des Zufallswertes k.

5.2.4 Merklesignaturen

Im Gegensatz zu den bisher beschriebenen Signaturverfahren beruht die Sicherheit dieses Algorithmus nicht auf der vermuteten Schwierigkeit eines mathematischen Problems, wie zum Beispiel Faktorisierung ganzer Zahlen oder der Berechnung des diskreten Logarithmus, sondern auf der kryptographischen Stärke einer Hashfunktion.

Für eine genaue Beschreibung siehe [14].

6 Instanzauthentisierung

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

Die Authentisierung erfolgt meist gegenseitig und geht mit einer Schlüsseleinigung einher, um die Vertraulichkeit und Integrität einer anschließenden Kommunikation zu gewährleisten, siehe Kapitel 7 für empfohlene Schlüsselaustausch- und Schlüsseleinigungsverfahren und Abschnitt A.2 für empfohlene Protokolle, die beide Verfahren kombinieren.

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

6.1 Symmetrische Verfahren

Für den Nachweis des Beweisenden (B) gegenüber einem Prüfer (P), dass B im Besitz des geheimen symmetrischen Schlüssels ist, sendet P einen Zufallswert r an B. B berechnet dann mittels des gemeinsamen geheimen Schlüssels K eine Prüfsumme von r und schickt diese zurück an P. P prüft dann diese Prüfsumme. Solche Verfahren heißen auch *Challenge-Response-Verfahren*, siehe Tabelle 6.1 für eine schematische Darstellung.

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

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

Die Berechnung und Verifikation der Prüfsumme hängt vom gewählten Verfahren ab. Grundsätzlich können alle in Kapitel 2 empfohlenen Verschlüsselungsverfahren und alle in Abschnitt 5.1 empfohlenen MAC-Verfahren eingesetzt werden. Für empfohlene Bitlängen und Nebenbedingungen an die benutzten Zufallswerte siehe Abschnitt A.2.

6.2 Asymmetrische Verfahren

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

Bemerkung 12 Auch wenn die in Abschnitt 5.2 empfohlenen Signaturverfahren zur Datenauthentisierung auch zur Instanzauthentisierung genutzt werden können, muss bei konkreten Implementierungen darauf geachtet werden, dass die eingesetzten Schlüssel verschieden sind. D.h., dass ein Schlüssel zur Erzeugung von Signaturen nicht zur Instanzauthentisierung eingesetzt werden darf. 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. Um trotzdem ein ausreichendes Sicherheitsniveau zu erreichen, wird die Anzahl der Zugriffsversuche üblicherweise begrenzt.

Folgende Nebenbedingungen werden empfohlen:

1. Die Entropie des Passwortes muss mindestens $\log_2(10^6)$ Bit betragen (z. B. sechs Ziffern),
2. Die Anzahl der Zugriffsversuche muss auf drei beschränkt sein.

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

Für kontaktbehaftete Chipkarten zum Beispiel ist das Verfahren sehr einfach. Das Passwort wird auf dem Pinpad des Kartenlesers eingegeben und ohne zusätzliche kryptographische Absicherung zur Chipkarte übertragen. Um trotzdem einen ausreichenden Schutz zu gewährleisten, muss ein Kartenleser eingesetzt werden, der entsprechend zertifiziert ist.

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 [37].

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 Schlüsseleini-

gungsverfahren durch, so dass im Anschluss eine vertrauliche und authentisierte Kommunikation durchgeführt werden kann.

Bemerkung 13 (*i*) Wie schon bei kontaktbehafteten Komponenten muss auch hier die Anzahl der Versuche beschränkt sein. Empfohlen wird, nach drei erfolglosen Versuchen die Chipkarte zu sperren.

(*ii*) Um Denial-of-Service Attacken zu verhindern, muss ein Mechanismus zum Aufheben der Sperrung vorgesehen sein. Die Entropie des Schlüssels zur Entsperrung sollte mindestens 100 Bit betragen.

7 Schlüsseleinigungsverfahren

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

Nach erfolgreicher Schlüsseleinigung befinden sich beide Parteien im Besitz eines gemeinsamen Geheimnisses. Für empfohlene Verfahren zur Generierung symmetrischer Schlüssel aus diesem Geheimnis siehe Abschnitt B.1.

Es wird empfohlen, nur Schlüsseleinigungsverfahren zu verwenden, in denen beide Kommunikationspartner Anteile für den neuen Schlüssel bereitstellen.

7.1 Symmetrische Verfahren

Grundsätzlich können alle der in Kapitel 2 empfohlenen symmetrischen Verschlüsselungsverfahren zum Aushandeln neuer Sitzungsschlüssel verwendet werden.

