Open Metering System

Technical Report 01
Security

Issued 1.1.0 - 2012-12-20

Draft

Table of contents

List of tables

Foreword

This Technical Report presents a technical solution of security methods for the wireless M-Bus according to the requirements of BSI TR03109-1 [BSI1] and TR03109-3 [BSI3]. These technical definitions will be integrated in the OMS-Specification Volume 2 - issue 4. For the OMS compliance test only the definitions of the OMS-Specification are applicable!

1 Introduction

This Technical Report describes the implementation of security demands from the technical guideline TR03109-1 [BSI1] and TR03109-3 [BSI3] based on the M-Bus-standard series EN 13757. To fulfill these requirements the standard has to be expanded by a new Authentication and Fragmentation Layer (AFL). Further modifications are required to be able to integrate the Key Derivation Function (KDF) and the Transport Layer Security (TLS) in the M-Bus.

2 Normative references

[BSI1] Technische Richtlinie BSI TR-03109-1 Version 1.0 (Release Candidate of 21 Dez. 2012) „Anforderungen an die Interoperabilität der Kommunikationseinheit eines intelligenten Messsystems für Stoff und Energiemengen“

[BSI3] Technische Richtlinie BSI TR-03109-3 Version 1.0 (Release Candidate of 21 Dez. 2012) „Kryptographische Vorgaben für die Infrastruktur von Messsystemen“

[EN-3] prEN13757-3 (2011) Communication systems for and remote reading of meters – Part 3: Dedicated application layer

[EN-4] prEN13757-4 (2011) Communication systems for meters and remote reading of meters - Part 4: Wireless meter readout (Radio meter reading for operation in SRD bands)

[OMSV2] OMS-Specification Volume 2 “Primary communication” Issue 3.0.1 (2011-01)

[RFC4492] IETF RFC 4492 – “Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS)” (2006-05)

[RFC5289] IETF RFC 5289 – “TLS Elliptic Curve Cipher Suites with SHA-256/384 and AES Galois Counter Mode (GCM)” (2008-08)

[RFC4493] IETF RFC 4493, “The AES-CMAC Algorithm” (2006-06)

[RFC5246] IETF RFC 5246, ” The Transport Layer Security (TLS) Protocol Version 1.2”, (2008-08)

[RFC6066] IETF RFC 6066, “Transport Layer Security (TLS) Extensions: Extension Definitions (2011-01)”

3 Terms and Abbreviations

3.1 Terms

fragment unit of data transferred from source to destination

NOTE: This was named telegrams in previous versions.

message functional set of data transferred from source to destination

NOTE: A message may consist of one or more fragments

3.2 Abbreviations

AFL	Authentication and Fragmentation Layer
ALA	Application Layer Address
APL	Application layer (as defined in [EN-3])
BSI	Bundesamt für Sicherheit in der Informationstechnik (Federal Office for Information Security)
CI	CI-Field is the Control Information Field (applied in [EN-3])
CMAC	Cipher based MAC
ELL	Extended Link Layer (as defined in [EN-4])
KDF	Key derivation function (based on CMAC-algorithm)
LLA	Link Layer Address
LSB	Least Significant Byte
MAC	Message Authentication Code
MSB	Most Significant Byte
TBD	To be defined
TLS	Transport Layer Security (a security standard)
TPL	Transport Layer (as defined in [EN-3])

3.3 Definitions

Statements using a “shall” or a “shall not” are mandatory and must be realized (It is identical to a “MUST”).

Statements using a “should” or “should not” are recommendations, which may not be applied if there are good reasons.

Hexadecimal values use a prefix of “0x” like “0xFF”. Decimal values use no prefix.

4 Necessary extensions of the EN13757-3 [EN-3]

4.1 General

This Technical Report specifies the following extension to the [EN-3] required to use the wireless M-Bus for secure data transmission.

4.2 Encryption Modes for Configuration Field

In the configuration field a new encryption mode for Transport Layer Security (TLS) shall be introduced to indicate a TLS secured application payload for existing Application protocols (like M-Bus, COSEM, alarms etc.). The value 13 (decimal) is used in this Document.

In the configuration field a new encryption mode shall be introduced to extend the operation of AES-Encryption for use with dynamic keys. The value 7 (decimal) is used in this Document.

Table 1 – List of additional Encryption Modes

Symbolic Mode Name	Encryption Mode M	Description	Control Field Size
EM-AESEXT	0x07	AES-CBC with dynamic keys	3 Bytes
EM-TLS	0x0D	Transport Layer Security (TLS) for high protection requirements	3 Bytes

4.3 Control Information (CI)

A new CI-field (CI-AFL) shall be added in Table 1 of [EN-3] to identify the Authentication and Fragmentation Layer as specified in chapter 5.

A new CI-field (CI-TCOL) shall be added in Table 1 of [prEN-3] to identify security management data (e.g. TLS Handshake) as specified in chapter 7.2 from other to meter.

Two new CI fields (CI-TCMS, CI-TCML) shall be added in Table 1 of [prEN-3] to identify security management data (e.g. TLS Handshake) as specified in chapter 7.2 from meter to other.

Table 2 – List of additional CI-Fields

Symbolic CI Name	CI Value	Description	Header Length
CI-AFL	0x90	Authentication and Fragmentation Layer	Variable (3-26)
CI-TCMS	0x9E	Security Management (Transport Layer Control) without address (meter to other Device)	Short (4)
CI-TCML	0x9F	Security Management (Transport Layer Control) with address (meter to other device)	Long (12)
CI-TCOL	0x5F	Security Management (Transport Layer Control) with address (other device to meter)	Long (12)

4.4 Configuration Field (CF)

The Configuration Field in general declares the length and method of data encryption. For encryption mode 7 and 13 additional definitions are necessary. For this an enhancement of the Configuration Field to three bytes (24 Bit) has to be introduced. The length of the Configuration Field will be mode dependent.

Refer to chapter 6 for the definition if the Configuration Field for Mode 7
Refer to chapter 7.2 for the definition if the Configuration Field for Mode 13

5 Authentication and Fragmentation Layer

5.1 Introduction

This section explains the usage of the Authentication and Fragmentation layer (AFL) in combination with the other Layers and encryption modes of the M-Bus.

The Authentication and Fragmentation layer provides two essential services

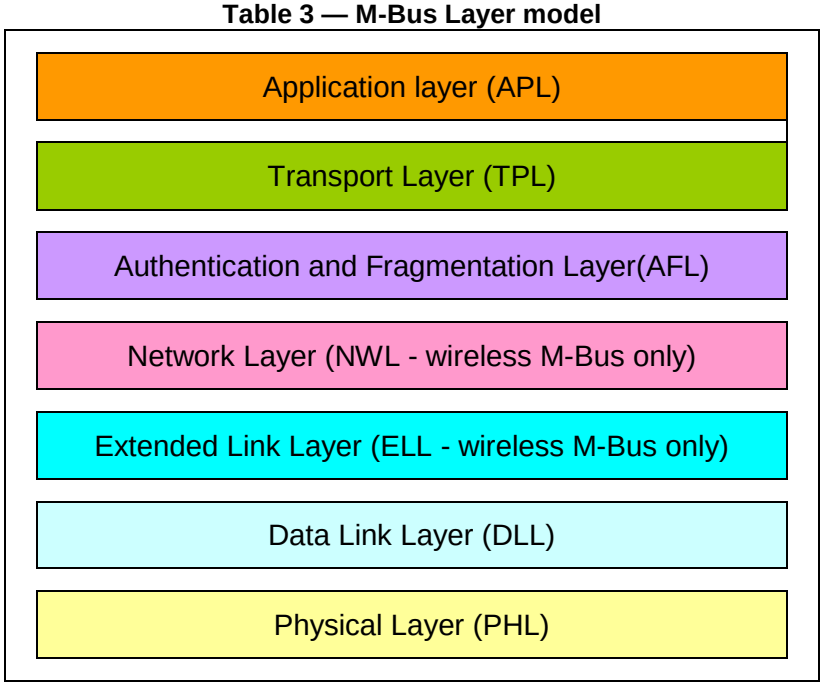
- Fragmentation of long messages in several fragments.
- Adding a Message Authentication Code (MAC) to prove the authenticity of the APL.

This optional Layer shall be applied if at least one of these services is required.

Fragmentation is required to transport large messages especially certificates for the TLS-handshakes. The AFL is separated in a section for each single fragment and a section for the whole message. The AFL as shown in Fig. 1 shall be the next layer under the Application layer or comparable Management or Transport layer. Optional layers like Network layer or Extend Link Layer has to be under the AFL.

The MAC protects all layers above the AFL. Therefore no changes in higher layers are possible, as soon as the MAC has been calculated.

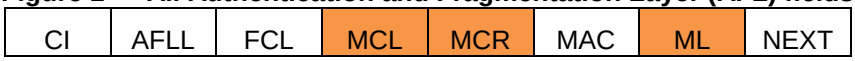
Table 3 — M-Bus Layer model



5.2 Structure of the AFL

5.2.1 Overview

Figure 1 — All Authentication and Fragmentation Layer (AFL) fields



The orange fields shall be included in the MAC calculation. A message shall consist of one or more fragments. Each fragment shall be transported in one Data Link Layer frame. The following Table 4 provides an overview of all possible AFL fields as shown in Figure 1.

A Receiver may ignore new fields or the entire AFL by skipping to the next CI by using the AFL -Length. The gray-shaded columns indicate optional fields. Their inclusion is defined by the Fragmentation Control Field specified in Table 5.

Table 4 — Overview of all AFL fields

Size (bytes)	Field Name	Description
1	CI	Indicates that an Authentication and Fragmentation Layer follows.
1	AFLL	AFL-Length The number of bytes in the AFL following the field AFL.AFLL.
2	FCL	Fragmentation-Control-Field
1	MCL	Message-Control-Field
4	MCR	Message-Counter-Field
8 to 16[1]	MAC	Message-Authentication-Code
2	ML	Message-Length-Field
…	NEXT	Next CI or Fragment Data

[1] NOTE: The applicable length of the CMAC may be limited (refer to 5.4)!

5.2.2 AFL Fragmentation Control Field (AFL.FCL)

The Fragmentation Control Field indicates size and presence of the following fields in the current fragment

Figure 2 — AFL Fragmentation Control Field bitmap (AFL.FCL)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	MF	MCLP	MLP	MCRP	MACP	Res	Res	FID							

The fields AFL.AFLL and AFL.FCL are not part of the MAC protected message.

The bits in the AFL.FCL field define the presence of the respective field in the current fragment.

