(8) Die Kennzeichnung akkreditierter ZDA nach § 17 SigG hat die Wortfolge „Akkreditierter Zertifizierungsdiensteanbieter“ zu enthalten. Akkreditierte ZDA sind berechtigt, das Bundeswappen mit dem Schriftzug „Akkreditierter Zertifizierungsdiensteanbieter“ zu führen.

Hinweis auf die Notifikation

§ 15. (1) Diese Verordnung wurde unter Einhaltung der Bestimmungen der Richtlinie 98/34/EG des Europäischen Parlaments und des Rates über ein Informationsverfahren auf dem Gebiet der Normen und technischen Vorschriften in der Fassung der Richtlinie 2006/96/EG des Rates notifiziert (Notifikationsnummer 2007/534/A).

Inkrafttreten

§ 16. Diese Verordnung samt Anhang tritt mit Kundmachung in Kraft. Gleichzeitig tritt die Verordnung des Bundeskanzlers über elektronische Signaturen, BGBl. II Nr. 30/2000, in der Fassung der Verordnung BGBl. II Nr. 527/2004, samt Anhang außer Kraft.

Gusenbauer

ANHANG

Algorithmen und Parameter für qualifizierte elektronische Signaturen

1. Definitionen

1. Signatursuite: Eine Signatursuite besteht aus folgenden Komponenten:
   - einem Signaturalgorithmus mit Parametern,
   - einem Algorithmus zur Schlüsselerzeugung,
   - einem Padding-Verfahren und
   - einer kryptographischen Hashfunktion.

2. Bitlänge: Die Bitlänge einer natürlichen Zahl p ist r, wenn $2^{r-1} \leq p < 2^r$ gilt.

3. Kryptographische Hashfunktion: Der Algorithmus „Hash-Funktion“ ist eine nicht umkehrbare Funktion, die eine umfangreiche Datenmenge (in der Regel einen Text) auf eine im Allgemeinen wesentlich kleinere Zielmenge fester Länge (Hash-Wert) abbildet.

2. Abkürzungen

A9C	„Article 9 Committee“ (Ausschuss für elektronische Signaturen gemäß Art. 9 der Richtlinie 1999/93/EG)
DSA	Digital Signature Algorithm
ECDSA	Elliptic Curve Digital Signature Algorithm
ECGDSA	Elliptic Curve German Digital Signature Algorithm
RSA	Verfahren von Rivest, Shamir und Adleman

3. Zulässige Signatursuiten

Algorithmen und Parameter für qualifizierte elektronische Signaturen dürfen nur in vordefinierten Kombinationen verwendet werden, die als Signatursuiten bezeichnet werden.

Falls eine Komponente der Suite ungültig ist, ist auch die gesamte Suite ungültig. Falls eine Komponente der Suite aktualisiert worden ist, ist auch die gesamte Suite zu aktualisieren.

Tabelle 1a – Liste der zulässigen Signatursuiten:

Kennzahl des Signatursuite-Eintrags	Signatur-Algorithmus	Parameter des Signaturalgorithmus	Algorithmus zur Schlüsselerzeugung	Padding-Verfahren	Kryptographische Hashfunktion
001	rsa	MinModLen = 1024	rsagen1	siehe Tabelle 3	siehe Tabelle 2
002	dsa	pMinLen = 1024 qMinlen = 160	dsagen1	-	siehe Tabelle 2

003	ecdsa-Fp	qMinLen = 160 r0Min = 10^4 MinClass = 200	ecgen1	-	siehe Tabelle 2
004	ecdsa-F2m	qMinlen = 160 r0Min = 10^4 MinClass = 200	ecgen2	-	siehe Tabelle 2
005	ecgdsa-Fp	qMinLen = 160 r0Min = 10^4 MinClass = 200	ecgen1	-	siehe Tabelle 2
006	ecgdsa-F2m	qMinLen = 160 r0Min = 10^4 MinClass = 200	ecgen2	-	siehe Tabelle 2

Einige der in diesem Anhang gegebenen Algorithmen sind über Objektidentifikatoren registriert. Diese werden als Information in Tabelle 1b wiedergegeben.