7.2 Asymmetrische Verfahren

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

Zusätzlich zu diesen werden die folgenden beiden Verfahren empfohlen:

1. Diffie-Hellman, siehe [42],
2. EC Diffie-Hellman, siehe [42].

Tabelle 7.1: Empfohlene asymmetrische Schlüsseleinigungsverfahren

Für die Spezifizierung sind folgende Algorithmen festzulegen:

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

7.2.1 Diffie-Hellman

Die Sicherheit dieser Verfahren beruht auf der vermuteten Schwierigkeit des Diskreten Logarithmenproblems in Gruppen $\mathbb{F}_p$, p eine Primzahl.

Systemparameter

1. Wähle eine Primzahl p.
2. Wähle einen Erzeuger g der multiplikativen Gruppe $\mathbb{F}_p^*$.

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

Schlüsselvereinbarung

1. A wählt gleichverteilt einen Zufallswert $x \in \{1, \ldots, p-1\}$ und sendet g^x an B.
2. B wählt gleichverteilt einen Zufallswert $y \in \{1, \ldots, p-1\}$ und sendet g^y an A.
3. A berechnet $(g^y)^x = g^{xy}$.
4. B berechnet $(g^x)^y = g^{xy}$.

Das ausgehandelte Geheimnis ist dann g^{xy}.

Schlüssellänge Die Länge von p sollte mindestens 2048 Bit betragen.

7.2.2 EC Diffie-Hellman

Die Sicherheit dieser Verfahren beruht auf der vermuteten Schwierigkeit des Diskreten Logarithmenproblems in elliptischen Kurven.

Systemparameter

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

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 $x \cdot P$ an B.
2. B wählt gleichverteilt einen Zufallswert $y \in \{1, \ldots, q-1\}$ und sendet $y \cdot P$ an A.
3. A berechnet $x \cdot (y \cdot P) = xy \cdot P$.
4. B berechnet $y \cdot (x \cdot P) = xy \cdot P$.

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

Schlüssellänge Die Länge von q sollte mindestens 224 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 kompromitiert wird.

Wir geben deshalb in diesem Kapitel ein Verfahren an, welches erlaubt, ein Geheimnis, wie zum Beispiel einen kryptographischen Schlüssel K, so in n Teilgeheimnisse $K_1, \ldots, K_n$ aufzuteilen, dass beliebige $t \leq n$ dieser Teilgeheimnisse genügen, um das Geheimnis zu rekonstruieren, $t-1$ Teilgeheimnisse aber keine Information über K liefern.

Eine weitere Anwendung dieses Verfahrens ist ein Vieraugenprinzip oder allgemeiner ein t-aus-n-Augenprinzip zu gewährleisten, um zum Beispiel das Passwort für eine kryptographische Komponente so auf n verschiedene Anwender zu verteilen, dass mindestens t Anwender benötigt werden, um das Passwort zu rekonstruieren.

Das hier vorgestellte Secret-Sharing-Verfahren wurde von A. Shamir entwickelt, siehe [52]. Wir nehmen im Folgenden an, dass das zu verteilende Geheimnis ein Schlüssel K der Bitlänge r ist: $K = (k_0, \ldots, k_{r-1}) \in \{0,1\}^r$.

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

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

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

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

Bemerkung 14 Die Koeffizienten $a_0, \ldots, a_{t-1}$ eines unbekannten Polynoms f vom Grad $t-1$ können mit Hilfe der sogenannten *Lagrange-Interpolations-Formel* aus t Punkten $(x_i, f(x_i))$ wie folgt gefunden werden:

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

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

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

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

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

Bemerkung 15 Die Wahl der Primzahl p unter der Bedingung $p \geq \max(2^r, n+1)$ garantiert, dass dieses Verfahren mindestens ein Sicherheitsniveau der Bitlänge des zu schützendes Geheimnisses hat. Zusätzliche Schlüssellängen müssen also nicht angegeben werden.

9 Zufallszahlengeneratoren

Zufallszahlen werden für eine Reihe von Anwendungen in kryptographischen Verfahren benötigt, wie zum Beispiel für kryptographische Schlüssel, Systemparameter für Signatur-, Authentisierungs- und asymmetrischen Verschlüsselungsverfahren, Instanzauthentisierung, probabilistische Signatur- bzw. Authentisierungsverfahren und Paddingverfahren.

In Anhang B geben wir für eine Vielzahl von Anwendungen Algorithmen an, wie aus Zufallszahlen die gewünschten Werte berechnet werden können.