Table 5 — AFL Fragmentation Control Field bitfield definitions

Bits	Field Name	Description
15	Res	Reserved
14	MF	More-Fragments 0 This is the last fragment. 1 More fragments are following.
13	MCLP	Message-Control in this Fragment Present
12	MLP	Message-Length in this Fragment Present
11	MCRP	Message-Counter in this Fragment Present
10	MACP	MAC in this Fragment is Present
9, 8	-	Reserved
7 to 0	FID	Fragment-ID. Set to1 for the first Fragment. of a fragmented message. Incremented with each fragment. Shall not wrap. [2]

5.2.3 AFL Message Control Field (AFL.MCL)

Figure 3 — AFL Message Control Field bitmap (AFL.MCL)

	7	6	5	4	3	2	1	0
Name	Res	MLMP	MCMP	Res	AT		ATO	

[2] Because the maximum message length is 16 kByte and the fragment size is larger than 64 bytes, the Fragment-.ID cannot wrap.

Table 6 — AFL Message Control Field bitfield definitions

Bits	Field Name	Description
7	Res	Reserved
6	MLMP	Message-Length in Message Present
5	MCMP	Message-Counter in Message Present
4	Res	Reserved
3 & 2	AT	Authentication-Type 00 None 01 CMAC-AES128 (see 5.4) 10 undefined 11 undefined
1 & 0	ATO	Authentication type options: AT=00 (None) always 00 AT=01 (CMAC-AES128) 00 undefined 01 8 bytes (refer to [01]) 10 undefined 11 reserved for max length

The bits in the AFL.MCL field define the presence of a field in at least one fragment of the message.

The AFL.MCL field shall always be present in the first fragment. It shall not be present in any following fragments of the same message.

5.2.4 AFL Message Length Field (AFL.ML)

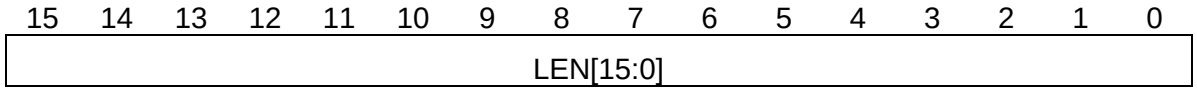


Figure 4 — AFL Message Length Field bitmap

Table 7 — AFL Message Length Field bitfield definitions

Bits	Field Name	Description
15 to 0	LEN	The message length shall be limited to 16 kbytes. Practical limits are lower, determined by the receiver buffer size. The message length contains the sum of all TPL (see Table 4) fragments for one message.. It does not include any AFL Fields.

The AFL.ML Message Length Field shall only be present in the first fragment of a fragmented message to indicate the total message length. For single-fragment messages the AFL.ML shall not be used.

NOTE: The maximum size for an “TLS-Fragment” (not identical to the fragment in this AFL layer context) is 16 kbytes. The maximum Message Length is 16 384 bytes plus 5 bytes for the TLS header plus Long Header (CI,ACC,ST,CF,ADDR) 13 bytes = 16 402 bytes. If both TLS Client and Server agree to use a smaller Maximum TLS-Fragment Size (Client-Hello-Extension), then the max. AFL Message length is lower.

5.2.5 AFL Message Counter Field (AFL.MCR)

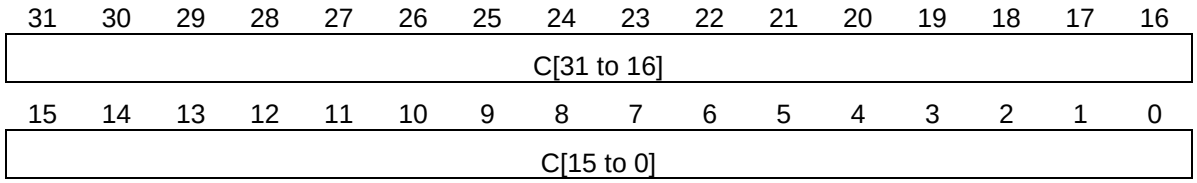


Figure 5 — AFL Message Counter Field bitmap

Table 8 — AFL Message Counter Field bitfield definitions

Bits	Field Name	Description
31 to 0	C	A 32-Bit Field, which is strictly monotonously increasing. The counter is incremented on every message-transmission. The counter never wraps. The initial value after manufacturing is 1.

For conformance with [BSI1A] the AFL.MCR shall be present in messages from meters which use unidirectional communication. See 5.5.4 for details on the requirements for the counter.

If the Counter Field is used, the AFL.MCR field shall always be present in the first fragment. It shall not be present in any following fragments of the same message.

5.2.6 AFL MAC Field (AFL.MAC)

The length of the MAC field depends on the selected option AFL.MCL.ATO indicated by the AFL.MCL field.

If the MAC Field is used, the AFL.MAC field shall only be present in the last fragment of a message.

For conformance with [BSI1A] the AFL.MAC shall be present in messages from meters which use unidirectional communication.

5.3 AFL with the Extended Link Layer (ELL)

If the Authentication and Fragmentation Layer on wireless M-Bus [EN-4] is used then the Extended Link layer shall be applied. For fragmented messages the master shall send all fragments with CI=0x8E (ELL with address). According to [EN-4], this Layer provides the link address of the receiver, the access number for each data gram and the Hop count bit for a potential repeater. When no fragmentation is needed (e.g. AFL used for Authentication) the master may use CI=0x8C (ELL without address) or CI=0x8E (ELL with address). The meter should always use CI=0x8C (ELL without address). The AFL Header follows the ELL header.

CI=0x8C	CC	ACC

Figure 6 – ELL without address

CI=0x8E	CC	ACC	MFCT_0	MFCT_1	ID_0	ID_1	ID_2	ID_3	VER	DEVTYPE

Figure 7 - ELL with address

5.4 The MAC-Generation

The authentication of the message is supported by the AFL using the MAC. This MAC shall be calculated as specified in AES128 for Crypto-Message-Authentication (CMAC-AES128) according to [RFC4493]. The MAC shall be calculated as follows:

AFL.MAC = CMAC (Kmac/Lmac, AFL.MCL || AFL.MCR[7..0] || AFL.MCR[15..8] || AFL.MCR[23..16] || AFL.MCR[31..24] || { AFL.ML[7..0] || AFL.ML[15..8] || } NextCI || ... || Last Byte of Message)

The presence of the AFL.ML field depends on the selection bits in the AFL.MCL field.

An example is given in 0.

For a transmission from gateway to meter the key Lmac is used, for a transmission from meter to gateway the key Kmac is used (see key derivation in chapter 5.5.7).

The 16 byte result of this CMAC-function shall be truncated to 8 byte as defined in [RFC4493].

In deviation to the usual transmission order for octet strings on the M-Bus, the MSB of the MAC shall be transmitted as first byte, the LSB as last.

5.5 The Key Derivation Function (KDF)

5.5.1 General

The MAC generation (as well as the encryption of the payload) shall be based on an ephemeral key, which is used for one message only. The ephemeral key shall be generated using the Key Derivation Function defined below. The Key Derivation Function shall also apply to the CMAC-Function according to [RFC4493]. There are 5 input values to the KDF specified in 5.5.2 to 5.5.6.

5.5.2 Individual Master Key - MK

Before each transmission two ephemeral keys Kenc (for encryption) and Kmac (for Authentication) are derived from the individual Master Key MK. There are two sets of key pairs (one set for the meter Kenc/Kmac and one set for the gateway Lenc/Lmac).

5.5.3 Derivation Constant – D

The constant is used to derive different Keys for both Encryption and Authentication as well as for the two directions - from and to the meter.

Table 9 – Constant D for the key derivation

D	Used for
0x00	Encryption from the meter (Kenc)
0x01	MAC from the meter (Kmac)
0x10	Encryption from the gateway (Lenc)
0x11	MAC from the gateway (Lmac)

5.5.4 Counter – C/C'

The changing secrets are generated by inclusion of a strictly monotonously increasing (non-secret) counter in the KDF. This counter is transmitted in the AFL.MCR field (see 5.2.5). The counter maintained by the meter is named C, the counter maintained by the Other Device is named C'. The generation of C' is based on Counter C by adding an increment. The increment step shall not exceed the value of 100.

NOTE: Because of the short time window (2ms) for replying in wireless M-Bus Mode T a gateway may calculate the Encryption and MAC Keys in advance, based on the assumption of an Counter Value (C') depending on the number of lost fragments.

5.5.5 Meter-ID

The usage of the meter identification prevents a parallel Key generation by different meters.

For messages from the meter to the gateway which uses a short header, (like CI=7Ah; see [EN-3]) the ID_0 to ID_3 corresponds to the LSB to MSB of the Link Layer Identification number of the meter. For messages with Long-header (like CI=72h; see [EN-3]) the ID_0 to ID_3 corresponds to LSB to MSB of the Application Layer Identification number of the meter.

For messages from the gateway to the meter always the Long header is used. The ID_0 to ID_3 corresponds to the LSB to MSB of the Application Layer Identification number (address) of the meter (not the gateway!).

5.5.6 Padding

To avoid the generation of the K2 (refer to [RFC4493]) in the KDF, the remaining bytes of the 16 byte block are filled with a padding sequence. For the generation of Kmac, Lmac and Kenc, Lenc the padding is fixed and consists of seven octets each containing the value of 0x07 according to the rule that the input to the MAC shall be padded with *0x(16-l mod 16)* bytes with value *0x(16-l mod 16)*, where *l* equals the byte length of the input[3].

5.5.7 Key calculation

The calculation of Kenc and Kmac for the meter:

```
Kenc = CMAC(MK, 0x00 ||C[7..0] ||C[15..8] ||C[23..16] ||C[31..24]
||ID_0||ID_1||ID_2||ID_3
||0x07||0x07||0x07||0x07||0x07||0x07||0x07)
```

[3] NOTE: In short: The value of the padding bytes contains the number of added padding bytes.

```
Kmac = CMAC(MK, 0x01 || C[7..0] ||C[15..8] ||C[23..16] ||C[31..24]
|| ID_0||ID_1||ID_2||ID_3
||0x07||0x07||0x07||0x07||0x07||0x07||0x07)
```

Where C[7..0] is the LSB and C[31..24] is the MSB (Big Endian) of the counter AFL.MCR.C from meter to other (gateway).

The gateway shall use a higher value C' than the last AFL.MCR.C received from the meter for the next message to the meter:

```
Lenc = CMAC(MK, 0x10 || C'[7..0] ||C'[15..8] ||C'[23..16] ||
C'[31..24] || ID_0 || ID_1 || ID_2 || ID_3 ||
0x07||0x07||0x07||0x07||0x07||0x07||0x07)
```

```
Lmac = CMAC(MK, 0x11 || C'[7..0] ||C'[15..8] ||C'[23..16] ||
C'[31..24]  || ID_0 || ID_1 || ID_2 || ID_3 ||
0x07||0x07||0x07||0x07||0x07||0x07||0x07)
```

Where C'[7..0] is the LSB and C'[31..24] is the MSB (Big Endian) of the counter AFL.MCR.C from other (gateway) to meter.

