Bundesamt
für Sicherheit in der
Informationstechnik

Technische Richtlinie BSI TR-03109-1

Anlage I: CMS-Datenformat für die Inhaltsdatenverschlüsselung und -signatur

Version 1.0

Datum: 18.03.2013

Bundesamt für Sicherheit in der Informationstechnik
Postfach 20 03 63
53133 Bonn
E-Mail: SmartMeter@bsi.bund.de
Internet: https://www.bsi.bund.de
© Bundesamt für Sicherheit in der Informationstechnik 2013

Inhaltsverzeichnis

1 Einleitung

In der Infrastruktur von intelligenten Messsystemen kann die Übermittlung von Daten zwischen Smart Meter Gateway und Marktteilnehmern auch über dritte Parteien (etwa den Gateway-Administrator) erfolgen. Im Weitverkehrsnetz (WAN) geschieht der Austausch von Daten innerhalb eines TLS-Kanals daher stets auf der Basis von für den Endempfänger auf Inhaltsebene verschlüsselten und signierten Nachrichten.

In diesem Dokument werden die Datenformate für diese Inhaltsdatenverschlüsselung und -signatur spezifiziert. Die in diesem Dokument definierten Datencontainer basieren auf Cryptographic Message Syntax (CMS) [7] mit ECC-Algorithmen aus der Technischen Richtlinie BSI TR-03111-Elliptic Curve Cryptography [2].

Die aktuell geltenden kryptographischen Anforderungen sind in TR-03116-3 [3] zu finden.

Die in diesem Dokument definierten CMS-Datenstrukturen müssen im Energie-Sektor beim Smart-Metering für die Inhaltsdatenverschlüsselung und -signatur nach [1] bei der Kommunikation zwischen Smart-Meter-Gateways und externen Marktteilnehmern sowie dem Smart Meter Gateway Administrator im WAN verwendet werden.

2 CMS-Datenformat der Inhaltsdatenverschlüsselung

Die Datenstruktur Authenticated-Enveloped-Data [5] soll für die Versendung verschlüsselter, MAC-gesicherter CMS-Nachrichten verwendet werden.

2.1 Authenticated-Enveloped-Data-Content-Type mit ECKA-EG

Der Content-Type `AuthEnvelopedData` hat gemäß [5] die folgende OID:

```
id-ct-authEnvelopedData OBJECT IDENTIFIER ::= {
    iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) ct(1) 23 }
```

Es soll das folgende Profil der Datenstruktur verwendet werden. Die ersten drei Spalten beziehen sich dabei auf die Definition der Datenstruktur gemäß [5]. Die rechte Spalte gibt verbindlich vor, wie das entsprechende Feld zu füllen ist.

Feld	***Typ***	***Wert***
AuthEnvelopedData	***SEQUENCE***	
version	CMSVersion	'0'
originatorInfo	OPTIONAL SEQUENCE	Entfällt, da der Public Key des Senders ephemeral ist.
certs	OPTIONAL [0] IMPLICIT CertificateSet	Entfällt
crls	OPTIONAL [1] IMPLICIT RevocationInfoChoices	Entfällt
RecipientInfos	SET SIZE (1..MAX) OF	Enthält Infos über den Empfänger

Feld	*Typ*	*Wert*
	`RecipientInfo (CHOICE: ktri KeyTransRecipientInfo, kari [1] KeyAgreeRecipientInfo, kekri [2] KEKRecipientInfo, pwri [3] PasswordRecipientInfo, ori [4] OtherRecipientInfo)`	Es muss die Choice `kari KeyAgreeRecipientInfo Type (SEQUENCE)` ausgewählt werden.
`version`	`CMSVersion`	'3'
`originator`	`[0] EXPLICIT OriginatorKeyOrIdentifier (CHOICE: issuerAndSerNo IssuerAndSerNo, subjectKeyId [0] SubjectKeyId, originatorKey [1] OriginatorPublicKey)`	Es wird die Choice `originatorKey OriginatorPublicKey` verwendet.
`algorithm`	`AlgorithmIdentifier`	Daten des ephemeralen Key Agreement Public Key des Senders
`algorithm`	`OID`	`id-ec-publicKey`
`parameters`	`ANY DEFINED BY ALGORITHM`	`named curve (OID)` gemäß 4 (oder `absent`) Die aktuell zu verwendenden Werte sind in [3] zu finden.
`publicKey`	`BIT STRING`	Wert des ephemeralen Public Keys des Absenders
`ukm`	`OPTIONAL [1] EXPLICIT UserKeyingMaterial (OCTET STRING)`	entfällt