Dabei spielt die Unvorhersagbarkeit, auch Entropie genannt, die entscheidende Rolle. Die Entropie gibt, einfach gesprochen, an, wie viele Versuche ein Angreifer mindestens benötigt, um alle Möglichkeiten durchzuprobieren. Wir messen im Folgenden die Entropie in Bit, d.h. zum Beispiel, besitzt ein symmetrischer Schlüssel eine Entropie von n Bit, so benötigt ein Angreifer mindestens 2^n Schritte, um den gesamten Schlüsselraum zu durchsuchen.

Es wird dringend empfohlen, einen physikalischen Zufallszahlengenerator zur Erzeugung von Zufallswerten für die in dieser Technischen Richtlinie empfohlenen Verfahren zu benutzen. Zumindest sollte der Startwert (bzw. *Seed*, siehe Abschnitt 9.2), für einen eingesetzten deterministischen Zufallszahlengenerator mit Hilfe eines physikalischen Zufallszahlengenerators erzeugt und der innere Zustand des deterministischen regelmäßig mit Zufallsfolgen des physikalischen Zufallszahlengenerators erneuert werden.

9.1 Physikalische Zufallszahlengeneratoren

Physikalische Zufallszahlengeneratoren beziehen Zufallszahlen aus physikalischen Rauschquellen, die beispielsweise auf elektromagnetischen, elektromechanischen oder quantenmechanischen Effekten beruhen. Häufig ist eine deterministische Nachbearbeitung der digitalen Rauschsignale nötig, um eventuell auftretende Schiefen in der Ausgabe des physikalischen Zufallszahlengenerators nicht in die für kryptographische Verfahren verwendeten Zufallswerte einfließen zu lassen.

Es wird empfohlen, einen P2-Generator mit Stärke der Mechanismen bzw. Funktionen HOCH im Sinne der AIS 31, siehe [2], einzusetzen. Dies bedeutet unter anderem:

1. Die Eigenschaften der digitalisierten Rauschsignale müssen sich hinreichend gut durch ein stochastisches Modell beschreiben lassen.

2. Der durchschnittliche Entropiezuwachs pro Zufallsbit liegt oberhalb einer gegebenen Mindestschranke.

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

4. Auf eine fehlerhafte Rauschsignalfolge muss zum Beispiel durch Stilllegen der Rauschquelle reagiert werden.

Die Entwicklung und sicherheitskritische Bewertung von physikalischen Zufallszahlengeneratoren setzt eine umfassende Erfahrung auf diesem Gebiet voraus. **Es wird empfohlen, sich bei Bedarf in diesem Zusammenhang frühzeitig an entsprechende Experten auf diesem Gebiet zu wenden.**

9.2 Deterministische Zufallszahlengeneratoren

Deterministische bzw. Pseudozufallszahlengeneratoren berechnen aus einem Zufallswert fester Länge, dem sogenannten *Seed*, eine pseudozufällige Bitfolge praktisch beliebiger Länge. Dazu wird der sogenannte innere Zustand des Pseudozufallszahlengenerators zunächst mit dem Seed initialisiert. In jedem Schritt wird dann der innere Zustand erneuert und hieraus eine Zufallszahl (meist eine Bitfolge fester Länge) abgeleitet. Der innere Zustand muss natürlich gegen Auslesen und Manipulation entsprechend geschützt werden.

Es wird empfohlen, einen K4-Generator mit Stärke der Mechanismen bzw. Funktionen HOCH im Sinne der AIS 20, siehe [1], einzusetzen. Dies bedeutet unter anderem:

1. Es ist einem Angreifer praktisch unmöglich, zu einer bekannten Zufallszahlenfolge Vorgänger oder Nachfolger dieser Teilfolge oder einen inneren Zustand zu berechnen.

2. Es ist einem Angreifer praktisch unmöglich, aus Kenntnis eines inneren Zustandes Vorgänger der Teilfolge oder Vorgängerzustände zu berechnen.

Zusätzlich muss die Entropie des Seed mindestens 100 Bit betragen.

Beispiele für Pseudozufallszahlengeneratoren finden sich ebenfalls in [1].

9.3 Seedgenerierung

Wie bereits im letzten Abschnitt beschrieben, wird für die Initialisierung eines Pseudozufallszahlengenerators ein Seed mit ausreichend hoher Entropie (hier mindestens 100 Bit) benötigt.