Sequence for meter:

1. Increment C by one, Store C.
2. Use C stored in the meter for derivation of next Kenc and Kmac
3. Generate Message with AFL.MCR.C=C
4. Encrypt Message with Kenc
5. Authenticate Message with Kmac
6. Transmit Message to gateway

Sequence for gateway:

1. Use last received C from the meter (stored as C_{Rx} in the gateway) to generate incremented C'
2. Use C' for derivation of next Lenc and Lmac
3. Generate Message with AFL.MCR.C=C'
4. Encrypt Message with Lenc
5. Authenticate Message with Lmac
6. Transmit Message to meter

See Annex E for an example of Message Counter handling.

NOTE: It should be noted that the TLS (transport layer security) described in this document is independent from the KDF and CMAC, but the AFL offers the reliable transport and protection against DoS attacks for the TLS.

6 AES128 Encryption Mode (Transport Layer)

For conformance with [BSI1], Encryption Mode 7 (0x07) uses AES-CBC with Key length 128 Bits and initialisation vector IV=0 (16 Bytes of 0x00). For the generation of the Key the KDF shall be applied as described in clause 5.5. The IV is implicit and cannot be modified.

NOTE: An implicit static initialisation vector can be used, because for each message a new key is derived.

The 3 byte configuration field (CF) to be used for Mode 7 is defined as follows:

Table 10 – Configuration field of Encryption Mode 7

Bit 23	Bit 22	Bit 21	Bit 20	Bit 19	Bit 18	Bit 17	Bit 16	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reserved	Reserved for Dynamic key	Dynamic key	Dynamic key	Reserved for Key-ID	Key-ID	Key-ID	Key-ID	Content of Message	Content of Message	Reserved for future field	Mode 4	Mode 3	Mode 2	Mode 1	Mode 0	number of encr. blocks	number of encr. blocks	number of encr. blocks	number of encr. blocks	Reserved	Reserved	Reserved	Reserved
R	0	D	D	0	K	K	K	C	C	0	M	M	M	M	M	N	N	N	N	0	0	0	0

M shall always be 7 (0x07) to mark AES128 with CBC and dynamic Key

K selects the Key Index for Encryption. For conformance with [BSI1], only the use of K=0 (MasterKey) is allowed.

D shall always be 1 (0x01) to mark Key Derivation Function as defined in clause 5.5.

C declares the Content of Message and shall conform to [OMSV2].

N contains the number of encrypted 16Byte Blocks for CBC Mode.

A two byte sequence 0x2F, 0x2F (Decryption verification) shall immediately follow the Configuration Field. The Encryption verification field is not part of the Application Layer.

7 TLS-Mode (Transport Layer)

7.1 Required TLS Operation in the scope of this Document

The encryption of application data using TLS provides a higher security level than the encryption based on AES-128. For transmissions where AES-based encryption with shared keys is not sufficient (see [BSI1]) the Transport Layer Security shall be applied as defined in [RFC5246].

To be able to implement the TLS protocol as defined in [RFC5246] in an environment with such limited networking resources as (Wireless-) M-Bus, the following limitations and extensions are required.

- Meters with wM-BUS [EN-4] should implement the TLS-Client functionality; gateways should implement the TLS-Server functionality.
- TLS1.2 shall be supported by client and server. In the future also higher TLS Versions should be supported.
- Servers shall support the client_hello max_fragment_length Extension (see [RFC6066]) with min. 512 bytes fragment size.
- Servers shall support the client_hello truncated_hmac Extension (see [RFC6066]) to send the first 10 bytes of the negotiated HMAC and verify the first 10 bytes of the calculated hash.
- Clients shall accept the additional ConnectionRequest Message. This message shall be authenticated with at least 8 bytes of AES-128-CMAC protection.
- Session renegotiation is not allowed.
- A session can be resumed. This document allows a max. Session lifetime of 1 month (31 days) unless not more than 5 Mbytes of TLS data (Record Layer bytes) is transmitted in both directions within this session. Session resumption shall be implemented in the gateway.
- A new (authenticated) ConnectionRequest from gateway to meter shall close any existing TLS session between meter and sender (gateway) and establishes (resumes or negotiates) a new Session.
- For TLS Crypto suites based on Elliptical Curves, the uncompressed Point format shall be used.

TLS Crypto suites based on Elliptical Curves shall be implemented according to [RFC4492] or [RFC5289].

NOTE: It is intended that a future version may also supports the Brainpool curves as referred in [BSI3].

Table 11 – List of supported elliptical curves

Curve name	TLS NamedCurveID
brainpoolP192r1	TBD
brainpoolP256r1	TBD
brainpoolP384r1	TBD
brainpoolP512r1	TBD
Secp192r1/NIST P192	0x0013
Secp256r1/NIST P256	0x0017
Secp384r1/NIST P384	0x0018

7.2 Transport Layer Encryption Configuration Field

The TLS Encryption Mode is declared with a 13 (0x0D) in the Mode bits. NOTE: Bits for link control are served by the extended link layer (ELL), which has always been applied together with TLS.

The 3 Byte configuration field for the Mode 13 is defined in Table 12:

Table 12 – Configuration field of Encryption Mode 13

Bit 23	Bit 22	Bit 21	Bit 20	Bit 19	Bit 18	Bit 17	Bit 16	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reserved	Reserved	Reserved	Reserved	Protocol Type 3	Protocol Type 2	Protocol Type 1	Protocol Type 0	Content of Message	Content of Message	Reserved	Mode 4	Mode 3	Mode 2	Mode 1	Mode 0	Number of encrypted Bytes	Number of encrypted Bytes	Number of encrypted Bytes	Number of encrypted Bytes	Number of encrypted Bytes	Number of encrypted Bytes	Number of encrypted Bytes	Number of encrypted Bytes
0	0	0	0	P	P	P	P	C	C	0	M	M	M	M	M	N	N	N	N	N	N	N	N

M shall always be 13 (0x0D) to declare an Encryption with TLS

NOTE: The applied TLS-Version has to traced from TLS-Header

C declares the Content of Message and shall conform to [OMSV2].

N Number of encrypted bytes. N indicates the number of bytes following the Configuration Field which are covered by the Protocol indictated by Protocol type P (TLS). N is limited to 255.

NOTE: For larger sizes the exact number of bytes (minus the TLS header size 5 Bytes) can be found in the 4th and 5th Byte of the TLS header.

P is the Protocol Type as defined in Table 13

Table 13 – Protocol types of Encryption Mode 13

Content of Protocol Type P	Meaning
0000	TLSCONN (Connection Request)
0001	TLSHS (TLS1.2 Handshake Record Layer)
0010	TLSALERT (TLS1.2 Alert Record)
0011	TLSAPP (TLS1.2 Application Record)
0100..1111	Reserved

NOTE: No Decryption check bytes follow the CF in this Encryption Mode!.

Messages with Protocol Type TLSAPP shall not be protected by an additional AFL MAC.

7.3 TLS-Handshake

7.3.1 TLS-Connection establishment and preventing Client Hello Flooding

The TLS Session starts, when the Client sends a TLS ClientHello Handshake Message to the Server (TLS 1st MsgBlock). The Server shall generate a Random Number and sends a (fragmented) TLS message with typically more than 400 bytes message length (session not resumed).

Most EN 13757 Channels are limited in bandwidth and/or duty cycle limited. Sending ClientHellos will rapidly reduce the transmit credits of the sender or fill the physical medium. This is true both for the master and the slave M-Bus device (gateway and meter). The Sender may limit its transmission rate to prevent a DoS Attack to the bandwidth limited medium.

Another problem is the (battery-)power consumed by calculation of the Random-Number and TLS Pre-Master-Secret, triggered by unnecessary ClientHellos.

To take measures against these problems the TLS-Role „Client“ is assigned to the meter and the TLS-Role „Server“ is assigned to the gateway.

Because a TLS-protected connection is requested by the gateway, a ConnectionRequest message is used to initiate the TLS-Connection establishment from the meter. At minimum the first message flight from Client to Server and Server to Client should be MAC-protected and verified (i.E. ConnectionRequest, ClientHello).

Any existing TLS-session from the gateway to the meter shall be silently closed in the meter. I.e. all temporary key material associated with this previous TLS session shall be erased. A session shall be identified by the gateways APL/TPL address ADDR (consists of ID, Version, Device-Type)

7.3.2 TLS Handshake Header for EN 13757

CI-TCxx (Transport Control)	[ADDR,]ACC,ST,CF(EM-TLS;TLSHS)	TLS-MsgType (0x14,0x15, 0x16)	TLS-Proto Major (0x03)	TLS-Proto Minor (0x03)	TLS-MsgLen MSB	TLS-MsgLen LSB	TLS-Msg Data
[Optional unencrypted Appl.Data]							

The 5 bytes MsgType, ProtoMajor, ProtoMinor, MsgLen, are defined in TLS1.2 (RFC 5246).

7.3.3 TLS Session initiated by gateway, meter is a TLS Client

CI=TCOL (Transport Control)	[ADDR,] ACC, ST, CF(EM-TLS,TLSCONN)	Connection Req (0x00)	Reserved (0x00)	Reserved (0x00)	MsgLen MSB (0x00)	MsgLen LSB (0x00)

5

Gateway (LLA=COM)	Message flight	Meter (LLA=MTR)
Request Connection to Meter.		
Optional immediate REQ_UD2 (for Meters with Receiver always on) with ConnectionRequest	SND-UD(C=53);CI=AFL;CMAC; CI=TC;MTR;ACC=1;CF(EM-TLS, TLSCONN) →	Since the receiver is not always open, the message is not received.
	← SND-NR(C=44); CI=7A;ACC=92	Meter sends periodic, encrypted data to gateway. Payload AES-encrypted and opens receiver after 2/3 ms.
REQ_UD2 with ConnectionRequest to enter frequent access mode with TLS connection	SND_UD(C=53);CI=AFL;CMAC; CI=TC;MTR;ACC=1;CF(EM-TLS, TLSCONN) → ← ACK(C=00);CI=8A;ACC=1	The meter enters frequent access mode and generates a new TLS session (random, sessionID). *Waits for REQ_UD2 from gateway to send ClientHello*

The Gateway requests to receive	REQ_UD2(C=7B);CI=TC;MTR;ACC=2;CF(EM-TLS; TLSHS) → RSP_UD(C=08);CI=TC;ACC=2 ← Flight 1	Meter sends ClientHello (with or without SessionID)
Gateway sends ServerHello without SessionID, Certificate, ServerKeyExchange, CertificateRequest and ServerHelloDone in fragmented messages.	SND_UD(C=53);CI=8E;ACC=3;CI=AFL;FragID=1; MsgLen;CI=TC;CF(EM-TLS ;TLSHS) → ACK(C=00);CI=8A;ACC=3 ← SND_UD(C=73);CI=8E;ACC=4;CI=AFL;FragID=2; → ACK(C=00);CI=8A;ACC=4 ← SND_UD(C=53);CI=8E;ACC=5;CI=AFL;FragID=3; → ACK (C=08);CI=8A;ACC=5 ← Flight 2	
Gateway requests to receive	REQ_UD2(C=7B);CI=TC;ACC=6;CF=(EM-TLS;TLSHS) → RSP_UD(C=08);CI=ELL;ACC=6;CI=AFL;FragID=1 MsgLen;CI=TC;CF(EM-TLS;TLSHS) ← REQ_UD2(C=5B);CI=TC;ACC=7;CF=TLS → RSP_UD(C=08);CI=ELL;ACC=7;CI=AFL;FragID=2 ← REQ_UD2(C=7B);CI=TC;ACC=8;CF=TLS → RSP_UD(C=08);CI=ELL;ACC=8;CI=AFL;FragID=3 ← REQ_UD2(C=5B);CI=TC;ACC=9;CF=TLS → RSP_UD(C=08);CI=ELL;ACC=9;CI=AFL;FragID=4 ← Flight 3	Meter Sends Certificate, ClientKeyExchange, CertificateVerify ChangeCipherSpec, Finished
Gateway sends ChangeCipherSpec, Finished.	SND_UD(C=73);CI=TC;ACC=10;CF(EM-TLS;TLSHS) → ACK(C=00);CI=8A;ACC=10 ← Flight 4	

All numbers are hexadecimal!

