Veröffentlicht im Bundesanzeiger Nr. 31 - Seite 1787 bis 1788 v. 14. Februar 1998

Gemäß § 17 Abs. 2 Satz 1 SigV veröffentlicht die Regulierungsbehörde für Telekommunikation und Post in der Anlage eine Übersicht über die Algorithmen und dazugehörige Parameter, die zur Erzeugung von Signaturschlüsseln, zum Hashen zu signierender Daten oder zur Erzeugung und Prüfung digitaler Signaturen als geeignet anzusehen sind, sowie den Zeitpunkt, bis zu dem die Eignung jeweils gilt.

Geeignete Kryptoalgorithmen gemäß § 17 Abs. 2 SigV

1.1. Kryptoalgorithmen

Die Sicherheit einer digitalen Signatur hängt primär von der Stärke der zugrundeliegenden Kryptoalgorithmen ab. Im folgenden werden Kryptoalgorithmen genannt, die für digitale Signaturen (mindestens) für die **kommenden sechs Jahre** als geeignet anzusehen sind. (Die bit-genauen Spezifikationen findet man in den entsprechenden Standards verschiedener Organisationen (ISO/IEC, NIST, IEEE usw.), sie sind nicht Gegenstand dieses Dokuments.) Es wird sich hierbei auf die wichtigsten praxisrelevanten Algorithmen beschränkt, deren kryptographische Eigenschaften aufgrund der heute vorliegenden Ergebnisse langjähriger Diskussionen und Analysen am besten eingeschätzt werden können.
Die Zertifizierungsstellen können abweichend von den hier vorgeschlagenen Algorithmen auch andere Verfahren einsetzen, wenn nach Angaben des Bundesamtes für Sicherheit in der Informationstechnik (BSI) deren Eignung von der zuständigen Behörde festgestellt wird.
Die Liste der hier genannten Algorithmen ist als offen und vorläufig anzusehen. Sie wird gemäß der weiteren Entwicklung der kryptologischen Forschung und den Erfahrungen mit praktischen Realisierungen von Signaturverfahren aktualisiert und gegebenenfalls ergänzt werden.

1.2. Kryptographische Anforderungen

Ein Schema zur digitalen Signatur im Sinne des Gesetzes umfaßt die folgenden Kryptoalgorithmen:

- einen schnellen Algorithmus zum Hashen von Daten (einer Hashfunktion), der die zu signierenden Daten auf einen Hashwert, d. h. eine Bitfolge fester kurzer Länge reduziert. Signiert werden dann nicht die Daten selbst, sondern stattdessen jeweils ihr Hashwert,
- einen Signaturalgorithmus, der aus einem Signier- und einem Verifizieralgorithmus besteht. Der Signaturalgorithmus hängt ab von einem Schlüsselpaar, bestehend aus einem privaten (d. h. geheimen) Schlüssel zum Signieren (Erzeugen einer Signatur) und dem dazugehörigen öffentlichen Schlüssel zum Verifizieren (Prüfen) der Signatur, und
- ein Verfahren zur Erzeugung von Schlüsselpaaren für die einzelnen Teilnehmer.

Ein solches Schema ist sicher, wenn nur der Inhaber des privaten Schlüssels in der Lage ist, eine Signatur zu erzeugen, die vom Verifizieralgorithmus als gültig erkannt wird (dies bezieht sich hier nur auf die abstrakt-mathematischen Eigenschaften des Schemas. Sicherheitsprobleme, die bei der konkreten technischen Implementierung auftreten, werden hier ausgeklammert). Welche Anforderungen sich hieraus an die oben genannten kryptographischen Algorithmen ergeben, soll im folgenden kurz beschrieben werden:

1.2.1. Hashfunktionen

Beim Signieren wird der Hashwert der zu signierenden Daten gewissermaßen wie ein 'digitaler Fingerabdruck' benutzt. Damit hierbei keine Sicherheitslücke entsteht, muß die Hashfunktion *H* folgenden Kriterien genügen:

- *H* muß eine *Einwegfunktion* sein; d. h., es ist praktisch unmöglich, zu einem gegebenen Bitstring aus dem Wertebereich ein Urbild bzgl. *H* zu finden, und

- *H* muß *kollisionsresistent* sein; d. h., es ist praktisch unmöglich, Kollisionen zu finden (Zwei verschiedene digitale Dokumente, die auf denselben Hashwert abgebildet werden, bilden eine Kollision).