Die Erzeugung dieses Seeds sollte mit physikalischen Zufallszahlengeneratoren erfolgen. In vielen Anwendungen steht dieser aber nicht zur Verfügung. Typische Szenarien sind der Einsatz kryptographischer Verfahren für E-Government- und E-Business- Anwendungen. Hier werden für viele Verfahren, wie zum Beispiel Verschlüsselung, Schlüsseleinigung oder Challenge-Response-Protokolle, Zufallszahlen mit hoher Entropie benötigt. Allerdings werden diese Anwendungen häufig auf Computern ohne zusätzliche kryptographische Hardware genutzt. Wir geben aus diesem Grund für die zwei wichtigsten Betriebssysteme geeignete Verfahren zur Seedgenerierung an.

Der Einsatz der in den folgenden beiden 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. Dies schließt also die Existenz von z.B. Viren oder Trojanern auf diesem Rechner aus. Insbesondere müssen Benutzerinnen und Benutzer über die Risiken aufgeklärt werden.

9.3.1 GNU/Linux

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

/dev/random (in der Kernelversion 2.6.21.5)

Tabelle 9.1: Empfohlenes Verfahren zur Seedgenerierung unter GNU/Linux

Bemerkung 16 Die Funktion /dev/random wurde bisher nur in der Kernelversion 2.6.21.5 vom BSI untersucht. Solange keine gravierenden Änderungen bei folgenden Kernelversionen in dieser Funktion vorgenommen werden, sollte die Funktion /dev/random auch für aktuellere Kernel das in dieser Technischen Richtlinie geforderte Sicherheitsniveau von 100 Bit erreichen.

9.3.2 Windows

Im Gegensatz zum System GNU/Linux gibt es derzeit keine vom BSI untersuchte Funktion im System Windows, das ein ausreichendes Sicherheitsniveau von 100 Bit gewährleistet. Zur Erzeugung sicherer Seeds sollten daher verschiedene Systemaufrufe in geeigneter Weise kombiniert werden. Dazu eignen sich die beiden Funktionen

1. ReadTimeStampCounter():
   gibt die seit Systemstart durchlaufenden Prozessorzyklen an, bei einem Prozessor mit einer Taktfrequenz von mindestens 1 GHz gibt es pro Sekunde also mindestens 2^{30} verschiedene Werte.

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

Kombiniert man diese beiden Funktionen in geeigneter Weise so, dass die Zeitpunkte, zu denen die Funktionen aufgerufen werden, paarweise voneinander weitestgehend unabhängig sind, so lässt sich ein Seed mit ausreichender Entropie erzeugen. Beispielhaft sei folgendes Verfahren angegeben. Dabei muss gewährleistet sein, dass sämtliche Schritte (inklusive das Starten des Rechners) von einem Benutzer ausgeführt werden und nicht automatisiert sind.

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

Geht man davon aus, dass ein Angreifer die obigen Zeitpunkte nur auf höchstens eine Sekunde genau schätzen kann, so hat der Wert SEED eine Entropie von mindestens 113 Bit.

Anhang A

Anwendung kryptographischer Verfahren

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

Weiter muss eine Schlüsseleinigung immer mit einer Instanzauthentisierung einher gehen, damit sich beide Parteien sicher sind, mit wem sie kommunizieren. Andernfalls kann durch eine sogenannte Man-in-the-Middle-Attacke die Kommunikation kompromitiert werden. Für beide Anwendungen geben wir in diesem Kapitel geeignete Verfahren an.

A.1 Verschlüsselungsverfahren mit Datenauthentisierung (Secure Messaging)

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

Allerdings müssen die folgenden beiden Nebenbedingungen eingehalten werden:

1. Authentisiert werden ausschließlich die verschlüsselten Daten.
2. Verschlüsselungs- und Authentisierungsschlüssel müssen verschieden und sollen nicht voneinander ableitbar sein.

Bemerkung 17 Es besteht die Möglichkeit, Verschlüsselungs- und Authentisierungsschlüssel aus einem gemeinsamen Schlüssel abzuleiten. Empfohlene Verfahren sind in Abschnitt B.1 zusammengefasst.

A.2 Schlüsselvereinbarung mit Instanzauthentisierung

Wie bereits in der Einleitung zu diesem Kapitel erwähnt, muss eine Schlüsselvereinbarung immer mit einer Instanzauthentisierung kombiniert werden. Wir geben in den folgenden beiden Unterabschnitten sowohl Verfahren an, die sich auf rein symmetrische Algorithmen, als auch auf rein asymmetrische Algorithmen stützen.

Nach dem erfolgreichen Ablauf der vorgestellten Protokolle befinden sich beide Kommunikationspartner im Besitz eines gemeinsamen Geheimnisses. Für die Berechnung symmetrischer Schlüssel für Verschlüsselungs- und Datenauthentisierungsverfahren aus diesem Geheimnis siehe Abschnitt B.1.