7.3.4 Resumed TLS Session initiated by Gateway, Meter is TLS Client

Gateway (LLA=COM)	Message flight	Meter (LLA=MTR)
Request Connection to Meter.		
Optional immediate SND_UD (for Meters with Receiver always on) The Gateway send ConnectionRequest	SND-UD(C=73);CI=ELL;ACC=1;CI=AFL;CMAC; CI=TC;CF(EM-TLS,TLSCONN) →	Since the receiver is not always open, the message is not received.
	← SND-NR(C=44);AH ;CI=7A;ACC=92	Meter sends periodic, encrypted data to gateway. Payload AES-encrypted and opens receiver after 2/3 ms.
SND_UD to enter frequent access mode with TLS connection another ConnectionRequest	SND-UD(C=73);CI=ELL;ACC=1;CI=AFL;CMAC; CI=TC;CF(EM-TLS,TLSCONN) → ← ACK(C=00);CI=8A;ACC=1	The meter enters frequent access mode and generates a new TLS session (random, sessionID). *Waits for REQ_UD2 from gateway to send ClientHello*
	REQ_UD2(C=5B); → ← RSP_UD(C=08);CI=TC;ACC=2;CF(EM-TLS,TLSHS) Flight 1	Meter sends ClientHello with SessionID
Gateway sends ServerHello with SessionID and ChangeCipherSpec, Finished in one message.	SND_UD(C=73);CI=ELL;ACC=3; CI=AFL;FragID=1;MsgLen; CI=TC;ACC=3;CF(EM-TLS,TLSHS) → ← ACK(C=00);CI=8A;ACC=3 Flight 2	

Gateway requests to receive	REQ_UD2(C=5B); RSP_UD(C=08);CI=TC;ACC=4;CF(EM-TLS;TLSHS) Flight 3	Meter sends ChangeCipherSpec, Finished

All numbers are hexadecimal!

In case of session resumption, the server shall ignore the Supported Elliptic Curves Extension and the Supported Point Formats Extension appearing in the current ClientHello message.

7.3.5 Terminate session

Because the Session may be silently terminated (closed) by either Client or Server (for example with a reboot and power-loss) the Client (e.g. meter) shall accept an (authenticated) ConnectionRequest to close an existing session to the sender of the ConnectionRequest and initiate a new TLS Session by sending a ClientHello to the sender of the ConnectionRequest.

The wireless M-Bus does not define a connection oriented session layer. Therefore termination of the TLS session cannot close the underlying session layer. In case a Frequent Access Mode (Link Layer) is active, the TLS Layer may terminate the Frequent Access Mode by initiating an SND_NKE.

Session Terminated by gateway :

Gateway (LLA=COM)	Message flight	Meter (LLA=MTR)
Sends TLS CloseNotify	SND_UD(C=73);CI=TC; ACC=10;CF(EM-TLS;TLSALERT) ACK(C=00);CI=8A;ACC010; Flight n-3	
Requests close notify from Meter	REQ_UD2(C=5B);CI=TC;ACC=4;CF(EM-TLS;TLSALERT) RSP_UD(C=08);CI=TC;CF(EM-TLS;TLSALERT) Flight n-2	Send TLS CloseNotify
Terminate Frequent Access Mode	SND_NKE(C=40) Flight n-1	

7.3.6 Exchange of Application Data

Gateway (LLA=COM)	Message flight	Meter (LLA=MTR)
Gateway (LLA=COM)	Message flight	Meter (LLA=MTR)
TLS-Encapsulated Request	SND_UD(C=73);CI=8C(ELL), CI=72;ACC=12;CF(EM-TLS ;TLSAPP) → ACK(C=00);CI=8A;CF=0;ACC=12 ← Flight x	
	Or REQ_UD2(C=53);CI=8C(ELL), CI=TC;ACC=13 → RSP_UD(C=08);CI=ELL,CI=AFL,CI=72;ACC=13 ;CF(EM-TLS ;TLSAPP) ← Flight y	Send TLS Encapsulated Response

All numbers are hexadecimal!

7.3.6.1 TLS Application Header for EN 13757-2/-4

CI=5B/60/64..App Data with LongHeader	[ADDR,] ACC,ST, CF(EM-TLS,TLSAPP)	TLS-MsgType (0x17) Application)	TLS-Proto Major (0x03)	TLS-Proto Minor (0x03)	TLS-MsgLen	TLS-MsgLen LSB	TLS-Random IV (16 Bytes)
Encrypted App.Data	TLS-HMAC (32 bytes [a])	TLS-Padding(1..16 bytes)	[Optional unencrypted Appl. Data]				
[a] In case of truncated HMAC the length will be 10 bytes only.							

8 Update of the individual Master Key MK

Both meter and gateway hold a (symmetric) individual Master Key MK used for the key derivation of session keys Kmac, Kenc and Lmac, Lenc to encrypt/decrypt and authenticate/verify the messages between meter and gateway. To update the Master Key MK to a new Master Key MK', a TLS connection between meter and gateway shall be established. After the key exchange the new key has to be verified. Additional the gateway needs to check if the counter C reset after a successful key exchange.

The procedure for an update of the Master Key is described in a future version of this document.

9 Update of the individual TLS certificate

The meter and the gateway have to support commands to exchange their certificates and store them for mutual direct trust during the pairing process using the AES-MAC and AES-Encryption. The command sequence will be described in a future revision of this document.

Annex A. (informative) - Examples for the usage of AFL and TLS

This chapter shows several examples of messages using AFL and TLS. All CI fields in hex.

The gray-shaded fields indicate the new AFL or TLS-elements.

Table 14 — Non Secured unfragmented M-Bus-Message

DLL Header (EN 13757-4): Len,MsgType Addr	Long/Short Transport App Header CI=5B/7A/72 (ADDR,)ACC,ST,CF	Application Data EN 13757-3 MBUS DIF/VIFs

Table 15 — Non-Fragmented Authenticated M-Bus Message

DLL Header (EN 13757-4): Len,MsgType Addr	Extended Link layer CI-ELL (ADDR),CC,ACC [EN-4]	Authentication and Fragmentation Layer CI-AFL AFLLen, AFLFlags, CTR, MAC	Long/Short Transport Header CI=5A/5B/72/7A (ADDR),ACC,ST,CF [EN-4]/[OMSV2]	Application Data [EN-3]

Table 16 — Fragmented Authenticated Application Message (2 fragments)

DLL Header (EN 13757-4): Len,MsgType, Addr	Extended Link layer CI-ELL (8E) CC,ACC,M2A2 [EN-4]	Authentication and Fragmentation Layer CI-AFL AFLLen, AFLFlags(MF=1), MsgLen,CTR	Long/Short Transport Header CI=5A/5B/72/7A/7E/7F (ADDR,)ACC,ST,CF [EN-4]/[OMSV2]	Application Data [EN-3]

DLL Header (EN 13757-4): Len,MsgType, Addr	Extended Link layer CI-ELL (8E) CC,ACC,M2A2 [EN-4]	Authentication and Fragmentation Header CI-AFL AFLLen, AFLFlags(MF=0), MAC	Application Data [EN-3]

Table 17 — Unfragmented TLS secured Application Command Message

DLL Header (EN 13757-4): Len,MsgType, Addr	Long/Short Transport Header CI=5A/5B (example) (ADDR,) ACC,ST, CF(EM- TLS) [EN-4]/[OMSV2]	TLS Record (includes Application data)		
		TLS-HDR,IV	M-Bus/Application data	HMAC/Padding

Annex B. (informative): Example of a CMAC-calculation

Table 18 — Example (AFL with unencrypted payload)

	Byte 0	Byte 1	Byte 2	Byte 3
DLL	L=0x1E	C=0x44 (SND_NR)	MFCT[7..0]	MFCT[15..8}
0x04	ID_0=0x78	ID_1=0x56	ID_2=0x34	ID_3=0x12
0x08	VER	DEVTYPE	CI=ELL(0x8C)	CC
0x0C	ACC	CI=AFL(90)	AFL.AFLL=0x0F	AFL.FCL[7..0] 0x01
0x10	AFL.FCL[15..8] 0x2C	AFL.MCL 0x25	AFL.MCR[7..0] 0x01	AFL.MCR[15..8] 0x04
0x14	AFL.MCR[23..16] 0x00	AFL.MCR[31..24] 0x00	CI=0x7A	ACC(0x11)
0x18	ST (0x00)	CF[7..0] 0x00	CF[15..8] 0x07	CF[16..23] 0x00
0x1C	DIF 0x01	VIF 0x5B (Temperature)	DATA 0x19 (25°)	

Following fields are used as input for the MAC-Calculation:

Individual Master Key MK = 0x00 0x11 0x22 0x33 0x44 0x55 0x66 0x77 0x88 0x99 0xAA 0xBB 0xCC 0xDD 0xEE 0xFF

AFL.MCL (Message Control)

AFL.MCR (Message counter C)=1025 (0x01 0x04 0x00 0x00 LSB first - before transmission)

NOTE: Thus AFL.ML (Message Length) is not present; it is not applied for this CMAC calculation

{CMAC Values below were calculated with AES128-CMAC according to [RFC4493]}

```
MAC=CMAC (Key, Message)
Kenc=CMAC (0x00||0x11||0x22||0x33||0x44||0x55||0x66||0x77||
0x88||0x99||0xAA||0xBB||0xCC||0xDD||0xEE||0xFF, -- Individual Master
Key
0x00|| -- Modifier Kenc
0x01||0x04||0x00||0x00|| -- Counter C
0x78||0x56||0x34||0x12|| -- ID
 0x07||0x07||0x07||0x07||0x07||0x07||0x07 -- Padding
)=
Kmac=CMAC (0x00||0x11||0x22||0x33||0x44||0x55||0x66||0x77||
0x88||0x99||0xAA||0xBB||0xCC||0xDD||0xEE||0xFF,
0x01|| -- Modifier Kmac
0x01||0x04||0x00||0x00|| -- Counter C
0x78||0x56||0x34||0x12|| -- ID
0x07||0x07||0x07||0x07||0x07||0x07||0x07 -- Padding
)=
```

```
MAC=CMAC (Kmac,
0x25|| -- AFL.MCL
0x01||0x04||0x00||0x00|| -- AFL.MCR

0x7A|| -- CI Short MBUS Data
0x11|| -- ACC
0x00|| -- ST
0x00||0x00|| -- CF (Bits 0/1 reset to zero)
0x01||0x5B||0x19 – DIF/VIF/Value for Temperature
)=
```

For MACLength=8 bytes the MAC in the AFLHeader is truncated to the first 8 bytes (most significant bytes according to RFC 4493) MAC = ...