Feld	*Typ*	*Wert*
keyEncryptionAlgorithm	KeyEncryptionAlgorithmIdentifier (AlgorithmIdentifier)	Informationen zum Schlüsselableitungs-algorithmus für die Verschlüsselung des Content-Encryption-Keys
algorithm	OBJECT IDENTIFIER	Es müssen die Algorithmen aus Kp. 2.2.1 unterstützt werden. Die dabei aktuell zu verwendenden Werte sind in [3] zu finden.
parameters	ANY DEFINED BY algorithm	Enthält den symmetrischen Key Encryption-Algorithmus für die Ver-schlüsselung des Content-Encryption-Keys.
	AlgorithmIdentifier (SEQUENCE)	Es müssen die Algorithmen aus Kp. 2.2.2 unterstützt werden. Die dabei aktuell zu verwendenden Werte sind in [3] zu finden.
recipientEncryptedKeys	RecipientEncryptedKeys (SEQUENCE OF RecipientEncryptedKey (SEQUENCE))	Enthält die Infos über den Content-Encryption-Key
rid	KeyAgreeRecipientIdentifier (CHOICE: issuerAndSerNo IssuerAndSerNo, rKeyID [0] IMPLICIT RecipientKeyIdentifier)	Es wird die Choice RecipientKeyIdentifier gewählt

Feld	*Typ*	*Wert*
encryptedKey	EncryptedKey (OCTET STRING)	Struktur abhängig vom gewählten contentEncryptionAlgorithm. Es müssen die Algorithmen aus Kp. 2.2.3 unterstützt werden. Die dabei aktuell zu verwendenden Werte sind in [3] zu finden.
authEncryptedContentInfo	EncryptedContentInfo (SEQUENCE)	Enthält den verschlüsselten und authentisierten Inhalt der Message
contentType	ContentType (OID)	Datentyp des Inhalts (id-data bzw. andere (compressed-data, ...)
contentEncryptionAlgorithm	ContentEncryptionAlgorithmIdentifier (AlgorithmIdentifier)	Es müssen die Algorithmen aus Kp. 2.2.3 unterstützt werden. Die dabei aktuell zu verwendenden Werte sind in [3] zu finden.
encryptedContent	OPTIONAL [0] IMPLICIT Encrypted Content (OCTET STRING)	Enthält den verschlüsselten Inhalt der Nachricht.
authAttrs	OPTIONAL [1] IMPLICIT AuthAttributes	Abhängig vom ContentType der verschlüsselten Daten. id-data: entfällt compressed-data: Enthält das content-Type-Attribut.
mac	MessageAuthenticationCode (OCTET STRING)	MAC-Wert
unauthAttrs	OPTIONAL [2] IMPLICIT UnauthAttributes	entfällt

2.2 Unterstützte Algorithmen

2.2.1 OIDs für den ECKA-EG-Algorithmus

Es sollen folgende OIDs aus TR-03111[2] unterstützt werden.

```
ecka-eg-X963KDF-SHA256 OBJECT IDENTIFIER ::= {ecka-eg ecka-eg-x963KDF(1) 3}
ecka-eg-X963KDF-SHA384 OBJECT IDENTIFIER ::= {ecka-eg ecka-eg-x963KDF(1) 4}
ecka-eg-X963KDF-SHA512 OBJECT IDENTIFIER ::= {ecka-eg ecka-eg-x963KDF(1) 5},
```

wobei

```
ecka-eg OBJECT IDENTIFIER ::= {id-ecc key-establishment(5) 1}
id-ecc OBJECT IDENTIFIER ::= {
     itu-t(0) identified-organization(4) etsi(0) reserved (127) etsi-identified-organization(0)
     bsi-de(7) algorithms(1) 1}.
```

2.2.2 Key Encryption-Algorithmen

Für die Key Encryption sollen die Algorithmen `id-aes128-wrap`, `id-aes192-wrap`, `id-aes256-wrap` aus [4] unterstützt werden. Das Parameter-Feld bleibt bei diesen Algorithmen leer.

2.2.3 Content-Authenticated-Encryption

Es sollen die folgenden Algorithmen und Betriebsarten für die authentisierte Content-Encryption unterstützt werden.