Bemerkung 18 Bei der konkreten Umsetzung der vorgestellten Verfahren müssen die folgenden beiden Bedingungen erfüllt werden.

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

2. Die für die Schlüsselvereinbarung benutzten Zufallswerte müssen mindestens eine Entropie erreichen, die den gewünschten Schlüssellängen der auszuhandelnden Schlüssel entspricht[1].

A.2.1 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:

Schlüsselvereinbarung mit Instanzauthentisierung gemäß [21], Abschnitt 8.7.

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

Bemerkung 19 In [21] wird als Blockchiffrierverfahren TDES und ein auf DES basierendes MAC-Verfahren empfohlen. Um konform mit der vorliegenden Technischen Richtlinie zu sein, muss allerdings als Blockchiffrierverfahren eines der in Kapitel 2 empfohlenen Verfahren (inklusive empfohlener Betriebsarten, Paddingverfahren und Verfahren zur Erzeugung von Initialisierungsvektoren) und als MAC eines der in Kapitel 5 empfohlenen Verfahren verwendet werden.

A.2.2 Asymmetrische Verfahren

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

Um allerdings Fehler in selbst konstruierten Protokollen auszuschließen, werden die in Tabelle A.2 aufgelisteten Verfahren zur Schlüsselvereinbarung mit Instanzauthentisierung basierend auf asymmetrischen Verfahren empfohlen.

Bemerkung 20 In einigen der in Tabelle A.2 empfohlenen Standards werden Signaturverfahren und asymmetrische Schlüsselaustauschverfahren vorgeschlagen, die in diesem Dokument nicht empfohlen werden. Dies betrifft vor allem Hashfunktionen und Paddingverfahren. 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 Verfahren zur Anwendung kommen.

Bemerkung 21 Bei dem Verfahren EC-KAEG findet keine gegenseitige Authentisierung statt, hier beweist nur eine Partei der anderen im Besitz eines privaten Schlüssels zu sein.

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

1. Elliptic Curve Key Agreement of ElGamal Type (EC-KAEG), siehe [8].
2. Instanzauthentisierung mit RSA und Schlüsselvereinbarung mit RSA, siehe [19], Annex D, A.14,
3. Instanzauthentisierung mit DSA und Schlüsselvereinbarung mit Diffie-Hellman, siehe [19], Annex D, A.15,
4. Instanzauthentisierung mit ECDSA und Schlüsselvereinbarung mit EC-Diffie-Hellman, siehe [19], Annex D, A.16,
5. Instanzauthentisierung und Schlüsselvereinbarung mit Diffie-Hellman, siehe [21], Anhang A.3,
6. MTI A(0), siehe [29], Key agreement mechanism 5, genauer Annex B.6.

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

Anhang B

Zusätzliche Funktionen und Algorithmen

Für einige der in dieser Technischen Richtlinie empfohlenen kryptographischen Verfahren werden zusätzliche Funktionen und Algorithmen benötigt, um zum Beispiel Systemparameter zu generieren, oder aus den Ergebnissen von Zufallszahlengeneratoren oder Schlüsseleinigungsverfahren symmetrische Schlüssel zu erzeugen. Diese Funktionen und Algorithmen sind sorgfältig zu wählen, um das in dieser Technischen Richtlinie geforderte Sicherheitsniveau zu erreichen und kryptoanalytische Angriffe zu verhindern.

B.1 Schlüsselableitung

Nach einem Schlüsseleinigungsverfahren sind beide Parteien im Besitz eines gemeinsamen Geheimnisses. Häufig müssen aus diesem Geheimnis mehrere symmetrische Schlüssel, zum Beispiel für die Verschlüsselung und zur Datenauthentisierung, abgeleitet werden.

Folgendes Verfahren zur Schlüsselableitung wird empfohlen:

Key Derivation Function aus [6], Abschnitt 5.6.3.

Tabelle B.1: Empfohlenes Verfahren zur Schlüsselableitung

Bemerkung 22 (*i*) In [6], Abschnitt 5.6.3 wird die Hashfunktion SHA1 zur Ableitung der symmetrischen Schlüssel genutzt. Obwohl diese Hashfunktion in Kapitel 4 nicht empfohlen wird, kann sie für dieses Verfahren eingesetzt werden.

(*ii*) Anstelle der Hashfunktion SHA1 kann auch RIPEMD160 und alle in Kapitel 4 empfohlenen Hashfunktionen eingesetzt werden.

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 zukünftig eingesetzte Initialisierungsvektoren nicht kennen darf.