Tabelle 1b - Objektidentifikatoren (OID)

Objekt-Kurzbezeichnung	Objektidentifikator OID	Bezeichnung in diesem Anhang
rsa	{ joint-iso-ccitt(2) ds(5) module(1) algorithm (8) encryptionAlgorithm(1) 1 }	rsa
rsaEncryption	{ iso(1) member-body(2) us(840) rsadsi (113549) pkcs(1) pkcs-1(1) 1 }	rsa
id-dsa	{ iso(1) member-body(2) us(840) x9-57(10040) x9cm(4) 1 }	dsa
id-ecPublicKey	{ iso(1) member-body(2) us(840) 10045 2 1 }	ecdsa
ecgPublicKey	{ iso(1) identified-organization(3) teletrust(36) algorithm(3) signatureAlgorithm(3) ecSign(2) ecgDsaStd(5) ecgKeyType(2) 1 }	ecgdsa
id-sha1	{ iso(1) identifiedOrganization(3) oIW(14) oIWSecSig(3) oIWSecAlgorithm(2) 26 }	sha1
ripemd160	{ iso(1) identifiedOrganization(3) teletrust(36) algorithm(3) hashAlgorithm(2) ripemd160(1) }	ripemd160
id-sha224	{ joint-iso-itu-t(2)country(16) us(840) organization(1) gov(101) csor(3) nistalgorithm (4) hashalgs(2) sha224(4) }	sha224
id-sha256	{ joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101) csor(3) nistalgorithm (4) hashalgs(2) 1 }	sha256
id-sha384	{ joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101) csor(3) nistalgorithm (4) hashalgs(2) 2 }	sha384
id-sha512	{ joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101) csor(3) nistalgorithm (4) hashalgs(2) 3 }	sha512
whirlpool	{ iso(1) standard(0) encryption-algorithms (10118) part3(3) algorithm(0) whirlpool(55) }	whirlpool
sha-1WithRSAEncryption	{ iso(1) member-body(2) us(840) rsadsi (113549) pkcs(1) pkcs-1(1) 5 }	rsa; emsa-pks1; sha1
sha224WithRSAEncryption	{ iso(1) member-body(2) us(840) rsadsi (113549) pkcs(1) pkcs-1(1) 14 }	rsa; emsa-pks1; sha224
sha256WithRSAEncryption	{ iso(1) member-body(2) us(840) rsadsi (113549) pkcs(1) pkcs-1(1) 11 }	rsa; emsa-pks1; sha256
sha384WithRSAEncryption	{ iso(1) member-body(2) us(840) rsadsi (113549) pkcs(1) pkcs-1(1) 12 }	rsa; emsa-pks1; sha384
sha512WithRSAEncryption	{ iso(1) member-body(2) us(840) rsadsi (113549) pkcs(1) pkcs-1(1) 13 }	rsa; emsa-pks1; sha512