Annex C. (informative): Message examples of TLS

Following colors are used:

Green background color for HandshakeMsg Types

Orange background color to present the Payload (the application protocol itself)

Grey Background colors for optional fields which may be not present in certain circumstances.

All Examples show with no radio adapter on the meter (i.e. for transmission the meter sents its address in the Link Layer)

C.1 Session Initiation - ConnectionRequest (from Server to Client)

Layer	0 (First Byte)	1	2	3
DLL	L=nn	C=53 (SND_UD)	MFCT0	MFCT1
DLL	ID_0(GW)	ID_1(GW)	ID_2(GW)	ID_3(GW)
DLL	VER(GW)	DEVTYPE(31)		
ELL	CI=8C(ELL)	CC	ACC	CI=90(AFL)
AFL	AFL.AFLL (0x00F)	AFL.FCL(0x01) LSB	AFL.FCL(0x2C) MSB MF=0,CTR,MCL,MAC	AFL.MCL(0x25), CMAC8,CTR
AFL	AFL.MCR[7..0]	AFL.MCR[15..8]	AFL.MCR[23..16]	AFL.MCR[31..24]
AFL	AFL.MAC	AFL.MAC	AFL.MAC	AFL.MAC
AFL	AFL.MAC	AFL.MAC	AFL.MAC	AFL.MAC
TPL	CI=TCL(Transport Control Long)	ID_0(Meter)	ID_1(Meter)	ID_2(Meter)
TPL	ID_3(Meter)	MFCT0(Meter)	MFCT1(Meter)	VER(Meter)
TPL	DEVTYPE(Meter)	ACC	ST	CF[7..0]
TPL	CF[15..8](EM-TLS)	CF[23..16] (TLSCONN)		
0x00	Connection MsgType (0x00)	Reserved (0x00)	Reserved (0x00)	Connection-MsgLen MSB (0x00)
0x04	Connection MsgLen (0x00)			

The meter answers with ACK (no data) and waits for an REQ_UD2 from the gateway.

C.2 First message flight - ClientHello (from Client to Server)

Offset	0 (First Byte)	1	2	3
DLL	L=nn	C=08/28 (RSP_UD)	MFCT0(MTR)	MFCT1(MTR)
DLL	ID_0(MTR)	ID_1(MTR)	ID_2(MTR)	ID_3(MTR)
DLL	VER(MTR)	DEVTYPE(MTR)		
ELL	CI=8C(ELL)	CC	ACC	CI=AFL
AFL	AFL.AFFL(0x0F)	AFL.FCL(0x01) LSB	AFL.FCL(0x2C)MSB MF=0,CTR,MCL,MAC	AFL.MCL(0x25), CMAC8,CTR
AFL	AFL.MCR[7..0]	AFL.MCR[15..8]	AFL.MCR[23..16]	AFL.MCR[31..24]
AFL	AFL.MAC	AFL.MAC	AFL.MAC	AFL.MAC
AFL	AFL.MAC	AFL.MAC	AFL.MAC	AFL.MAC
TPL	CI=TCS(Transport Control Short)	ACC	STS	CF[7..0]
TPL	CF[15..8] (EM-TLS,)	CF[23..16] (TLSHS)		
0x00	MsgType (0x16, Handshake)	ProtoMajor (0x03) TLS1.2	ProtoMinor(0x03) TLS1.2	TLS-FragmentLen(MSB, 0x00)
0x04	TLS-FragmentLen(LSB, 0xll1)	Handshake-Type (0x01 ClientHello)	HS-Record-Len(MSB, 0x00)	HS-Record-Len (0x00)
0x08	HS-Record-Len(LSB,0xl2)	ProtoMajor(0x03)	ProtoMinor(0x03) TLS1.2	UTC-Unix-Time(MSB)
0x0C	UTC-Unix-Time	UTC-Unix-Time	UTC-Unix-Time(LSB)	ClientRandom 28
0x10	Random 27	Random	Random	Random
0x14	Random 23	Random	Random	Random
0x18	Random 19	Random	Random	Random
0x1C	Random 15	Random	Random	Random
0x20	Random 11	Random	Random	Random
0x24	Random 7	Random	Random	Random
0x28	Random 3	Random 2	ClientRandom 1	SessionIDLength(0x00 No Resume)[4]
0x2C	CipherSuitesLength (MSB, 0x00)	CipherSuitesLength (LSB, 0x08)	CipherSuite Prio1 (MSB, 0xC0) TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384	CipherSuitePrio1(LSB, 0x24) TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384

[4] NOTE: If no session resumption is requested, the Session ID length is 0x00. With Session Resumption the maximum length is 32 Bytes.

0x30	CipherSuitePrio2(MSB, 0xC0) TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256	CipherSuitePrio2(LSB, 0x23) TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256	CipherSuitePrio3(MSB, 0xC0) *TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256*	CipherSuitePrio3(LSB, 0x2B) *TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256*
0x34	CipherSuitePrio4(MSB, 0xC0) *TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384*	CipherSuitePrio4(LSB, 0x2C) *TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384*	CompressionMethodsLength (0x01)	NULL Compression (0x00)
0x38	ExtensionsLength (MSB, 0x00)	ExtensionsLength(LSB,0x08)	Extension elliptical_curves (MSB,0x00)	Extension elliptical_curves (LSB, 0x0A)
0x3C	Elliptical_curves_length(MSB,0x00)	Elliptical_curves_length(LSB,0x0A)	Secp256r1 (NIST P-256) MSB (0x00)	Secp256r1(NIST P256) LSB. (0x17)
0x40	BrainpoolP256r1 MSB (0xFE)	BrainpoolP256r1LSB (0xnn)	BrainpoolP384r1 MSB (0xFE)	BrainpoolP384r1 LSB (0xnn)
0x44	BrainpoolP512r MSB (0xFE)	BrainpoolP512r1 LSB (0xnn)	Secp384r1(NIST P-384) MSB (0x00)	Secp384r1(NIST P-384) MSB(0x18)
0x48	Extension truncated_hmac (MSB, 0x00)	Extension truncated_hmac (LSB, 0x04)	ExtensionData_length (MSB 0x00)	ExtensionData_len (LSB 0x00))
0x4C	Extension max.fragment_length (MSB 0x00)	Extension max.fragment_length (LSB 0x01)	Extension length (MSB 0x00)	Extension Length (0x01)
0x50	Extension Data (0x02) Max.Fragment Size 1 kbyte			

NOTE 1: The Ciphersuites in the example above use the NIST P-256 .

NOTE 2: The default FragmentSize Limit in TLS is 16 kbytes (which shall be buffered by the receiver and the transmitter). For memory size restricted Devices a max_fragment_length Extension can be negotiated, in case also the Client supports it. The negotiation is defined in RFC6066.

NOTE 3: TLS1.2 suggests but does not require a limitation of the (resumable) TLS Session lifetime. For security reasons (for example the derived ephemeral key material may be compromised by memory-reads) the TLS session life time shall be limited.

NOTE 4: A session renegotiation is not necessary and shall not be allowed. With nearly the same number of handshake messages and data, the existing session can be closed and a new session can be established.

NOTE 5: After termination of the session (in an orderly manner the gateway should send a CloseNotify message to the meter), the meter may immediately start a new TLS session by sending a ClientHello to the gateway.

C.3 2nd message flight ServerHello, Certificate, ServerKeyExchange, CertificateRequest, ServerHelloDone (from Server to Client)

C.3.1 First TLS fragment

Offset	0 (First Byte)	1	2	3
DLL	L=nn	C=53/73 (SND_UD)	MFCT0(GW)	MFCT1(GW)
DLL	ID_0(GW)	ID_1(GW)	ID_2(GW)	ID_3(GW)
DLL	VER(GW)	DEVTYPE(31)		
ELL	CI=8C(ELL)	CC	ACC	CI=90(AFL)
AFL	AFL.AFFL(0x11)	AFL.FCL(0x01) LSB	AFL.FCL(0x7C)MSB MF=1,CTR,MCL,MAC,ML	AFL.MCL(0x65), CMAC8,CTR,ML
AFL	AFL.MCR[7..0]	AFL.MCR[15..8]	AFL.MCR[23..8]	AFL.MCR[31..8]
AFL	AFL.MAC	AFL.MAC	AFL.MAC	AFL.MAC
AFL	AFL.MAC	AFL.MAC	AFL.MAC	AFL.MAC
AFL/ TPL	AFL.ML (0x8F, LSB)	AFL.ML (0x01, MSB)	CI=TCL (Transport Control Long)	ID_0(MTR)
TPL	ID_1(MTR)	ID_2(MTR)	ID_3(MTR)	MFCT0(MTR)
TPL	MFCT1(MTR)	VER(MTR)	DEVTYPE(MTR)	ACC
TPL	STS	CF[7..0]	CF[15..8](EM-TLS)	CF[23..16](TLSHS)
0x00	MsgType (0x16, Handshake)	ProtoMajor(0x03)	ProtoMinor(0x03) TLS1.2	MsgFragmentLenMSB, 0x01)
0x04	MsgFragmentLeLSB, 0x85)	Handshake-Type (0x02 ServerHello)	HS-Record-Len(MSB, 0x00)	HS-Record-Len (0x00)
0x08	HS-Record-Len(LSB 0x26)	ProtoMajor(0x03)	ProtoMinor(0x03) TLS1.2	UTC-Unix-Time(MSB)
0x0C	UTC-Unix-Time	UTC-Unix-Time	UTC-Unix-Time(LSB)	ServerRandom 28
0x10	Random 27	Random	Random	Random
0x14	Random 23	Random	Random	Random
0x18	Random 19	Random	Random	Random
0x1C	Random 15	Random	Random	Random
0x20	Random 11	Random	Random	Random
0x24	Random 7	Random	Random	Random

0x28	Random 3	Random 2	ServerRandom 1	SessionIDLength(0x00)[5]
0x2C	CipherSuite (MSB, 0xC0) TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256	CipherSuite (LSB, 0x23) TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256	CompressionSelected (NONE, 0x00)	Handshake-Type (0x0B) Certificate
0x30	HS-Record-Len(MSB,0x00)	HS-Record-Len (0x00)	HS-Record-Len(LSB,0xF9)	ASN.1 CertData(SEQUENCE 0x30)
0x34	ASN.1 CertData (0x82)	ASN.1 CertData	ASN.1 CertData	ASN.1 CertData
…	…	…	…	…
0xCC	ASN.1 CertData	ASN.1 CertData	ASN.1 CertData	ASN.1 CertData

The meter answers with ACK without data.