Die Existenz von Kollisionen und - wenn man ein pseudo-zufälliges Verhalten der Hashfunktion unterstellt und damit die Surjektivität - die Existenz von Urbildern ist unvermeidbar. Dies ist aber nur eine theoretische Aussage. Bei der praktischen Anwendung kommt es nur darauf an, daß es, wie oben verlangt, unmöglich ist, Kollisionen bzw. Urbilder zu *finden*.

1.2.2. Signaturalgorithmen

Niemand anderes als der Besitzer des Signierschlüssels darf in der Lage sein, Signaturen zu erzeugen. Insbesondere bedeutet dies, daß es praktisch unmöglich ist, den Signierschlüssel aus dem (öffentlichen) Verifizierschlüssel zu berechnen.

1.2.3 Schlüsselerzeugung

Die verschiedenen Signaturalgorithmen benötigen Schlüssel, die bestimmte Bedingungen erfüllen. In einigen Fällen kommen weitere Bedingungen hinzu, deren Nichtbeachtung zu Schwächen des jeweiligen Verfahrens führen könnte. Im Rahmen dieser Anforderungen müssen die Schlüssel zufällig erzeugt werden (siehe Abschnitt 1.5).

1.3. Vorschläge für geeignete Hashfunktionen

Die Hashfunktion MD4 wurde 1990 von Ron Rivest eingeführt ([9], [6]). Das Design ist sehr stark auf eine gute Performance ausgerichtet und speziell zugeschnitten auf die heute sehr verbreiteten 32-Bit-Prozessoren. MD4 hat drei interne Runden. Auf der Basis der Designprinzipien von MD4 sind später eine Reihe weiterer komplexerer Hashfunktionen (mit größerer Rundenzahl) vorgeschlagen worden. Die Hashfunktionen dieser MD4-Familie sind in den letzten Jahren sehr intensiv untersucht worden. Während sich MD4, im wesentlichen wegen der noch zu geringen Rundenzahl, als nicht kollisionsresistent erwies, kann man nach heutigem Kenntnisstand der Analyse von der langfristigen Sicherheit folgender beider Hashfunktionen der MD4-Familie ausgehen:

- RIPEMD-160 ([7], [3]),
- SHA-1 ([2], [3]).

Diese beiden Hashfunktionen sind (mindestens) in den **kommenden sechs Jahren**, d. h. bis Ende 2003, für die Anwendung bei digitalen Signaturen geeignet.

1.4. Vorschläge für geeignete Signaturalgorithmen

Im Jahr 1977 haben Rivest, Shamir und Adleman als erste ein Verfahren zum Erzeugen digitaler Signaturen explizit beschrieben. Es handelt sich um das nach seinen Erfindern benannte RSA-Verfahren [10]. In 1984 hat El'gamal [8] einen anderen Signaturalgorithmus vorgeschlagen. Eine Variante dieses El'gamal-Verfahrens ist der 1991 vom National Institute of Standards an Technology (NIST) publizierte Digital Signature Standard (DSS) [1], der den Digital Signature Algorithm (DSA) spezifiziert. Relativ neu sind Varianten des DSA, die auf Punktegruppen E(K) elliptischer Kurven über endlichen Körpern K basieren, wobei $K = F_p$ ein endlicher Primkörper bzw. $K = F_{2^m}$ ein endlicher Körper der Charakteristik 2 ist.