id-RSASSA-PSS mit mgf1SHA1Identifier, mgf1SHA224Identifier, mgf1SHA256Identifier; mgf1SHA384Identifier oder mgf1SHA512Identifier	{ iso(1) member-body(2) us(840) rsadsi (113549) pkcs(1) pkcs-1(1) 10 }	rsa; emsa-pss; sha1, sha224, sha256, sha384, sha512
rsaSignatureWithripemd160	{iso(1) identified-organization(3) teletrust(36) algorithm(3) signatureAlgorithm(3) rsaSignature(1) rsaSignatureWithripemd160 (2)}	rsa; ripemd160
id-dsa-with-sha1	{ iso(1) member-body(2) us(840) x9-57 (10040) x9cm(4) 3 }	dsa; sha1
id-dsa-with-sha224	{ joint-iso-ccitt(2) country(16) us(840) organization(1) gov(101) csor(3) algorithms(4) id-dsa-with-sha2(3) 1 }	dsa; sha224
id-dsa-with-sha256	{ joint-iso-ccitt(2) country(16) us(840) organization(1) gov(101) csor(3) algorithms(4) id-dsa-with-sha2(3) 2 }	dsa; sha256
ecdsa-with-SHA1	{ iso(1) member-body(2) us(840) ansi-X9-62 (10045) signatures(4) sha1(1) }	ecdsa; sha1
ecdsa-with-Recommended	{ iso(1) member-body(2) us(840) ansi-X9-62 (10045) signatures(4) recommended(2) }	ecdsa; sha1, sha224, sha256, sha384, sha512
ecdsa-with-Sha224	{ iso(1) member-body(2) us(840) ansi-X9-62 (10045) signatures(4) specified(3) 1 }	ecdsa; sha224
ecdsa-with-Sha256	{ iso(1) member-body(2) us(840) ansi-X9-62 (10045) signatures(4) specified(3) 2 }	ecdsa; sha256
ecdsa-with-Sha384	{ iso(1) member-body(2) us(840) ansi-X9-62 (10045) signatures(4) specified(3) 3 }	ecdsa; sha384
ecdsa-with-Sha512	{ iso(1) member-body(2) us(840) ansi-X9-62 (10045) signatures(4) specified(3) 4 }	ecdsa; sha512
ecdsa-plain-RIPEMD160	{itu-t(0) identified-organization(4) etsi(0) reserved(127) etsi-identified-organization(0) bsi-de(7) algorithms (1) id-ecc(1) signatures(4) ecdsa-signatures(1) 6 }	ecdsa; ripemd160
ecgSignatureWithripemd160	{ iso(1) identified-organization(3) teletrust(36) algorithm(3) signatureAlgorithm(3) ecSign(2) ecgDsaStd(5) ecgSignature(4) 1 }	ecgdsa; ripemd160
ecgSignatureWithsha1	{ iso(1) identified-organization(3) teletrust(36) algorithm(3) signatureAlgorithm(3) ecSign(2) ecgDsaStd(5) ecgSignature(4) 2 }	ecgdsa; sha1
ecgSignatureWithsha224	{ iso(1) identified-organization(3) teletrust(36) algorithm(3) signatureAlgorithm(3) ecSign(2) ecgDsaStd(5) ecgSignature(4) 3 }	ecgdsa; sha224
ecgSignatureWithsha256	{ iso(1) identified-organization(3) teletrust(36) algorithm(3) signatureAlgorithm(3) ecSign(2) ecgDsaStd(5) ecgSignature(4) 4 }	ecgdsa; sha256
ecgSignatureWithsha384	{ iso(1) identified-organization(3) teletrust(36) algorithm(3) signatureAlgorithm(3) ecSign(2) ecgDsaStd(5) ecgSignature(4) 5 }	ecgdsa; sha384
ecgSignatureWithsha512	{ iso(1) identified-organization(3) teletrust(36) algorithm(3) signatureAlgorithm(3) ecSign(2) ecgDsaStd(5) ecgSignature(4) 6 }	ecgdsa; sha512

4. Zulässige kryptographische Hashverfahren

Für qualifizierte elektronische Signaturen dürfen nur kollisionsresistente Hashfunktionen eingesetzt werden. Diese Voraussetzung ist erfüllt, wenn es rechnerisch nicht realisierbar ist, zwei Dokumente zu finden, die denselben Hashwert liefern.