C.3.2 2nd TLS fragment

Offset	0 (First Byte)	1	2	3
DLL	L=nn	C=53/73 (SND_UD)	MFCT0(GW)	MFCT1(GW)
DLL	ID_0(GW)	ID_1(GW)	ID_2(GW)	ID_3(GW)
DLL	VER(GW)	DEVTYPE(31)		
ELL	CI=8E(ELL)	CC	ACC	MFCT_0 (Meter)
ELL	MFCT_1(Meter)	ID_0(Meter)	ID_1(Meter)	ID_2(Meter)
ELL	ID_3(Meter)	VER(Meter)	DEVTYPE(Meter)	CI=90(AFL)
AFL	AFL.AFFL(0x02)	AFL.FCL[7..0] (0x02)	AFL.FCL[15..8] (0x00)	
0x00	ASN1.CertData	ASN1.CertData	ASN1.CertData	ASN1.CertData
…	…	…	…	…
0x58	ASN.1 CertData	ASN.1 CertData	ASN.1 CertData	ASN.1 CertData(CertDataEnd)
0x5C	Handshake-Type (0x0C) ServerKeyExchange	HS-Record-Len(MSB, 0x00)	HS-Record-Len (0x00)	HS-Record-Len(LSB 0x25)
0x60	CurveType=namedCurve (0x03)	0x00 (MSB)	0x17 (LSB,23=secp256r1)	Public ECC Key Point Len uncompress.(0x41)
0x64	ECC Point format	ECC Point Public	ECCPoint Public	ECCPoint Public

[5] NOTE: If session resumption is used, this field contains the session-id (up to 32 Bytes)

	(0x04)	(2*32 bytes)		
0x68	ECCPoint Public	ECCPoint Public	ECCPoint Public	ECCPoint Public
…	…	…	…	…
0xA4	ECCPoint Public	Handshake-Type (0x0D) CertificateRequest	HS-Record-Len(MSB, 0x00)	HS-Record-Len (0x00
0xA8	HS-Record-Len(LSB 0x08)	CertificateTypeList Len (0x01)	0x40 ECDSA_Sign	0x00 (MSB SignatureHashAlgoListLen)
0xAC	0x02 (LSB SignatureHashAlgoListLen)	0x04 (SHA256)	0x04 (SHA256)0x03 (ECDSA)	CAListLen (MSB 0x00)
0xB0	CAListLen (LSB 0x00)	MsgFragmentLenLSB, 0x29)	Handshake-Type (0x0E) ServerHelloDone	HS-Record-Len(MSB, 0x00)
0xB4	HS-Record-Len (0x00)	HS-Record-Len(LSB 0x00)		

The meter answers with ACK (no data). The meter waits for REQ_UD2 from the gateway.

C.4 3rd message flight Certificate, ClientKeyExchange, CertificateVerify, ChangeCipherSpec, Finished (from Client to Server)

C.4.1 1st TLS fragment

Grey shaded AFL parts are optional (fields present are indicated by AFL.FCL).

Offset	0 (First Byte)	1	2	3
DLL	L=nn	C=08/28 (RSP_UD)	MFCT0(MTR)	MFCT1(MTR)
DLL	ID_0(MTR)	ID_1(MTR)	ID_2(MTR)	ID_3(MTR)
DLL	VER(MTR)	DEVTYPE(MTR)		
ELL	CI=8C(ELL)	CC	ACC	CI=90(AFL)
AFL	AFL.AFFL(0x11)	AFL.FCL(0x01) LSB	AFL.FCL(0x7C)MSB MF=1,CTR,MCL,MAC,ML	AFL.MCL(0x65), CMAC8,CTR,ML
AFL	AFL.MCR[7..0]	AFL.MCR[15..8]	AFL.MCR[23..8]	AFL.MCR[31..8]
AFL	AFL.MAC	AFL.MAC	AFL.MAC	AFL.MAC
AFL	AFL.MAC	AFL.MAC	AFL.MAC	AFL.MAC
AFL/TPL	AFL.ML (0xB8, LSB)	AFL.ML (0x01, MSB)	CI=TCS(Transport Control Short)	ACC
TPL	STS	CF[7..0]	CF[15..8] (EM-TLS)	CF[23..16] (TLSHS)
0x00	MsgType (0x16,	ProtoMajor(0x03)	ProtoMinor(0x03)	TLS-

	Handshake)		TLS1.2	RecordLen(MSB, 0x01)
0x04	TLS-RecordLen(LSB, 0x6B)	Handshake-Type (0x0B) Certificate	HS-Record-Len(MSB, 0x00)	HS-Record-Len (0x00)
0x08	HS-Record-Len(LSB 0xF9)	ASN.1 CertData(SEQUENCE 0x30)	ASN.1 CertData (0x82)	ASN.1 CertData
0x0C	ASN1.CertData	ASN1.CertData	ASN1.CertData	ASN1.CertData
…	…	…	…	…
0xCC	ASN.1 CertData	ASN.1 CertData	ASN.1 CertData	ASN.1 CertData

The meter waits for REQ_UD2 from the gateway.

C.4.2 2nd TLS fragment

Offset	0 (First Byte)	1	2	3
DLL	L=nn	C=08/28 (RSP_UD)	MFCT0(MTR)	MFCT1(MTR)
DLL	ID_0(MTR)	ID_1(MTR)	ID_2(MTR)	ID_3(MTR)
DLL	VER(MTR)	DEVTYPE(MTR)		
ELL	CI=8C(ELL)	CC	ACC	CI=90(AFL)
AFL	AFL.AFFL(0x02)	AFL.FCL(0x02) LSB	AFL.FCL(0x00) MSB	
0x00	ASN.1 CertData	ASN.1 CertData	ASN.1 CertData	ASN.1 CertData
…	…			
0x30	ASN.1 CertData	ASN.1 CertData(End)	Handshake-Type (0x10) ClientKey Exchange	HS-Record-Len(MSB, 0x00)
0x38	HS-Record-Len (0x00)	HS-Record-Len(LSB 0x42)	Public ECC Key Point Len uncompressed (0x41)	ECC Point format (0x04)
0x3C	ECC Point Public (2*32 bytes)	ECCPoint Public	ECCPoint Public	ECCPoint Public
…	…	…	…	…
0x78	ECCPoint Public	ECCPoint Public	ECCPoint Public	ECCPoint Public
0x7C	Handshake-Type (0x0F) CertificateVerify	HSMsgLenMSB (0x00)	HSMsgLen	HSMsgLenLSB(0x20)
0x80	SignedHash	SignedHash	SignedHash	SignedHash
0x84	SignedHash	SignedHash	SignedHash	SignedHash
0x88	SignedHash	SignedHash	SignedHash	SignedHash

0x8C	SignedHash	SignedHash	SignedHash	SignedHash
0x90	SignedHash	SignedHash	SignedHash	SignedHash
0x94	SignedHash	SignedHash	SignedHash	SignedHash
0x98	SignedHash	SignedHash	SignedHash	SignedHash
0x9C	SignedHash	SignedHash	SignedHash	SignedHash
0xA0	MsgType (0x14, ChangeCipherSpec)	ProtoMajor (0x03)	ProtoMinor (0x03) TLS1.2	MsgFragmentLenMSB (0x00)
0xA4	MsgFragmentLenLSB, 0x05)	Handshake-Type (0x01) ChangeCipherSpec	HSMsgLenMSB(0x00)	HSMsgLen
0xA8	HSMsgLenLSB(0x01)	0x01	MsgType (0x16, Handshake)	ProtoMajor(0x03)
0xAC	ProtoMinor(0x03) TLS1.2	MsgFragmentLenMSB, 0x00)	MsgFragmentLenLSB, 0x34)	Handshake-Type (0x14) Finished
0xB0	HSMsgLenMSB(0x00)	HSMsgLen	HSMsgLenLSB(0x30)	Encrypted Padded Hash
0xB4	Encrypted Padded Hash	Encrypted Padded Hash	Encrypted Padded Hash	Encrypted Padded Hash
0xB8	Encrypted Padded Hash	Encrypted Padded Hash	Encrypted Padded Hash	Encrypted Padded Hash
0xBC	Encrypted Padded Hash	Encrypted Padded Hash	Encrypted Padded Hash	Encrypted Padded Hash
0xC0	Encrypted Padded Hash	Encrypted Padded Hash	Encrypted Padded Hash	Encrypted Padded Hash
0xC4	Encrypted Padded Hash	Encrypted Padded Hash	Encrypted Padded Hash	Encrypted Padded Hash
0xC8	Encrypted Padded Hash	Encrypted Padded Hash	Encrypted Padded Hash	Encrypted Padded Hash
0xCC	Encrypted Padded Hash	Encrypted Padded Hash	Encrypted Padded Hash	Encrypted Padded Hash
0xD0	Encrypted Padded Hash	Encrypted Padded Hash	Encrypted Padded Hash	Encrypted Padded Hash
0xD4	Encrypted Padded Hash	Encrypted Padded Hash	Encrypted Padded Hash	Encrypted Padded Hash
0xD8	Encrypted Padded Hash	Encrypted Padded Hash	Encrypted Padded Hash	Encrypted Padded Hash
0xDC	Encrypted Padded Hash	Encrypted Padded Hash	Encrypted Padded Hash	Encrypted Padded Hash
0xE0	Encrypted Padded Hash	Encrypted Padded Hash	Encrypted Padded Hash	

The meter waits for SND_UD from the gateway.

NOTE: The Handshake Hash is not truncated, if MAC Truncation is negotiated.

C.5 4th message flight ChangeCipherSpec, Finished (from Server to Client)

Grey shaded AFL is optional.