2.2.3.1 AES im GCM-Mode

Für den GCM-Mode sollen die Algorithmen mit den OIDs `id-aes128-gcm`, `id-aes192-gcm`, `id-aes256-gcm` aus [6] unterstützt werden. Das Parameter-Feld dieser Algorithmen ist ebenfalls in [6] zu finden.

Bei Verwendung einer dieser OIDs ist im Feld `encryptedKey` der mit dem gewählten Key-Encryption-Algorithmus verschlüsselte (zufällig erzeugte) Content-Encryption-Key *K* enthalten (AESxxx-Wrap(*K*)).

Der Content-Encryption-Key K für die Verschlüsselung und MAC-Sicherung der Daten muss stets unmittelbar vor seiner Verwendung zufällig erzeugt werden und darf jeweils nur für die Versendung einer Nachricht verwendet werden.

2.2.3.2 AES im CBC-CMAC-Mode

Außerdem soll die Verschlüsselung und MAC-Sicherung im CBC-CMAC-Mode via der Algorithmen `id-aes-CBC-CMAC-128`, `id-aes-CBC-CMAC-192`, `id-aes-CBC-CMAC-256` unterstützt werden. Die OIDs werden in Anhang A definiert.

Das `parameters`-Feld muss hier weggelassen werden, d.h. es gilt IV=0.und der MAC besteht aus 16 `OCTETS`. (vgl. Anhang A).

Bei Verwendung einer dieser OIDs ist im Feld `encryptedKey` die mit dem gewählten Key-Encryption-Algorithmus verschlüsselte Konkatenation $K_{enc}||K_{MAC}$ enthalten (AESxxx-Wrap($K_{enc}||K_{MAC}$)). Hierbei ist K_{enc} der (zufällig erzeugte) Schlüssel für die Verschlüsselung des Inhalts mit AES im CBC-Mode und K_{MAC} der (zufällig erzeugte) Schlüssel für die MAC-Sicherung mit AES im CMAC-Mode.

Die Schlüssel K_{enc} und K_{MAC} für die Verschlüsselung und MAC-Sicherung der Daten müssen stets unmittelbar vor seiner Verwendung zufällig erzeugt werden und dürfen jeweils nur für die Versendung einer Nachricht verwendet werden.

3 CMS-Datenformat der Inhaltsdatensignatur

Es soll die Datenstruktur Signed-Data [5] für die Versendung der signierten CMS-Nachrichten verwendet werden.

3.1 Signed-Data Content-Type mit ECDSA

Der Content-Type Signed`Data` hat gemäß [5] die folgende OID:

```
id-ct-authEnvelopedData OBJECT IDENTIFIER ::= {
     iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-7(7) 2 }
```

Es soll das folgende Profil der Datenstruktur verwendet werden. Die ersten drei Spalten beziehen sich dabei auf die Definition der Datenstruktur gemäß ***[5].*** Die rechte Spalte gibt verbindlich vor, wie das entsprechende Feld zu füllen ist.

Feld	***Typ***	***Wert***
`Signed-Data`		
`version`	`CMSVersion`	'3'
`digestAlgorithms`	`DigestAlgorithmIdentifiers (SET OF DigestAlgorithmIdentifier (AlgorithmIdentifier))`	Siehe Kp. 3.2.1
`encapContentInfo`	`EncapsulatedContentInfo (SEQUENCE)`	Enhält die zu signierende Nachricht.
`eContentType`	`ContentType (OBJECT IDENTIFIER)`	`Id-ct-authEnvelopedData` (vgl. Kp 2.1)
`eContent`	`[0] EXPLICIT OCTET STRING`	Datenstruktur aus Kp. 2.1

Feld	*Typ*	*Wert*
Certificates	OPTIONAL [0] IMPLICIT CertificateSet (SET OF CertificateChoices CHOICE: certificate Certificate, extendedCertficate [0] IMPLICIT ExtendedCertificate, v1AttrCert [1] IMPLICIT AttributeCertificateV1, v2AttrCert [2] IMPLICIT AttributeCertificateV2, other [3] IMPLICIT OtherCertificateFormat)	Kann optional das X.509-Signaturzertifikat enthalten, mit dem die Nachricht signiert ist. Ansonsten entfällt das Feld.
crls	OPTIONAL [1] IMPLICIT RevocationInfoChoices (SET OF RevocationInfoChoices CHOICE: crl CertificateList, other [1] IMPLICIT OtherRevocationInfoFormat)	entfällt
signerInfos	SignerInfos (SET OF SignerInfo (SEQUENCE))	Besteht aus einem `SignerInfo` und enthält die Informationen über den Ersteller der Signatur.
version	CMSVersion	3
sid	SignerIdentifier (CHOICE: issuerAndSerNo IssuerAndSerNo, subjectKeyId [0] SubjectKeyId)	Es wird die Choice „`subjectKeyId`“ gewählt.