Folgende Signaturalgorithmen sind geeignet:

1. RSA [10],
2. DSA [1]
3. DSA-Varianten, basierend auf elliptischen Kurven, und zwar im einzelnen:
   - ISO/IEC 14883-3 [4], Annex A.2.2 ("Agnew-Mullin-Vanstone analogue"),
   - IEEE-Standard P1363 [5], Abschnitt 5.3.3 ("Nyberg-Rueppel version"),
   - IEEE-Standard P1363 [5], Abschnitt 5.3.4 ("DSA version").

Dies trifft auch für weitere in ISO/IEC 14883-3 beschriebene Verfahren zu. Einige davon werden voraussichtlich in zukünftigen Versionen dieses Dokuments berücksichtigt werden.
Die Sicherheit der oben genannten Verfahren hängt jeweils zusammen mit

1. dem Faktorisierungsproblem für ganze Zahlen,
2. dem Diskreten-Logarithmus-Problem (DLP) in der multiplikativen Gruppe von Primkörpern F_p,
3. dem DLP in Gruppen der Form $E(F_p)$ bzw. $E(F_{2^m})$.

Um festzulegen, wie groß die Systemparameter bei diesen Verfahren gewählt werden müssen, um deren Sicherheit zu gewährleisten, muß man zum einen die besten heute bekannten Algorithmen zum Faktorisieren ganzer Zahlen bzw. zum Berechnen diskreter Logarithmen (in den oben genannten Gruppen) betrachten, und zum anderen die Leistungsfähigkeit der heutigen Rechnertechnik berücksichtigen. Um eine Aussage über die Sicherheit für einen bestimmten zukünftigen Zeitraum zu machen, muß man außerdem eine Prognose für die beiden genannten Aspekte zugrunde legen. Solche Prognosen sind nur für relativ kurze Zeiträume möglich (und können sich natürlich jederzeit aufgrund unvorhersehbarer dramatischer Entwicklungen als falsch erweisen).
Die Sicherheit der einzelnen Verfahren ist (mindestens) für die **kommenden sechs Jahre**, d. h. bis Ende 2003, bei folgender Wahl der Parameter gewährleistet:

1. **RSA**

   Der zugrunde liegende Modulus $n = pq$ (*p* und *q* Primzahlen) soll eine Bitlänge von mindestens 1024 haben:

   $$\log_2(n) = \log_2(p) + \log_2(q) \geq 1024 .$$

   Für den Zeitraum der **kommenden 3 Jahre** reicht eine Minimallänge des Modulus von 768 Bit aus:

   $$\log_2(n) \geq 768 .$$

   Die Primfaktoren *p* und *q* von n sollten ungefähr gleich groß sein, aber nicht zu dicht beieinander liegen, d. h. konkret etwa

   $$0.5 < \left|\log_2(p) - \log_2(q)\right| < 30$$

   Die Primfaktoren *p* und *q* werden unter Beachtung der genannten Nebenbedingungen unabhängig voneinander zufällig erzeugt.
   Der öffentliche Exponent *e* wird mit $ggT(e,(p-1)(q-1)) = 1$ gewählt. Der zugehörige geheime Exponent *d* wird dann berechnet, so daß $ed \equiv 1 \bmod (p-1)(q-1)$ gilt.

   Bemerkung:

   1. Die Forderung, daß *p* und *q starke* Primzahlen sein müssen (d. h. $p-1$ und $q-1$ haben große Primfaktoren etc.), erscheint im Hinblick auf die heute bekannten besten Faktorisierungsalgorithmen nicht mehr ausreichend begründet.
   2. Der öffentliche Exponent sollte im Prinzip zufällig gewählt werden. Auf der anderen Seite haben kleine öffentliche Exponenten den Vorteil, daß die Verifikation der Signatur sehr schnell durchgeführt werden kann. Bei entsprechender Formatierung des Hashwertes, d. h. der Expansion des Hashwertes auf die Blockbreite des asymmetrischen Verfahrens, ist hierbei (im Gegensatz zum Verschlüsseln mit kleinen Exponenten) kein Risiko bekannt.

