Bundesamt
für Sicherheit in der
Informationstechnik

Technische Richtlinie BSI TR-03109-1

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

Version 1.0.9

Datum: 18.12.2019

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

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) [9] 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.

Für die genauere Unterscheidung zwischen normativen und informativen Inhalten werden die dem [11] entsprechenden in Großbuchstaben geschriebenen, deutschen Schlüsselworte verwendet:

MUSS bedeutet, dass es sich um eine normative Anforderung handelt.

DARF NICHT / DARF KEIN bezeichnet den normativen Ausschluss einer Eigenschaft.

SOLL beschreibt eine dringende Empfehlung. Abweichungen zu diesen Festlegungen müssen begründet werden.

SOLL NICHT / SOLL KEIN kennzeichnet die dringende Empfehlung, eine Eigenschaft auszuschließen. Abweichungen zu diesen Festlegungen müssen begründet werden.

KANN / DARF bedeutet, dass die Eigenschaften fakultativ oder optional sind.

2 CMS-Datenformat der Nachrichten

Die Datenstrukturen nach Kapitel 3 und Kapitel 4 MÜSSEN für den Transport in die Datenstruktur `ContentInfo` nach [9] eingebettet werden.

Als `ContentType` in der Datenstruktur `ContentInfo MUSS` die OID id-signedData verwendet werden.

3 CMS-Datenformat der Inhaltsdatenverschlüsselung

Die Datenstruktur AuthEnvelopedData [7] MUSS in authentisch verschlüsselten CMS-Nachrichten verwendet werden.

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

Der Content-Type für die Datenstruktur `AuthEnvelopedData` hat gemäß [7] 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 MUSS das folgende Profil der Datenstruktur verwendet werden. Die ersten drei Spalten beziehen sich dabei auf die Definition der Datenstruktur gemäß [7]. Die rechte Spalte gibt verbindlich vor, wie das entsprechende Feld zu füllen ist.

Feld	*Anforderung*	*Tag*	*Typ*	*Wert*
AuthEnvelopedData	***MUSS***		***SEQUENCE***	
version	MUSS		CMSVersion	'0'
originatorInfo	DARF NICHT		SEQUENCE	Entfällt, da der Public Key des Senders ephemer ist.
certs	DARF NICHT	[0] IMPLICIT	CertificateSet	Entfällt
crls	DARF NICHT	[1] IMPLICIT	RevocationInfoChoices	Entfällt
RecipientInfos	MUSS		SET SIZE (1..MAX) OF	Enthält Infos über den Empfänger
	MUSS	[1] IMPLICIT	RecipientInfo (CHOICE: ktri KeyTransRecipientInfo, kari [1] IMPLICIT KeyAgreeRecipientInfo, kekri [2] IMPLICIT KEKRecipientInfo, pwri [3] IMPLICIT PasswordRecipientInfo, ori [4] IMPLICIT OtherRecipientInfo)	Es MUSS die Variante kari (Typ: KeyAgreeRecipientInfo SEQUENCE) ausgewählt werden.
version	MUSS		CMSVersion	'3'
originator	MUSS	[0] EXPLICIT	OriginatorKeyOrIdentifier	

Feld	*Anforderung*	*Tag*	*Typ*	*Wert*
		[1] EXPLICIT	`(CHOICE: issuerAndSerNo IssuerAndSerNo, subjectKeyId [0] SubjectKeyId, originatorKey [1] OriginatorPublicKey)`	Es MUSS die Variante `originatorKey (Typ: OriginatorPublicKey)` verwendet werden.
`algorithm`	MUSS		`AlgorithmIdentifier`	Daten des ephemeren Key Agreement Public Key des Senders
`algorithm`	MUSS		`OBJECT IDENTIFIER`	`id-ecPublicKey`
`parameters`	KANN		`ANY DEFINED BY algorithm`	`Falls vorhanden: namedCurve (OID)` gemäß Kap. 5 Die aktuell zu verwendenden Werte sind in [3] zu finden. Sofern namedCurve vorhanden ist, MUSS die OID der namedCurve des der rid zugeordneten PublicKeys des Empfängers entsprechen. Sofern namedCurve nicht vorhanden ist, MUSS implizit die namedCurve-OID des der rid zugeordneten PublicKeys verwendet werden.
`publicKey`	MUSS		`BIT STRING`	Wert des ephemeren Public Keys des Absenders
`ukm`	DARF NICHT	[1] EXPLICIT	`UserKeyingMaterial (OCTET STRING)`	entfällt