Feld	*Typ*	*Wert*
digestAlgorithm	DigestAlgorithIdentifier (AlgorithmIdentifier)	Enthält Infos über den Hash-Algorithmus. Es müssen die Algorithmen aus Kp. 3.2.1 unterstützt werden. Die dabei aktuell zu verwendenden Werte sind in [3] zu finden.
signedAttr	OPTIONAL [0] IMPLICIT SignedAttributes (SET SIZE (1..MAX) OF Attribute (SEQUENCE))	Enthält das Content-Type Attribut.
attrType	OBJECT IDENTIFIER	id-contentType und id-messageDigest
attrValues	SET OF AttributeValue (ANY)	Enthält genau einen Attribut-Wert
contentType	OBJECT IDENTIFER	id-ct-authEnvelopedData
SignatureAlgorithm	SignatureAlgorithmIdentifier (AlgorithmIdentifier)	Enthält den Algorithmus, mit dem die Nachricht signiert wurde. Es müssen die Algorithmen aus Kp. 3.2.2 unterstützt werden. Die dabei aktuell zu verwendenden Werte sind [3] zu finden.
signature	SignatureValue (OCTET STRING)	Enthält das Ergebnis der Signaturerzeugung.
unsignedAttr	OPTIONAL [1] IMPLICIT UnsignedAttributes (SET SIZE (1..MAX) OF Attribute (SEQUENCE))	entfällt

3.2 Unterstützte Algorithmen

3.2.1 OIDs für Hashfunktionen

Es sind die OIDs `id-sha-256`, `id-sha-384`, `id-sha-512` für die Berechnung des Message Digest zu unterstützen. Die OIDs sind in [8] spezifiziert. Das Parameter-Feld bleibt leer.

3.2.2 Signaturalgorithmen

Die Signaturalgorithmen mit den folgenden OIDs aus TR-03111[2] sind für die Signaturerzeugung zu unterstützen:

```
ecdsa-with-Sha-256 OBJECT IDENTIFIER ::= {id-ecSigType ecdsa-with-specified(3) 2},
ecdsa-with-Sha-384 OBJECT IDENTIFIER ::= {id-ecSigType ecdsa-with-specified(3) 3},
ecdsa-with-Sha-512 OBJECT IDENTIFIER ::= {id-ecSigType ecdsa-with-specified(3) 4},
```

wobei

```
id-ecSigType OBJECT IDENTIFIER ::= {iso(1) member-body(2) us(840) ansi-X9-62(10045) signatures(4)}
```

Als Signatur-Algorithmus ist hierbei jeweils die OID zu verwenden, deren Suffix mit der Hash-Funktion übereinstimmt, welche im `DigestAlgorithm`-Feld der entsprechenden `SignerInfos` enthalten ist.

Das Parameter-Feld entfällt.

Die Codierung der Signatur muss im X9.62 Format gemäß [2], Kp. 5.2.2 erfolgen.

4 EC-Domain-Parameter

Die NIST- und Brainpool-Kurven über Primkörpern mit folgenden OIDs werden von diesem Dokument unterstützt:

```
secp256r1 OBJECT IDENTIFIER::= {iso(1) member-body(2) us(840) ansi-X9-62(10045) curves(3) prime(1) 7}
secp384r1 OBJECT IDENTIFIER::= {iso(1) identified-organization(3) certicom(132) curve(0) 34}
```

bzw.

```
brainpoolP256r1 OBJECT IDENTIFIER ::= {ellipticCurve versionOne(1) 7}
brainpoolP384r1 OBJECT IDENTIFIER ::= {ellipticCurve versionOne(1) 11}
brainpoolP512r1 OBJECT IDENTIFIER ::= {ellipticCurve versionOne(1) 13},
```

wobei

```
ellipticCurve OBJECT IDENTIFIER::= {
     iso(1) identified-organization(3) teletrust(36) algorithm(3) signature-algorithm(3)
     ecSign(2) ecStdCurvesAndGeneration(8) 1}
```

Verbindliche Vorgaben über die jeweils aktuell zu verwendenden EC-Domain-Parameter sind in [3] zu finden.