Tabelle 2 - Liste der derzeit zulässigen Hashfunktionen

Kennzahl der Hashfunktion	Kurzbezeichnung der Hashfunktion
2.01	sha1
2.02	ripemd160
2.03	sha224
2.04	sha256
2.05	sha384
2.06	sha512
2.07	whirlpool

5. Zulässige Padding-Verfahren

Tabelle 3 - Liste der zulässigen Padding-Verfahren

Kennzahl des Padding-Verfahrens	Kurzbezeichnung des Füllverfahrens	Erzeugung der Zufallszahlen	Parameter des Zufallszahlengenerators
3.01	emsa-pkcs1-v1_5	-	-
3.02	emsa-pss	trueran oder pseuran	min. 64 bit
3.03	emsa-pkcs1-v2_1	-	-
3.04	iso9796ds2	trueran oder pseuran	min. 64 bit
3.05	iso9796-din-rn	trueran oder pseuran	min. 64 bit
3.06	iso9796ds3	-	-

6. Zulässige Signaturalgorithmen

Tabelle 4 - Liste der zulässigen Signaturalgorithmen

Kennzahl des Signatur-algorithmus	Kurzbezeichnung des Signaturalgorithmus	Parameter des Signaturalgorithmus	Algorithmus zur Schlüssel- und Parametererzeugung
1.01	rsa	MinModLen = 1024	rsagen1
1.02	dsa	pMinLen = 1024 qMinLen = 160	dsagen1
1.03	ecdsa-Fp	qMinLen = 160 r0Min = 10^4 MinClass = 200	ecgen1
1.04	ecdsa-F2m	qMinLen = 160 r0Min = 10^4 MinClass = 200	ecgen2
1.05	ecgdsa-Fp	qMinLen = 160 r0Min = 10^4 MinClass = 200	ecgen1
1.06	Ecgdsa-F2m	qMinLen = 160 r0Min = 10^4 MinClass = 200	ecgen2

Tabelle 5 - Liste der zulässigen Schlüsselerzeugungsalgorithmen für die in Tabelle 4 aufgelisteten Signaturalgorithmen

Kennzahl des Schlüssel-erzeugungs-algorithmus	Kurzbezeichnung des Schlüsselerzeugungs-algorithmus	Signaturalgorithmus	Verfahren der Zufallszahlen-erzeugung	Parameter des Zufallszahlen-erzeugungs-verfahrens
4.01	rsagen1	rsa	trueran oder pseuran	EntropyBits $\geq$ 80 or SeedLen $\geq$ 80
4.02	dsagen1	dsa	trueran oder pseuran	EntropyBits $\geq$ 80 or SeedLen $\geq$ 80
4.03	ecgen1	ecdsa-Fp, ecgdsa-Fp	trueran oder pseuran	EntropyBits $\geq$ 80 or SeedLen $\geq$ 80
4.04	ecgen2	ecdsa-F2m, ecgdsa-F2m	trueran oder pseuran	EntropyBits $\geq$ 80 or SeedLen $\geq$ 80

7. Erläuterungen zu einzelnen Parametern der zulässigen Signaturalgorithmen

7.1 RSA

Die Sicherheit des RSA-Algorithmus beruht auf der Schwierigkeit, große ganze Zahlen zu faktorisieren. Um die Signaturerstellungsdaten und Signaturprüfdaten zu erzeugen, sind zufällig und unabhängig zwei Primzahlen p und q zu erzeugen, wobei die Bitlänge des Moduls $n = pq$ mindestens MinModLen betragen muss; seine Länge wird auch als ModLen bezeichnet; Jede Primzahl muss effektiv von EntropyBits Bits tatsächlichem Zufall oder einem Ausgangswert der Länge SeedLen beeinflusst sein. p und q sollten etwa dieselbe Länge aufweisen, zB soll ein Bereich wie $\mathbf{0.5 < |\log_2 p - \log_2 q| < 30}$ festgelegt werden.

7.2 DSA

Die Sicherheit des DSA-Algorithmus beruht auf der Schwierigkeit, den diskreten Logarithmus in der multiplikativen Gruppe eines Primkörpers F_p zu berechnen.