Zwei Verfahren werden zur Erzeugung unvorhersagbarer Initialsisierungsvektoren 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 100 Bit betragen.
2. **Verschlüsselte Initialisierungsvektoren:** Nutze ein deterministisches Verfahren zur Erzeugung sogenannter Pre-Initialisierungsvektoren (z.B. einen Zähler). Verschlüssele den Pre-Initialisierungsvektor mit der einzusetzenden Blockchiffre und dem einzusetzenden Schlüssel und nutze den Chiffretext als Initialisierungsvektor.

Tabelle B.2: Empfohlene Verfahren zur Erzeugung unvorhersagbarer Initialsisierungsvektoren

B.3 Erzeugung von EC-Systemparametern

Die Sicherheit asymmetrischer Verfahren, die in Gruppen arbeiten, die von elliptischen Kurven erzeugt werden, beruht auf der vermuteten Schwierigkeit des Diskreten Logarithmenproblems in diesen Gruppen.

Zur Festlegung der EC-Systemparameter werden benötigt:

1. Eine Primzahl p,
2. Kurvenparameter $a, b \in \mathbb{F}_p$ mit $4a^3 + 27b^2 \neq 0$, die eine elliptische Kurve
$$E = \{(x, y) \in \mathbb{F}_p \times \mathbb{F}_p;\ y^2 = x^3 + ax + b\} \cup \{\mathcal{O}_E\}$$
festlegen, und
3. ein Basispunkt P auf $E(\mathbb{F}_p)$.

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

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

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

Siehe [20] für eine Erläuterung.

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

Bemerkung 23 Es wird empfohlen, die unter Punkt 1. zu generierende Systemparameter nicht selbst zu erzeugen, sondern stattdessen auf standardisierte Werte zurückzugreifen, die von einer vertrauenswürdigen Instanz zur Verfügung gestellt werden.

Die in Tabelle B.3 aufgelisteten Systemparameter werden empfohlen:

1. brainpoolP224t1, siehe [20],
2. brainpoolP256t1, siehe [20],
3. brainpoolP320t1, siehe [20],
4. brainpoolP384t1, siehe [20],
5. brainpoolP512t1, siehe [20].

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

B.4 Generierung von Zufallszahlen für probabilistische asymmetrische Verfahren

Bei den in dieser Technischen Richtlinie vorgestellten asymmetrischen Verfahren wird häufig ein Zufallswert $k \in \{1, \ldots, q-1\}$ benötigt, wobei q die Ordnung der jeweiligen Gruppe ist, in der die Berechnung stattfindet. Wie bereits in Bemerkungen 4, 5, 10 und 11 erwähnt, sollte k möglichst gleichverteilt gewählt werden.

Auf der anderen Seite liefern die in Kapitel 9 beschriebenen Zufallszahlengeneratoren gemäß der Gleichverteilung nur Zufallswerte aus $\{0, 1, \ldots, 2^n - 1\}$, $2^n - 1 \geq q$. Bei einer konkreten Implementierung muss darauf geachtet werden, dass sich die Gleichverteilung auf $\{0, 1, \ldots, 2^n - 1\}$ in $\{1, \ldots, q\}$ wiederfindet.

Wir empfehlen im Folgenden drei Verfahren. Sei dazu $n \in \mathbb{N}$ so, dass $2^{n-1} \leq q < 2^n - 1$ (q habe also eine Bitlänge von n).

Verfahren 1.

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

Verfahren 2.

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

Verfahren 3.

1. Wähle $x, y \in \{0, 1, \ldots, 2^n - 1\}$ gleichverteilt.
2. `return` $k = x \cdot y \bmod q$.

Tabelle B.4: Berechnung von Zufallswerten

Bemerkung 24 (i) Verfahren 2. und 3. liefern keine Gleichverteilung auf $\{0, \ldots, q-1\}$, die

Abweichung ist allerdings so gering, dass dies nach derzeitigem Wissensstand von einem Angreifer nicht ausgenutzt werden kann.

(*ii*) Im Gegensatz dazu liefert Verfahren 1. eine Gleichverteilung, hat allerdings den Nachteil, dass unter Umständen mehrere Iterationen notwendig sind. Um zu garantieren, dass das Verfahren nach einer bestimmten Zeit terminiert, wird empfohlen, die Anzahl der Iterationen zu beschränken. Da die Wahrscheinlichkeit dafür, dass ein zufälliger Wert $k \in \{0, 1, \ldots, 2^n - 1\}$ durch 2^{n-1} (und damit auch durch q) von oben beschränkt ist (d.h. $k < 2^{n-1} \leq q$ gilt), gerade $2^{n-1}/2^n = 1/2$ ist, genügen in der Praxis 8 Iterationen. Die Wahrscheinlichkeit, dass das Verfahren tatsächlich einen Wert kleiner q zurückgibt, ist dann ungefähr $1 - 1/2^8$.