Offset	0 (First Byte)	1	2	3
DLL	L=nn	C=53/73 (SND_UD)	MFCT0(GW)	MFCT1(GW)
DLL	ID_0(GW)	ID_1(GW)	ID_2(GW)	ID_3(GW)
DLL	VER(GW)	DEVTYPE(31)		
ELL	CI=8E(ELL)	CC	ACC	MFCT_0(Meter)
ELL	MFCT_1(Meter)	ID_0(Meter)	ID_1(Meter)	ID_2 (Meter)
ELL	ID_3 (Meter)	VER(Meter)	DEVTYPE(Meter)	CI=90(AFL)
AFL	AFL.AFFL(0x0F)	AFL.FCL(0x01) LSB	AFL.FCL(0x2C)MSB MF=0,CTR,MCL,MAC	AFL.MCL(0x25), CMAC8,CTR
AFL	AFL.MCR[7..0]	AFL.MCR[15..8]	AFL.MCR[23..8]	AFL.MCR[31..8]
AFL	AFL.MAC	AFL.MAC	AFL.MAC	AFL.MAC
AFL	AFL.MAC	AFL.MAC	AFL.MAC	AFL.MAC
TPL	CI=TCS(Transport Control Short)	ACC	STS	CF[7..0]
TPL	CF[15..8](EM-TLS,)	CF [23..16](TLSHS)		
0x00	MsgType (0x14, ChangeCipherSpec)	ProtoMajor (0x03)	ProtoMinor (0x03) TLS1.2	MsgFragmentLenMSB (0x00)
0x04	MsgFragmentLenLSB, 0x05)	Handshake-Type (0x01) ChangeCipherSpec	HSMsgLenMSB(0x00)	HSMsgLen
0x08	HSMsgLenLSB(0x01)	0x01	MsgType (0x16, Handshake)	ProtoMajor(0x03)
0x0C	ProtoMinor(0x03) TLS1.2	MsgFragmentLenMSB, 0x00)	MsgFragmentLenLSB, 0x34)	Handshake-Type (0x14) Encrypted Finished
0x10	HSMsgLenMSB(0x00)	HSMsgLen	HSMsgLenLSB(0x30)	Encrypted Padded Hash
0x14	Encrypted Padded Hash	Encrypted Padded Hash	Encrypted Padded Hash	Encrypted Padded Hash
0x18	Encrypted Padded Hash	Encrypted Padded Hash	Encrypted Padded Hash	Encrypted Padded Hash
0x1C	Encrypted Padded Hash	Encrypted Padded Hash	Encrypted Padded Hash	Encrypted Padded Hash
0x20	Encrypted Padded Hash	Encrypted Padded Hash	Encrypted Padded Hash	Encrypted Padded Hash
0x24	Encrypted Padded Hash	Encrypted Padded Hash	Encrypted Padded Hash	Encrypted Padded Hash

0x28	Encrypted Padded Hash	Encrypted Padded Hash	Encrypted Padded Hash	Encrypted Padded Hash
0x2C	Encrypted Padded Hash	Encrypted Padded Hash	Encrypted Padded Hash	Encrypted Padded Hash
0x30	Encrypted Padded Hash	Encrypted Padded Hash	Encrypted Padded Hash	Encrypted Padded Hash
0x34	Encrypted Padded Hash	Encrypted Padded Hash	Encrypted Padded Hash	Encrypted Padded Hash
0x38	Encrypted Padded Hash	Encrypted Padded Hash	Encrypted Padded Hash	Encrypted Padded Hash
0x3C	Encrypted Padded Hash	Encrypted Padded Hash	Encrypted Padded Hash	Encrypted Padded Hash
0x40	Encrypted Padded Hash	Encrypted Padded Hash	Encrypted Padded Hash	

The meter answers with ACK (no data).

C.6 x-th message flight Application Data request (from gateway to meter)

Example: Cmd Set with M-Bus VIF/DIF

Grey shaded AFL is optional

Offset	0 (First Byte)	1	2	3
DLL	L=nn	C=53/73 (SND_UD)	MFCT0(GW)	MFCT1(GW)
DLL	ID_0(GW)	ID_1(GW)	ID_2(GW)	ID_3(GW)
DLL	VER(GW)	DEVTYPE(31)		
ELL	CI=8E(ELL)	CC	ACC	MFCT_0 (Meter)
ELL	MFCT_1(Meter)	ID_0 (Meter)	ID_1(Meter)	ID_2 (Meter)
ELL	ID_3 (Meter)	VER(Meter)	DEVTYPE(Meter)	CI=90(AFL)
AFL	AFL.AFFL(0x0F)	AFL.FCL[15..8] (0x01)	AFL.FCL[7..0] (0x2C)	AFL.MCL(0x25)
AFL	AFL.MCR[7..0]	AFL.MCR[15..8]	AFL.MCR[23..16]	AFL.MCR[31..24]
AFL	AFL.MAC	AFL.MAC	AFL.MAC	AFL.MAC
AFL	AFL.MAC	AFL.MAC	AFL.MAC	AFL.MAC
TPL	CI=0x5B (M-Bus Cmd to Device)	APL_ID 0x12	ID 0x34	ID 0x56
TPL	ID 0x78	MFCT 0xA2	MFCT 0x52	VER 0x01
TPL	DEVTYPE 0x31	ACC	STS	CF[7..0]
TPL	CF[15..8](EM-TLS)	CF[23..16](TLSAPP)		
0x00	MsgType (0x17, Application Data)	ProtoMajor (0x03)	ProtoMinor (0x03) TLS1.2	MsgFragmentLenMSB (0x00)

0x04	MsgFragmentLenLSB, 0x5D) incl. IV,HMAC and padding	IV (random)	IV (random)	IV (random)
0x08	IV (random)	IV (random)	IV (random)	IV (random)
0x0C	IV (random)	IV (random)	IV (random)	IV (random)
0x10	IV (random)	IV (random)	IV (random)	IV (random)
0x14	IV (random)	AppData DIF	AppData VIF	AppData Value
0x18	AppData Value	AppData Value	AppData Value	AppData Value
0x1C	AppData Value	AppData Value	AppData Value	AppData Value
0x20	AppData Value	AppData Value	AppData SHA256-HMAC FirstByte (truncated *or full*)	AppData HMAC
0x24	AppData HMAC	AppData HMAC	AppData HMAC	AppData HMAC
0x28	AppData HMAC	AppData HMAC	AppData HMAC	AppData HMAC
0x2C	AppData HMAC (only full HMAC)	AppData HMAC	AppData HMAC	AppData HMAC
0x30	AppData HMAC	AppData HMAC	AppData HMAC	AppData HMAC
0x34	AppData HMAC	AppData HMAC	AppData HMAC	AppData HMAC
0x38	AppData HMAC	AppData HMAC	AppData HMAC	AppData HMAC
0x3C	AppData HMAC	AppData HMAC	AppData HMAC	AppData HMAC
0x40	AppData HMAC	AppData HMAC 32(Last)	AES Padding 0..15 Byes for AES-128	Padding
0x44	Padding	Padding	Padding	Padding
0x48	Padding	Padding	Padding	Padding
0x4C	Padding	Padding	Padding	Padding
0x50	Padding	PadLen (0x0F)		

The meter answers with ACK (no data). The gateway sends REQ_UD2 to receive Application data from the meter.

The actual Application Layer Payload is shown with background color (orange).

Optional fields are shown in Italics.

The HMAC size for SHA-256 is 32 bytes. If both sides (Server and client) support HMAC truncation (see RFC6066), the HMAC size can be truncated to 10 bytes. A minimum of 0 to 15 Pad bytes should be added before the PadLen byte. The reason is the 16-byte Block length requirement for AES128. To hide the actual payload size, a multiple of 16 bytes Padding can be added. (up to a total of 255 Padding bytes)

Because the LinkLayer (12), Extended LinkLayer (3/11), the AFL Header (4/19) and the TPL Header (0/5/13) adds 19 to 55 bytes, the APL Application Layer Data Size is typically around 200 bytes. It is the task of the application layer to discover the maximum unfragmented TPL+APL size. To reduce the overall data overhead as much as possible data should be aggregated into a TLS message “fragment”. Because of the HMAC, padding and IV overhead it is more efficient to do the fragmentation in the AFL layer than within the TLS application layer.

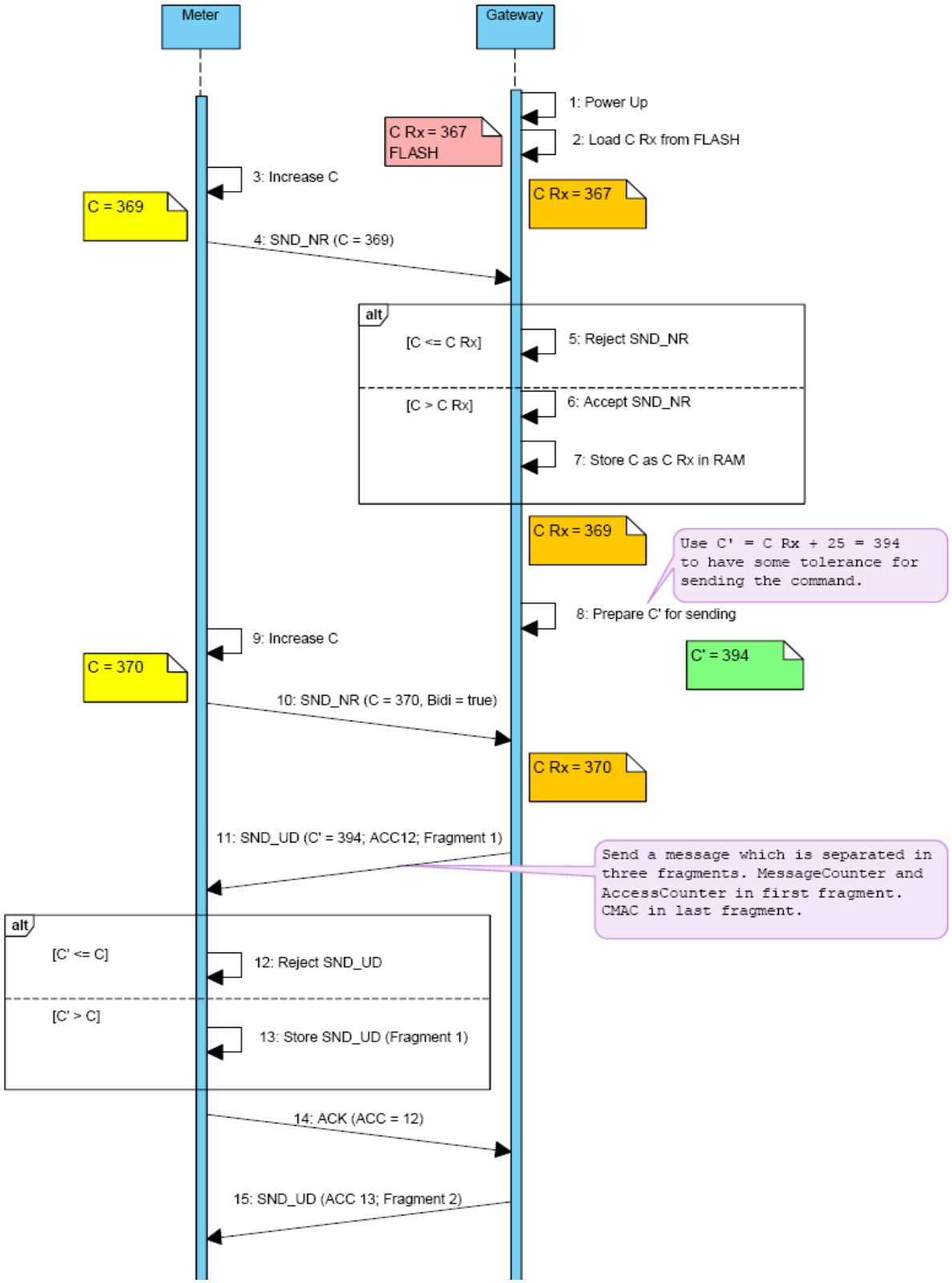
Annex D. (informative): Example Certificate