Die Signaturerstellungsdaten bestehen aus

- den öffentlichen Parametern p, q und g,
- einer zufällig oder pseudozufällig erzeugten ganzen Zahl x, $0 < x < q$, die signatorspezifisch ist, und
- einer zufällig oder pseudozufällig erzeugten ganzen Zahl k, $0 < k < q$, die für jede Signatur neu zu erzeugen ist.

Die öffentlichen Parameter p, q und g dürfen für eine Gruppe von Benutzern gleich sein. Der prime Modul p muss mindestens pMinLen Bits lang sein. q, das ein Primfaktor von $(p-1)$ ist, muss mindestens qMinLen Bits lang sein.

Die Signaturprüfdaten bestehen aus p, q, g und einer ganzen Zahl y, die als $y = g^x \bmod p$ berechnet wird.

7.2.1 DSA-Varianten mit elliptischen Kurven basierend auf einer Gruppe $E(F_p)$

Die Sicherheit des Algorithmus ecdsa-Fp beruht auf der Schwierigkeit, den diskreten Logarithmus über elliptischen Kurven zu berechnen.

Die öffentlichen Parameter sind wie folgt:

- p eine große Primzahl,
- q eine große Primzahl mit einer Länge von mindestens qMinLen Bits, $p \neq q$;
- E eine elliptische Kurve über dem endlichen Körper F_p, deren Ordnung durch q teilbar ist, und
- P ein fixer Punkt auf E mit der Ordnung q.

Die Klassenzahl der maximalen Ordnung des Endomorphismenrings von E muss mindestens MinClass betragen. Der Wert $r_0 := \min(r: q \text{ teilt } p^r - 1)$ muss größer als r0Min sein.

Die Signaturerstellungsdaten bestehen aus

- den öffentlichen Parametern E, q und P;
- einer statistisch einzigartigen und nicht voraussagbaren ganzen Zahl x, $0 < x < q$, die signatorspezifisch ist und
- einer statistisch einzigartigen und nicht voraussagbaren ganzen Zahl k, $0 < k < q$, die für jede Signatur neu zu erzeugen ist.

Die Signaturprüfdaten bestehen aus E, q, P und einem Punkt Q auf E, der als $Q = xP$ berechnet wird. Die elliptische Kurve über F_p muss so gewählt werden, dass ihre Ordnung durch eine Primzahl q der Länge $\mathbf{\geq qMinLen \geq 160}$ teilbar ist.

7.2.2 DSA-Varianten mit elliptischen Kurven basierend auf einer Gruppe E(F2m)

Die Sicherheit des Algorithmus ecdsa-F2m beruht auf der Schwierigkeit, den diskreten Logarithmus über elliptischen Kurven zu berechnen.

Die öffentlichen Parameter sind wie folgt:

- m eine Primzahl,

- q eine große Primzahl mit einer Länge von mindestens qMinLen Bits,
- E eine elliptische Kurve über dem endlichen Körper F2m, deren Ordnung durch q teilbar ist,
- es darf nicht möglich sein, E über F_2 zu definieren, und
- P ein fixer Punkt auf E mit der Ordnung q.

Die Klassenzahl der maximalen Ordnung des Endomorphismenrings von E muss mindestens MinClass betragen. Der Wert r_0:=min(r: q teilt 2^{mr} -1) muss größer als r0Min sein.

Die Signaturerstellungsdaten bestehen aus

- den öffentlichen Parametern E, q und m;
- einer statistisch einzigartigen und nicht voraussagbaren ganzen Zahl x, $0 < x < q$, die signatorspezifisch ist, und
- einer statistisch einzigartigen und nicht voraussagbaren ganzen Zahl k, $0 < k < q$, die für jede Signatur neu zu erzeugen ist.

Die Signaturprüfdaten bestehen aus E, q, P und einem Punkt Q auf E, der als $Q = xP$ berechnet wird. Die elliptische Kurve über F2m muss so gewählt werden, dass ihre Ordnung durch eine Primzahl q der Länge $\geq$ **qMinLen** $\geq$ **160** teilbar ist.