Feld	*Anforderung*	*Tag*	*Typ*	*Wert*
keyEncryptionAlgorithm	MUSS		KeyEncryptionAlgorithmIdentifier (AlgorithmIdentifier)	Informationen zum Schlüsselableitungsalgorithmus für die Verschlüsselung des Content-Encryption-Keys
algorithm	MUSS		OBJECT IDENTIFIER	Algorithmus aus Kap. 3.2.1 Die dabei aktuell zu verwendenden Werte sind in [3] zu finden.
parameters	MUSS		ANY DEFINED BY algorithm	Enthält den symmetrischen Key Encryption-Algorithmus für die Verschlüsselung des Content-Encryption-Keys.
	MUSS		AlgorithmIdentifier (SEQUENCE)	Algorithmus aus Kap. 3.2.2 Die dabei aktuell zu verwendenden Werte sind in [3] zu finden.
recipientEncryptedKeys	MUSS		RecipientEncryptedKeys (SEQUENCE OF RecipientEncryptedKey (SEQUENCE))	Enthält die Infos über den Content-Encryption-Key
rid	MUSS	[0] IMPLICIT	KeyAgreeRecipientIdentifier (CHOICE: issuerAndSerNo IssuerAndSerNo, rKeyID [0] IMPLICIT RecipientKeyIdentifier)(SEQUENCE)	Es wird die Variante rKeyID (Typ: RecipientKeyIdentifier) gewählt. Die optionalen Felder date und other in der SEQUENCE RecipientKeyIdentifier entfallen.
subjectKeyIdentifier	MUSS		SubjectKeyIdentifier (OCTET STRING)	Enthält den SKI des Zertifikates des Empfängers für den verschlüsselt wurde.

Feld	*Anforderung*	*Tag*	*Typ*	*Wert*
date	DARF NICHT		`GeneralizedTime`	entfällt
other	KANN		`OtherKeyAttribute`	Kann Hinweise für den Empfänger enthalten, den passenden PublicKey auszuwählen.
`encryptedKey`	MUSS		`EncryptedKey (OCTET STRING)`	Struktur abhängig vom gewählten `contentEncryptionAlgorithm`. Es müssen die Algorithmen aus Kap. 3.2.3 unterstützt werden. Die dabei aktuell zu verwendenden Werte sind in [3] zu finden.
`authEncryptedContentInfo`	MUSS		`EncryptedContentInfo (SEQUENCE)`	Enthält den verschlüsselten und authentisierten Inhalt der Nachricht
`contentType`	MUSS		`ContentType (OBJECT IDENIFIER)`	Datentyp des Inhalts
`contentEncryptionAlgorithm`	MUSS		`ContentEncryptionAlgorithmIdentifier (AlgorithmIdentifier)`	Algorithmus aus Kap. 3.2.3 Die dabei aktuell zu verwendenden Werte sind in [3] zu finden.
`encryptedContent`	KANN	[0] IMPLICIT	`EncryptedContent (OCTET STRING)`	Enthält den verschlüsselten Inhalt der Nachricht.

Feld	*Anforderung*	*Tag*	*Typ*	*Wert*
authAttrs	KANN	[1] IMPLICIT	AuthAttributes	Abhängig vom contentType der verschlüsselten Daten, vgl. [7]. MUSS vorhanden sein, falls contentType in EncryptedContentInfo nicht vom Typ id-data: ist. In diesem Fall MUSS mindestens das contentType-Attribut enthalten sein.
mac	MUSS		MessageAuthenticationCode (OCTET STRING)	MAC-Wert
unauthAttrs	DARF NICHT	[2] IMPLICIT	UnauthAttributes	entfällt