2. **DSA**

Laut FIPS-186 wird für den Parameter *p* (*p* Primzahl) eine Bitlänge von mindestens 512 und höchstens 1024 verlangt. Diese Bedingung wird hier verschärft, 1024 Bit ist die untere Grenze:

$$\log_2(p) \geq 1024 .$$

Zur Erzeugung von *p* und der weiteren Parameter siehe [1]. Der DSA verlangt für den Parameter *q* die Bitlänge 160. Dies erlaubt die Konstruktion von 'Kollisionen' im Sinne von Serge Vaudenay ('Hidden collisions in DSS', Proceedings of Crypto'96, Lecture Notes in Computer Science, Band 1109, Springer Verlag, 1996, S. 83-88) bei der Parametergenerierung. Diese Kollisionen haben jedoch in der Praxis keine Bedeutung. Wenn man die Möglichkeit, diese Kollisionen konstruieren zu können, ausschließen möchte, muß man $\log_2(q) > 160$ wählen.

3. a) **DSA-Varianten basierend auf** $E(F_p)$

Um die Systemparameter festzulegen, werden eine elliptische Kurve *E* und ein Punkt *P* auf $E(F_p)$ erzeugt, so daß folgende Bedingungen gelten:

- $\#E(F_p) = a \cdot q$ mit einer von *p* verschiedenen Primzahl *q* und

$$\log_2(q) \geq 160 .$$

- $ord(P) = q$.
- $r_0 := \min(r : q \text{ teilt } p^r - 1)$ ist groß, konkret etwa $r_0 > 10^4$.
- Die Klassenzahl des Endomorphismenringes von *E* ist mindestens 100.

Bemerkung: Die untere Abschätzung für r_0 hat den Sinn, Attacken auszuschließen, die auf einer Einbettung der von *P* erzeugten Untergruppe in die multiplikative Gruppe eines Körpers F_{p^r} beruhen. In der Regel (bei zufälliger Wahl der elliptischen Kurve) ist diese Abschätzung erfüllt, denn r_0 ist die Ordnung von $p \pmod q$ in F_q^* und hat deshalb im allgemeinen sogar dieselbe Größenordnung wie *q*. Im Idealfall sollte r_0 explizit bestimmt werden, was allerdings die etwas aufwendige Faktorisierung von $q-1$ voraussetzt. Demgegenüber ist $r_0 > 10^4$ wesentlich schneller zu verifizieren und wird in diesem Zusammenhang als ausreichend angesehen.

3. b) **DSA-Varianten basierend auf** $E(F_{2^m})$

Um die Systemparameter festzulegen, werden eine elliptische Kurve *E* und ein Punkt *P* auf $E(F_{2^m})$ erzeugt, so daß folgende Bedingungen gelten:

- $E(F_{2^m})$ ist in keinem echten Teilkörper von F_{2^m} definierbar (d.h. die j-Invariante der Kurve liegt in keinem echten Teilkörper von F_{2^m}).
- $\#E(F_{2^m}) = a \cdot q$ mit *q* prim und

$$\log_2(q) \geq 160 .$$

- $ord(P) = q$.
- $r_0 := \min(r : q \text{ teilt } 2^{mr} - 1)$ ist groß, konkret etwa $r_0 > 10^4$.
- Die Klassenzahl des Endomorphismenringes von *E* ist mindestens 100.

Bemerkung: In Bezug auf die oben erwähnten 'Kollisionen' im Sinne von Vaudenay gilt für die auf elliptische Kurven basierenden Verfahren dasselbe wie für DSA.