7.2.3 EC-GDSA basierend auf einer Gruppe $E(F_p)$

Der ecgdsa-Fp Algorithmus ist eine Variante des ecdsa-Fp Algorithmus mit modifizierter Gleichung zur Signaturerstellung und modifiziertem Verfahren zur Signaturprüfung. Die Parameter sind dieselben wie für ecdsa-Fp.

7.2.4 EC-GDSA basierend auf einer Gruppe $E(F2m)$

Der Algorithmus ecgdsa-F2m ist eine Variante des Algorithmus ecdsa-F2m mit modifizierter Gleichung zur Signaturerstellung und modifiziertem Verfahren zur Signaturprüfung.

8. Erzeugung von Zufallszahlen

Tabelle 6– Liste der zulässigen Verfahren zur Erzeugung von Zufallszahlen

Kennzahl des Zufallszahlengenerators	Kurzbezeichnung des Zufallszahlengenerators	Parameter der Zufallszahlen-erzeugung
5.01	Trueran	EntropyBits
5.02	Pseuran	SeedLen
5.03	cr_to_X9.30_x	SeedLen
5.04	cr_to_X9.30_k	SeedLen

8.1 Anforderungen an Zufallszahlengeneratoren trueran

Ein physikalischer Zufallszahlengenerator basiert auf einer physikalischen Rauschquelle (Primärrauschen) und einer kryptographischen oder mathematischen Nachbehandlung des Primärrauschens. Das Primärrauschen muss regelmäßig einer geeigneten statistischen Prüfung unterzogen werden. Der erwartete Aufwand des Erratens eines kryptographischen Schlüssels soll mindestens gleich groß sein, wie der Aufwand des Ratens eines Zufallswerts der Länge EntropyBits.

8.2 Anforderungen an Zufallszahlengeneratoren pseuran

Ein Pseudo-Zufallszahlengenerator muss mit einer echten Zufallszahl initialisiert werden. Der Anfangswert wird als „Seed" bezeichnet und hat die Länge SeedLen. Die Ausgabe des Generators muss folgenden Anforderungen genügen:

- keine Information hinsichtlich der erzeugten Ausgabebits ist vorab bestimmbar;
- die Kenntnis einer Teilsequenz der Ausgabe erlaubt keinen Rückschluss auf ein verbleibendes Bit mit einer Wahrscheinlichkeit, die sich nicht-vernachlässigbar von Zufall unterscheidet;
- es gibt kein verwendbares Verfahren, um aus der Ausgabe des Generators eine zuvor generierte oder zukünftige Ausgabe, einen internen Status oder den Anfangswert („Seed") zu erlangen.

Der erwartete Aufwand des Erlangens jedweden internen Status des Generators soll im Wesentlichen der Schwierigkeit des Erratens eines Zufallswerts der Länge SeedLen Bits sein.

Wenn der Generator mit mindestens SeedLen Bits initialisiert wurde, können bis zu $n = 100$ in Folge erzeugte Signaturerstellungsdaten gleichermaßen verwendet werden, als ob sie von einem Generator trueran erzeugt worden wären. Für die Massenproduktion (durch den ZDA) von k Schlüsseln, $k > n$ ist es zulässig, dass zusätzlich zur initialen Entropieanforderung echter Zufall (von einem trueran Generator) langsam mit einer Rate von $j = 8$ Bits pro Ausgabewert beigegeben wird, andernfalls sollte der Generator komplett neu initialisiert werden.

Wenn Re-Initialisierung angewandt wird, muss die Sicherheit des Re-Initialisierungsprozesses zumindest so stark sein, wie die ursprüngliche Initialisierung und Prozeduren folgen, die der Erstellung von Root-Schlüsseln ähnlich sind. Die Re-Initialisierung von Smartcards ist nicht zulässig.

Keine Backups des Anfangswerts („Seed") oder interner Stati von Pseudo-Zufallszahlengeneratoren sind zulässig."