A AES-CBC-CMAC Algorithm Identifier und Parameter

Dieser Anhang enthält die allgemeine Definition der Algorithm-Identifier für AES im CBC-CMAC-Mode sowie ihre Verwendung bei der Content-Encryption mit dem CMS Content Type Authenticated-Enveloped-Data. Die konkrete Verwendung der OIDs im Falle der Inhaltsdatenverschlüsselung wird in Kp. 2.2.3.2 festgelegt.

Die folgenden OIDs definieren die entsprechenden AES-Algorithmen im CBC-CMAC-Mode.

```
id-aes-CBC-CMAC-128 OBJECT IDENTIFIER ::= { bsi-de algorithm(1) x.x.x }
id-aes-CBC-CMAC-192 OBJECT IDENTIFIER ::= { bsi-de algorithm(1) x.x.x }
id-aes-CBC-CMAC-256 OBJECT IDENTIFIER ::= { bsi-de algorithm(1) x.x.x }
```

Für diese Algorithmen bezeichne K_{enc} den (zufällig erzeugten) Schlüssel für die Verschlüsselung des Inhalts mit AES im CBC-Mode und K_{MAC} den (zufällig erzeugten) Schlüssel für die MAC-Sicherung mit AES im CMAC-Mode (jeweils in der entsprechenden Bitlänge).

In allen Fällen ist das `parameters`-Feld optional.

Ist das `parameters`-Feld nicht vorhanden, müssen für den CBC-Mode IV=0 und als MAC-Länge 16 Oktetts verwendet werden. In diesem Fall müssen die Content-Encryption-Keys K_{enc} und K_{MAC} unmittelbar vor der Verwendung zufällig erzeugt werden und dürfen nur für die Verschlüsselung und MAC-Sicherung einer Nachricht verwendet werden.

Ist das `parameters`-Feld vorhanden, so besitzt es folgende Struktur:

```
CBC-CMAC-Param ::= SEQUENCE {
     aes-iV OCTET STRING
     aes-MacLen INTEGER (12 | 13 | 14 | 15 | 16) DEFAULT 16 }
```

Das Feld `aes-iV` enthält den Initialisierungsvektor für die CBC-Verschlüsselung und soll aus 16 `OCTETS` bestehen. Das Feld `aes-MacLen` gibt die Länge des Outputs des CMACs. Es wird empfohlen eine MAC-Länge von 16 `OCTETS` zu verwenden. Innerhalb der Verwendungszeit der zufälligen

Content-Encryption-Keys K_{enc} und K_{MAC} muss ein IV für die Verschlüsselung einer Nachricht unvorhersagbar sein. Insbesondere darf ein IV bei gleichen Schlüsseln praktisch nicht zweimal verwendet werden.

Um im Falle des Vorhandenseins des Parameter-Feldes die Authentiziät des Initialisierungvektors für den CBC-Mode sicherzustellen, müssen die Parameter als authentisierte Attribute im Feld `authAttr` in die MAC-Berechnung eingehen. Hierbei muss als Content-Type die folgende OID verwendet werden:

```
id-aes-CBC-CMAC-Param OBJECT IDENTIFIER ::= { bsi-de x.x.x.x}
```

Literaturverzeichnis

[1] BSI TR-03109-1, Anforderungen an die Interoperabilität der Kommunikationseinheit eines intelligenten Messsystems, 2013
[2] BSI TR-03111, Elliptic Curve Cryptography, Version 2.0, 2012
[3] BSI TR-03116-3, eCard-Projekte der Bundesregierung - Kryptographische Vorgaben für die Infrastruktur von intelligenten Messsystemen
[4] IETF RFC 3565, J. Schaad, Use of AES Encryption Algorithm in CMS, 2003
[5] IETF RFC 5083, R. Housley, Cryptographic Message Syntax (CMS) - Authenticated-Enveloped-Data Content Type, 2007
[6] IETF RFC 5084, R. Housley, Using AES-CCM and AES-GCM Authenticated Encryption in the Cryptographic Message Syntax, 2007
[7] IETF RFC 5652, R. Housley, Cryptographic Message Syntax (CMS), 2009
[8] IETF RFC 5754, S. Turner, Using SHA2 Algorithms with Cryptographic Message Syntax, 2010