B.5 Probabilistische Primzahltests

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

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

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

Tabelle B.5: Empfohlenes Verfahren zur Erzeugung von Primzahlen

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

Miller-Rabin, siehe [42], Algorithmus 4.24.

Tabelle B.6: Empfohlener probabilistischer Primzahltest

Bemerkung 25 (*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 voneinaner bezüglich der Gleichverteilung gewählten Basen $a_1, \ldots, a_{50} \in \{2, 3, \ldots, p-2\}$ aufgerufen werden, siehe Abschnitt B.4 für empfohlene Verfahren zur Berechnung gleichverteilter Zufallszahlen aus $\{2, 3, \ldots, p-2\}$.

(*iii*) (Average Case) Um eine bezüglich der Gleichverteilung gewählte ungerade Zahl $p \in [2^{b-1}, 2^b - 1]$ auf ihre Primzahleigenschaft zu testen, reichen bei weitem weniger Iterationen des Miller-Rabin-Algorithmus aus, vergleiche [16], [44] und [35], Annex A. So sind für $b = 1024$ lediglich 4 Iterationen notwendig, um mit Wahrscheinlichkeit 2^{-109} auszuschließen, dass p zusammengesetzt ist, obwohl der Miller-Rabin-Algorithmus p als Primzahl erkennt. Auch hier

müssen die Basen unabhängig und bezüglich der Gleichverteilung aus $\{2, 3, \ldots, p-2\}$ gewählt werden.

Literaturverzeichnis

[1] AIS 20 (W. Schindler). *Functionality classes and evaluation methodology for deterministic random number generators.* Bundesamt für Sicherheit in der Informationstechnik, 1999. Verfügbar unter http://www.bsi.de/zertifiz/zert/interpr/aisitsec.htm.

[2] AIS 31 (W. Schindler). *Functionality classes and evaluation methodology for physical random number generators.* Bundesamt für Sicherheit in der Informationstechnik, 2001. Verfügbar unter http://www.bsi.de/zertifiz/zert/interpr/aisitsec.htm.

[3] R. Anderson, E. Biham, and L. Knudsen. Serpent. Verfügbar unter http://www.cl.cam.ac.uk/~rja14/serpent.html.

[4] ANSI X9.62. *Triple Data Encryption Algorithm, Modes of Operation*, 1998.

[5] ANSI X9.62. *Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA)*, 2005.

[6] ANSI X9.63. *Public Key Cryptography for the Financial Services Industry: Key Agreement and Key Transport Using Elliptic Curve Cryptography*, 2001.

[7] R. M. Avanzi, H. Cohen, C. Doche, G. Frey, T. Lange, K. Nguyen, and F. Vercauteren. *Handbook of Elliptic and Hyperelliptic Curve Cryptography.* Chapman & Hall/CRC, 2005.

[8] BSI (H. Baier, D. Kügler und M. Margraf). *Elliptic Curve Cryptography based on ISO 15946.* BSI Technical Guideline TR-03111, 2007.

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

[10] BNetzA, Bekanntmachung zur elektronischen Signatur nach dem Signaturgesetz und der Signaturverordnung (Übersicht über geeignete Algorithmen).

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

[12] S. Bradner. *Key words for use in RFCs to indicate requirement levels (RFC2119)*, 1999. Verfügbar unter http://www.ietf.org/rfc/rfc2119.txt.

[13] BSI: *IT-Grundschutzkataloge,* 2006.

[14] J. Buchmann, L.C. Coronado García, E. Dahmen, M. Döring und E. Klintsevich. *CMSS – An Improved Merkle Signature Scheme.* Progress in Cryptography – INDOCRYPT 2006, Springer LNCS 4329, pp. 349-363 (2006).

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

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

[17] M. Daum und S. Lucks. *The Story of Alice and Bob,* "Rump Session"-Vortrag Eurocrypt 2005, `http://www.cits.rub.de/imperia/md/content/magnus/rump_ec05.pdf`.

[18] DIN V66291. *Spezifikation der Schnittstelle zu Chipkarten mit Digitaler Signatur-Anwendung/Funktion nach SigG und SigV,* Annex A, 2.1.1, 1999.

[19] DIN V66291-4. *Chipkarten mit Digitaler Signatur-Anwendung/Funktion nach SigG und SigV, Teil 4: Grundlegende Sicherheitsdienste,* DIN NI-17.4, 2002.