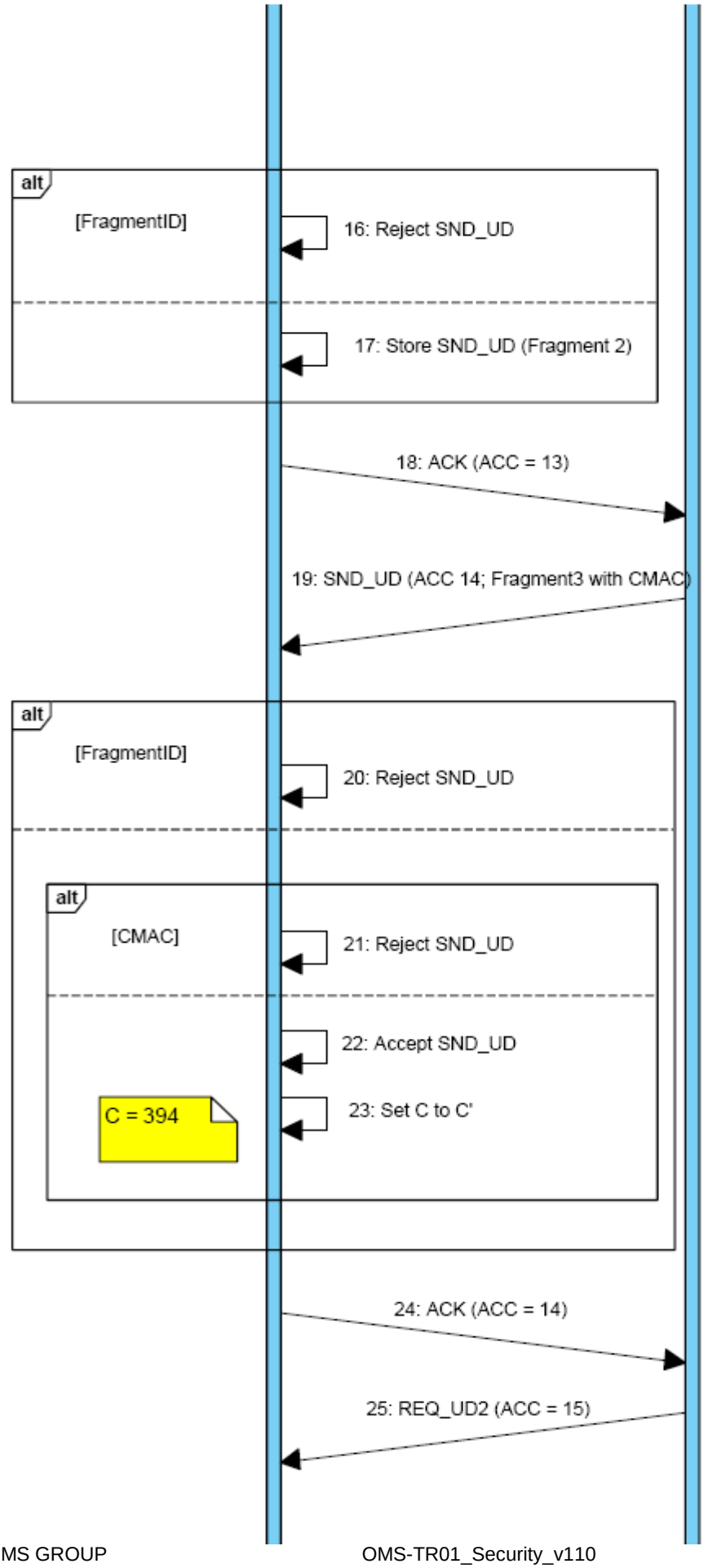
A small selfsigned certificate is shown below.

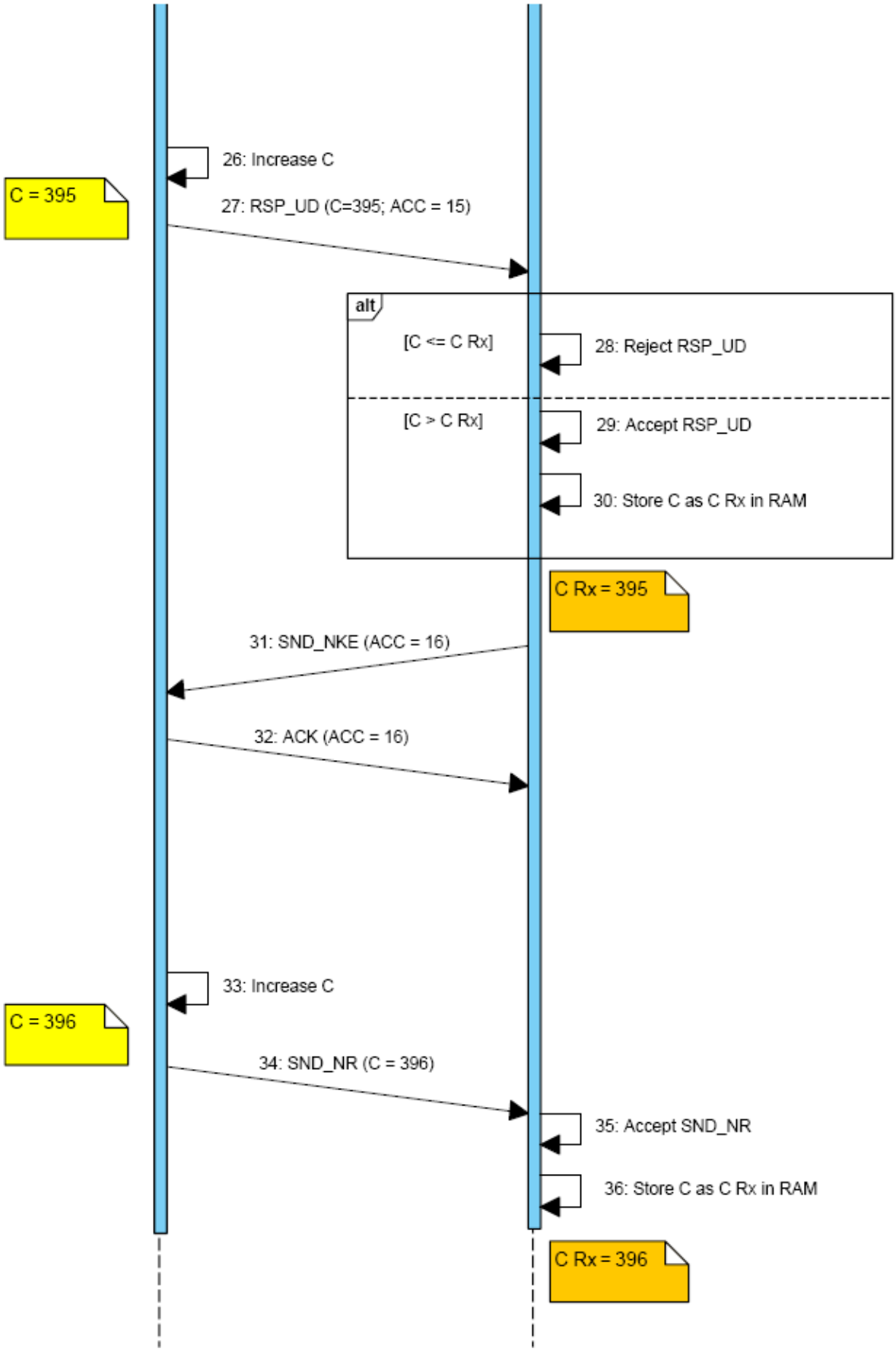
NOTE: No certificate extensions are present and an empty issuer and subject field is used. The size of the certificate below is 249 bytes.

```
X509v3 Certificate Data:
    Version: 3 (0x2)
    Serial Number:
      d6:ae:40:cb:99:cc:27:b9
    Signature Algorithm: ecdsa-with-SHA256
    Issuer:
    Validity
      Not Before: Oct 25 08:30:36 2011 GMT
      Not After : Oct 22 08:30:36 2021 GMT
    Subject:
    Subject Public Key Info:
      Public Key Algorithm: id-ecPublicKey
        Public-Key: (256 bit)
        pub:
        04:4f:ba:9b:29:f9:3f:b8:86:31:24:31:97:bf:da:5b:eb:6d:b7:71:ad:32:54:34:e5:51:f8:dc:a0:60:
cd:bc:3f:11:07:87:89:3d:7b:8f:16:32:31:a9:ef:c2:a9:e0:87:1d:c0:c3:d8:ea:90:7b:35:9b:bd:ac:85:ff:55:
53:6d
        ASN1 OID: prime256v1
  Signature Algorithm: ecdsa-with-SHA256
    30:45:02:21:00:d1:4f:45:1f:b3:a5:bd:87:b1:00:36:8a:da:
    6f:5d:ec:54:23:33:90:33:c8:d9:8e:fd:80:f8:0e:04:b7:47:
    09:02:20:1c:d7:a2:63:30:6a:c2:d0:78:21:57:a2:cb:84:bc:
    2e:99:5e:c1:49:84:5e:e7:b1:0c:c0:75:84:0f:3b:7e:10
```

Annex E. (informative): Example usage of Message Counter C/C'




alt
[FragmentID]
16: Reject SND_UD
17: Store SND_UD (Fragment 2)
18: ACK (ACC = 13)
19: SND_UD (ACC 14; Fragment3 with CMAC)
alt
[FragmentID]
20: Reject SND_UD
alt
[CMAC]
21: Reject SND_UD
22: Accept SND_UD
C = 394
23: Set C to C'
24: ACK (ACC = 14)
25: REQ_UD2 (ACC = 15)


C = 395
26: Increase C
27: RSP_UD (C=395; ACC = 15)
alt
[C <= C Rx]
28: Reject RSP_UD
[C > C Rx]
29: Accept RSP_UD
30: Store C as C Rx in RAM
C Rx = 395
31: SND_NKE (ACC = 16)
32: ACK (ACC = 16)
33: Increase C
C = 396
34: SND_NR (C = 396)
35: Accept SND_NR
36: Store C as C Rx in RAM
C Rx = 396