3.2 Unterstützte Algorithmen

3.2.1 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 ::= {
     bsi-de algorithms(1) 1}.
bsi-de OBJECT IDENTIFIER ::= {
     itu-t(0) identified-organization(4) etsi(0) reserved (127) etsi-identified-organization(0) 7 }
```

3.2.2 Key Encryption-Algorithmen

Für die Key Encryption SOLLEN die Algorithmen mit den OIDs `id-aes128-wrap`, `id-aes192-wrap`, `id-aes256-wrap` aus [5] unterstützt werden. Das Parameter-Feld bleibt bei diesen Algorithmen leer. Die Algorithmen sind in [4] Kapitel 2.2 beschrieben.

3.2.3 Content-Authenticated-Encryption

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

3.2.3.1 AES im GCM-Mode

Für den GCM-Mode SOLLEN die Algorithmen mit den OIDs `id-aes128-gcm`, `id-aes192-gcm`, `id-aes256-gcm` aus [8] unterstützt werden. Das Parameter-Feld dieser Algorithmen ist ebenfalls in [8] zu finden.

Das `parameters`-Feld MUSS eine `aes-ICVlen` von 16 `OCTETS` enthalten.

Das `parameters`-Feld MUSS eine `nonce` mit einer Länge von 12 `OCTETS` enthalten

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 MUSS jeweils nur für die Versendung einer Nachricht verwendet werden.

3.2.3.2 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($K_{enc}||K_{MAC}$) nach [4] Kapitel 2.2). 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.

4 CMS-Datenformat der Inhaltsdatensignatur

Es MUSS die Datenstruktur SignedData [9] in signierten CMS-Nachrichten verwendet werden.

4.1 Signed-Data Content-Type mit ECDSA

Als `ContentType für die Datenstruktur SignedData MUSS` die folgende OID nach [7] verwendet werden:

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

Das Feld `content der Datenstruktur ContentInfo` MUSS dann vom Typ `SignedData` sein, vgl. [7]. Hierbei MUSS das im Folgenden definierte Profil der Datenstruktur verwendet werden. Die ersten drei Spalten beziehen sich dabei auf die Definition der Datenstruktur gemäß [7]**.** Die rechte Spalte gibt verbindlich vor, wie das entsprechende Feld zu füllen ist.

Feld	***Anforderung***	***Tag***	***Typ***	***Wert***
`Signed-Data`	***MUSS***			
`version`	MUSS		`CMSVersion`	'3'
`digestAlgorithms`	MUSS		`DigestAlgorithmIdentifiers (SET OF DigestAlgorithmIdentifier (AlgorithmIdentifier))`	Siehe Kap. 4.2.1
`encapContentInfo`	MUSS		`EncapsulatedContentInfo (SEQUENCE)`	Enthält die zu signierende Nachricht.

Feld	*Anforderung*	*Tag*	*Typ*	*Wert*
eContentType	MUSS		ContentType (OBJECT IDENTIFIER)	id-ct-authEnvelopedData (vgl. Kp 3.1)
eContent	MUSS	[0] EXPLICIT	OCTET STRING	Datenstruktur aus Kap. 3.1
certificates	KANN	[0] IMPLICIT (SEQUENCE certificate)	CertificateSet (SET OF CertificateChoices CHOICE: certificate Certificate, extendedCertficate [0] IMPLICIT ExtendedCertificate, v1AttrCert [1] IMPLICIT AttributeCertificateV1, v2AttrCert [2] IMPLICIT AttributeCertificateV2, other [3] IMPLICIT OtherCertificateFormat)	Das Feld ist optional. Falls es vorhanden ist enthält es das X.509-Signaturzertifikat, mit dem die Nachricht signiert ist.
crls	DARF NICHT	[1] IMPLICIT	RevocationInfoChoices (SET OF RevocationInfoChoices CHOICE: crl CertificateList, other [1] IMPLICIT OtherRevocationInfoFormat)	Entfällt
signerInfos	MUSS		SignerInfos (SET OF SignerInfo (SEQUENCE))	Besteht aus einer SignerInfo und enthält die Informationen über den Ersteller der Signatur.
version	MUSS		CMSVersion	3

Feld	*An-forde-rung*	*Tag*	*Typ*	*Wert*
sid	MUSS	[0]	SignerIdentifier (CHOICE: issuerAndSerNo IssuerAndSerNo, subjectKeyId [0] SubjectKeyId)	Es wird die Variante „subjectKeyId“ gewählt.
digestAlgorithm	MUSS		DigestAlgorithmIdentifier (AlgorithmIdentifier)	Enthält Infos über den Hash-Algorithmus. Es müssen die Algorithmen aus Kap. 4.2.1 unterstützt werden. Die dabei aktuell zu verwendenden Werte sind in [3] zu finden.
signedAttrs	MUSS	[0] IMPLICIT	SignedAttributes (SET SIZE (1..MAX) OF Attribute (SEQUENCE))	Enthält das Content-Type Attribut.
attrType	MUSS		OBJECT IDENTIFIER	id-contentType und id-messageDigest
attrValues	MUSS		SET OF AttributeValue (ANY)	Enthält jeweils den zugehörigen (genau einen) Attribut-Wert
contentType	MUSS		OBJECT IDENTIFIER	id-ct-authEnvelopedData
SignatureAlgorithm	MUSS		SignatureAlgorithmIdentifier (AlgorithmIdentifier)	Enthält den Algorithmus, mit dem die Nachricht signiert wurde. Algorithmus aus Kap. 4.2.2 unterstützt werden. Die dabei aktuell zu verwendenden Werte sind in [3] zu finden.

Feld	*An-forde-rung*	*Tag*	*Typ*	*Wert*
signature	MUSS		SignatureValue (OCTET STRING)	Enthält das Ergebnis der Signaturerzeugung gemäß [7].
unsignedAttr	DARF NICHT	[1] IMPLICIT	UnsignedAttributes (SET SIZE (1..MAX) OF Attribute (SEQUENCE))	Entfällt

4.2 Unterstützte Algorithmen

4.2.1 OIDs für Hashfunktionen

Es SOLLEN die OIDs `id-sha-256`, `id-sha-384`, `id-sha-512` für die Berechnung des Message Digest unterstützt werden. Die OIDs sind in [10] spezifiziert. Das optionale Parameter-Feld entfällt.

4.2.2 Signaturalgorithmen

Die Signaturalgorithmen mit den folgenden OIDs aus TR-03111[2] für die Signaturerzeugung SOLLEN unterstützt werden:

```
ecdsa-with-specified OBJECT IDENTIFIER ::= {id-ecSigType 3},
ecdsa-with-Sha256 OBJECT IDENTIFIER ::= {ecdsa-with-specified) 2},
ecdsa-with-Sha384 OBJECT IDENTIFIER ::= {ecdsa-with-specified 3},
ecdsa-with-Sha512 OBJECT IDENTIFIER ::= {ecdsa-with-specified 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], Kap. 5.2.2 erfolgen.

5 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-Authenticated-Encryption mit dem CMS Content Type Authenticated-Enveloped-Data.

Die konkrete Verwendung der OIDs im Falle der Inhaltsdatenverschlüsselung wird in Kap. 3.2.3.2 festgelegt.

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

```
id-aes-CBC-CMAC-128 OBJECT IDENTIFIER ::= {
     bsi-de algorithm(1) symmetricCiphers(3) AES(1) authEncr (1) 2 }
id-aes-CBC-CMAC-192 OBJECT IDENTIFIER ::= {
     bsi-de algorithm(1) symmetricCiphers(3) AES(1) authEncr (1) 3 }
id-aes-CBC-CMAC-256 OBJECT IDENTIFIER ::= {
     bsi-de algorithm(1) symmetricCiphers(3) AES(1) authEncr (1) 4 }
```

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).

Das Feld `encryptedContent` MUSS die mit AES im CBC-Mode und dem Schlüssel K_{enc} verschlüsselte Nachricht enthalten.

Die Berechnung des Feldes `mac` MUSS gemäß den Vorgaben von [7] mit dem `encryptedContent` sowie den DER-codierten `authAttrs(als AAD)` als Input via AES im CMAC-Mode mit den Schlüssel K_{MAC} erfolgen.

Der CMAC MUSS nach [6] über die Konkatenation aus AAD || C gebildet, wobei C der AES-CBC Verschlüsselte Inhalt und AAD der Inhalt der `authAttrs` ist..

Vor Aufruf der Funktion AES-CBC MUSS der Input dabei jeweils wie in [9] beschrieben gepaddet werden.

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 jedem Fall MÜSSEN die Content-Encryption-Keys K_{enc} und K_{MAC} unmittelbar vor der Verwendung zufällig erzeugt werden und DÜRFEN NUR nur für die Verschlüsselung und MAC-Sicherung einer Nachricht verwendet werden.

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

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

Das Feld `aes-iV` enthält den Initialisierungsvektor für die CBC-Verschlüsselung und MUSS aus 16 `OCTETS` bestehen. Das Feld `aes-MacLen` gibt die Länge des Outputs des CMACs. Es MUSS eine MAC-Länge von 16 `OCTETS` zu verwendet werden. 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 symmetricCiphers(3) AES(1) authEncr (1) 1}
```

Literaturverzeichnis

[1] BSI TR-03109-1, Anforderungen an die Interoperabilität der Kommunikationseinheit eines intelligenten Messsystems
[2] BSI TR-03111, Elliptic Curve Cryptography
[3] BSI TR-03116-3, eCard-Projekte der Bundesregierung - Kryptographische Vorgaben für die Infrastruktur von intelligenten Messsystemen
[4] IETF, J. Schaad, R. Housley RFC 3394, AES Key Wrap Algorithm, 2002
[5] IETF, J. Schaad RFC 3565, Use of AES Encryption Algorithm in CMS, 2003
[6] IETF, JH. Song, R. Poovendran, J. Lee, T. Iwata: RFC4493, The AES-CMAC Algorithm, 2006
[7] IETF, R. Housley RFC 5083, Cryptographic Message Syntax (CMS) - Authenticated-Enveloped-Data Content Type, 2007
[8] IETF, R. Housley RFC 5084, Using AES-CCM and AES-GCM Authenticated Encryption in the Cryptographic Message Syntax, 2007
[9] IETF, R. Housley RFC 5652, Cryptographic Message Syntax (CMS) - Authenticated-Enveloped-Data Content Type, 2007
[10] IETF, S. Turner RFC 5754, Using SHA2 Algorithms with Cryptographic Message Syntax, 2010
[11] IETF, S.Bradner RFC 2119, Key words for use in RFCs to Indicate Requirement Levels, 1997