[20] ECC Brainpool. *ECC Brainpool Standard Curves and Curve Generation*, 2005. Verfügbar unter `http://www.ecc-brainpool.org/ecc-standard.htm`, Version 1.0.

[21] European Committee for Standardization. *Application Interface for smart cards used as Secure Signature Creation Devices - Part 1: Basic requirements* 08.03.2004.

[22] Federal Information Processing Standards Publication 180-2 (FIPS PUB 180-2). *Secure Hash Standard*, 2002.

[23] Federal Information Processing Standards Publication 186-2 (FIPS PUB 186-2). *Digital Signature Standard (DSS)*, 2000.

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

[25] M. Gebhardt, G. Illies und W. Schindler. *A Note on the Practical Value of Single Hash Collisions for Special File Formats,* Sicherheit 2006, Köllen-Verlag, LNI P-77 (2006), 333-344. Erweiterte Version: NIST Cryptographic Hash Workshop 2005, `http://www.csrc.nist.gov/pki/Hashworkshop/2005/Oct31_Presentations/Illies_NIST_05.pdf`.

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

[27] ISO/IEC 10118-3-2004. *Information technology – Security techniques – Hash functions – Part 3: Dedicated hash functions*, 2003.

[28] ISO/IEC 11770-3-1999. *Information technology – Security techniques – Key management – Part 2: Mechanisms using symmetric techniques*, 1996.

[29] ISO/IEC 11770-3-1999. *Information technology – Security techniques – Key management – Part 3: Mechanisms using asymmetric techniques*, 1999.

[30] ISO/IEC 14888-3-1999. *Information technology – Security techniques – Digital signatures with appendix – Part 3: Certificate-based mechanisms*, 1999.

[31] ISO/IEC 15946-1-2002. *Information technology – Security techniques – Cryptographic techniques based on elliptic curves – Part 1: General*, 2002.

[32] ISO/IEC 15946-2-2002. *Information technology – Security techniques – Cryptographic techniques based on elliptic curves – Part 2: Digital signatures*, 2002.

[33] ISO/IEC 15946-3-2002. *Information technology – Security techniques – Cryptographic techniques based on elliptic curves – Part 3: Key establishment*, 2002.

[34] ISO/IEC 15946-4-2004. *Information technology – Security techniques – Cryptographic techniques based on elliptic curves – Part 4: Digital signatures giving message recovery*, 2004.

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

[36] ISO/IEC 9796-2-2002. *Information technology – Security techniques – Digital Signature Schemes giving message recovery – Part : Integer Factorization based mechanisms*, 2002.

[37] D. Kügler und M. Margraf. *PACE: Password Authenticated Connection Establishment.* preprint (2007).

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

[39] ISO/IEC 7816-8-2004. *Identification cards – Integrated circuit cards – Part 8: Commands for security operations*, 2004.

[40] ISO/IEC 7816-4-2005. *Identification cards – Integrated circuit cards – Part 4: Organization, security and commands for interchange*, 2005.

[41] A. Menezes. *Elliptic Curve Public Key Cryptosystems.* Kluwer Academic Publishers, 1993.

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

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

[44] NIST. Two Methods to Calculate the Required Number of Rounds of Miller-Rabin Testing, Januar 2007. `csrc.nist.gov/groups/ST/toolkit/documents/dss/Miller-RabinTrails(12Jan07).doc`

[45] NIST. Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication, Special Publication SP800-38B, National Institute of Standards and Technology, Technology Administration, U.S. Department of Commerce, 2001.

[46] NIST. Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) for Confidentiality and Authentication, Special Publication SP800-38D, National Institute of Standards and Technology, Technology Administration, U.S. Department of Commerce, 04.2006.

[47] P. Nguyen and I. Shparlinski. The insecurity of the elliptic curve signature algorithm with partially known nonces. *Designs, Codes and Cryptography*, 30(2):201–217, 2003.

[48] R. Housley. Cryptographic Message Syntax (CMS). *RFC 3852,* 2004.

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

[50] PKCS #1 v2.1: RSA Cryptographic Standard, 14.06.2002.

[51] B. Schneier, J. Kelsey, D. Whiting, D. Wagner, C. Hall, and N. Ferguson. Twofish: A 128-Bit Block Cipher. Verfügbar unter `http://www.schneier.com/paper-twofish-paper.html.`

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

[53] X. Wang, Y.L. Yin, and H. Yu. Collision search attacks on SHA-1. In *Proceedings of Crypto 2005.* Springer Verlag, 2005.