1.5. Erzeugung von Zufallszahlen

Bei der Erzeugung von Systemparametern für Signaturalgorithmen und die Schlüsselgenerierung werden Zufallszahlen benötigt. Bei DSA-ähnlichen Signaturalgorithmen benötigt man bei jeder Generierung einer Signatur eine neue Zufallszahl.
Für diese Zwecke bieten sich als Zufallszahlengeneratoren Systeme an, die

- eine physikalische Rauschquelle und
- eine kryptographische (bzw. mathematische) Nachbehandlung des Primärrauschens

besitzen. Die Entnahme der Bits aus der physikalischen Rauschquelle sollte sich hinreichend gut durch ein stochastisches Modell beschreiben lassen. Das Primärrauschen sollte, soweit dies technisch möglich ist, permanent einem angepaßten statistischen Test unterzogen werden. Die mathematische Nachbehandlung sollte modellbedingte Abhängigkeiten auflösen. Zur Schlüsselerzeugung sollte stets ein physikalischer Zufallszahlengenerator verwendet werden.
Wenn für andere Anwendungen (z. B. beim Signieren mit einer DSA-Variante) kein physikalischer Zufallszahlengenerator zur Verfügung steht, kommt als Alternative ein Pseudozufallsgenerator in Frage. Dieser muß durch eine echte Zufallszahl (seed) initialisiert werden. Entscheidend ist, daß eine dem Pseudozufallsgenerator entnommene Bitfolge - wie eine durch physikalischen Zufall erzeugte Bitfolge - die folgende Anforderung erfüllt:

- Es ist a priori keinerlei Aussage über die Bits möglich, die generiert werden. Auch die Kenntnis einer Teilfolge erlaubt keinerlei Rückschlüsse über die restlichen Bits.

Jedes Verfahren zur digitalen Signatur wird unsicher, wenn der benutzte Zufallszahlengenerator die genannten Forderungen nicht erfüllt.
Deshalb ist jeder Zufallszahlengenerator auf seine Eignung sorgfältig zu überprüfen. Eine aussagekräftige Bewertung eines Zufallszahlengenerators setzt umfassende Erfahrungen voraus. Das Bundesamt für Sicherheit in der Informationstechnik (BSI) verfügt über solche Erfahrungen, und es wird empfohlen, in diesem Zusammenhang auf das Know-how des BSI zurückzugreifen.
Eine sehr nützliche Zusammenstellung von praktischen Kriterien und Hinweisen für Zufalls- und Pseudozufallsgeneratoren findet man in [5], Abschnitt G.

Literatur

[1] NIST: *FIPS Publication 186: Digital Signature Standard (DSS),* Mai 1994.
[2] NIST: *FIPS Publication 180-1: Secure Hash Standard (SHS-1)*, Mai 1995.
[3] ISO/IEC 10118-3: *Information technology – Security techniques - Hash functions - Part 3: Dedicated hash functions*, 1997.
[4] ISO/IEC 14888-3: *Information technology – Security techniques - Digital signatures with appendix – Part 3: Certificate-based mechanisms*, draft, 1997.
[5] IEEE: *P1363 Standard (Draft)*, 6. Februar 1997.
[6] Request for Comments (RFC) 1320: *The MD4 message-digest algorithm*, R. Rivest, Internet Activities Board, Internet Privacy Task Force, April 1992.
[7] H. Dobbertin, A. Bosselaers, B. Preneel: *RIPEMD-160: A strengthend version of RIPEMD*, Fast Software Encryption - Cambridge Workshop 1996, LNCS, Band 1039, S. 71 - 82, Springer-Verlag, 1996. (Weitere Informationen findet man per ftp unter: ftp.esat.kuleuven.ac.be/pub/COSIC/bosselae/ripemd/).
[8] T. El'gamal: *A public key cryptosystem and a signature scheme based on discrete logarithms*, Crypto '84, LNCS, Band 196, S. 10 - 18, Springer-Verlag, 1985.
[9] R. Rivest: *The MD4 message digest algorithm*, Crypto '90, LNCS, Band 537, S. 303 – 311, Springer-Verlag, 1991.
[10] R. Rivest, A. Shamir, L. Adleman: *A method for obtaining digital signatures and public key cryptosystems*, Communications of the ACM, vol. 21 no. 2, 1978.