Open Metering System Specification

Security Mode 13 - Transport Layer Security (TLS)

Annex F to
Volume 2 Primary Communication
Issue 5.0.1

RELEASE C (2023-12)

Document History

Version	Date	Comment	Editor
A 0.1.0	2013-07-18	Draft based on “Technical Report 01 Security”	Dirk Matussek
A 0.1.1	2013-09.24	Revision of first draft	Dirk Matussek/Uwe Pahl
A 0.2.0	2013-10-07	Review with M. Staubermann	Uwe Pahl
A 0.2.1	2013-10-17	Update of the Brain pool curves	Uwe Pahl
A 0.2.2	2013-10-23	Update ELL usage; Editorial revision of Message flights	Uwe Pahl
A 0.3.0	2014-01-10	Update References	Uwe Pahl
A.0.3.1	2014-01-15	F.B.4/F.B.6 Examples corrected (CMAC into last fragment, PadLen calculation explained)	T.Blank, M.Staubermann
A.0.3.2	2014-01-17	Processed comments since A0.3.0	M.Staubermann
A.0.3.3	2015-05-27	F.B.2 and F.B.3.1 replace UTC_unix_time by random,	M.Staubermann
A. 0.3.4	2015-08-28	F.2.1 update Rules for truncated HMAC	M.Dold, M.Staubermann
A. 0.3.5	2016-03-18	F.3 Requirements for Certificate select	U.Pahl, M.Rac
A. 0.3.6	2016-04-18 2016-04-26	F.C SITP Examples Review TF-S	M.Dold. TF-S
A 0.3.7	2016-09-09 2016-09-28	F.3 Update Security Information: Master key, Update Meter and Gatway certificates using SITP. Splitting Tab.2 CF in Tab.2 CF+Tab.3 CFE Tab.7+8 remove footnote HMAC size F.B.1+ F.B.6 Add wireless M-Bus example F.2.5 Add new " Meter behaviour for different TLS channel states" F.B. new Annex Key Exchange added	M. Staubermann U.Pahl
A-0.3.8	2016-09-30	Update table 8	U.Pahl
A-0.3.9	2016-10-27	F.3 ff KeyVersion handling updated	M.Staubermann
A-0.3.10	2016-10-28	Revison in TF-S meeting	U.Pahl
A-0.3.11	2016-11-16	Revison of F.2.3 and F.3 key versioning by “Generator” of Key.	Th. Blank
A 0.3.12	2016-11-19 2016-12-15 2016-12-16 2016-12-16	Encrypt-then-mac for AES-CBC required. Examples updated. MsgType->ContentType Update of F.3 F.D.1 Revison of Example Revision of Terms in 2.3.1, Edit Tab.5 +6, Add Tab.7	M.Staubermann T.Blank U.Pahl TF-S
A-0.3.13	2016-12-16 2017.02.03	Clean Version Revision F2.4.1 Update Color codes in F.C and F.D	 TF-S U.Pahl
A-0.3.14	2017-02-17	Correct BCF values Rework F.3 with Table F.ormats	T. Blank T. Blank
A- 0.3.15	2017-02-22 2017-03-03 2017-03-15	Revision of Appendix F.D Revision of Appendix F.B Replace partially terms meter/gateway by client/server	D.Jäckle, U.Pahl, M.Bernaud
A- 0.3.16	2017-03-17	Review by TF-S	TF-S
A-0.3.17	2017-06-17	Add F.2 and F.4.1, F.4.3.3 Update Appendix F.B and F.D	TF-S, U.Pahl
A-0.3.18	2017-08-11 2017-08-30	Add new Appendix F.A “SITP” Extend Tab. F.A.7 by error code 09h	U.Pahl
A-03.19	2017-09-04 2017-09.08	Editorial Changes of F.C Update of figures of F.C	M.Staubermann U.Pahl

A-0.3.20	2017-09-11 2017-09-20 2017-09-25	Reset of change log Insert key counter maximum and new status response byte definitions Add asymmetric DSIs in BCF table Add new F.3.5.1 and rework F.3.5.2 regarding usage of Sec.Profile B and C, Rename all Table *x* to Table F.*x*, Change Ref to FprEN13757-3,-7 Update embedded figures in Appendix F.C	T. Blank U.Pahl U.Pahl
A-0.3.21	2017-10-27	F.2 Add terms TLS session, TLS channel, TLS connection, F.3.2 usage of length N in CF	TF-S
A-0.3.22	2018-01-02	F.4.2.2 Updated Synchronisation Procedure for Failed Master key Update. F.4.2.3 Restoring MessageCounter for Factory Master key Reset	M.Staubermann
A-0.3.23	2018-01-19	F4.2.3 Anpassung im TF-S Meeting	TF-S
A-0.3.24	2018-03-02	F.4.2.3+F.4.4.3 Deletion of MK+CRT with reset of MK0;Delete F.4.3.6; Tab.F7: new footnote b; F.D: Add Footnote to preamble+CRC	TF-S
A-0.3.25	2018-03-23 2018-04-13 2018-04-14	Explain “date of issue” in F.4.3.3 Delete any key versions of certificates and key pairs in F.4.3 Explain principle of ActivePair and NewPair in F.4.3.1 Update Annex A “SITP” Revision in TF security meeting Update Figure F.C.2 and F.C.4	T. Blank TF-S U. Pahl
A-0.3.26	2018-04-14 2018-04-16 2018-05-15 2018-05-16	Reset of change-log; Add Table F.ootnote a in Tab.F.A.6 Add new Tab.F7, Update Tab. F7 and F.D.2 Separate Transfer and Activation in F.4.3. and F.4.4; Move 4.2.3 and 4.4.3 to new 4.5 F.D. editorial changes	U. Pahl U.Pahl / M.Staubermann T. Blank U.Pahl D. Jäckle
A-0.3.27	2018-05-18	Join F.4.3.3. and F.4.3.4, editorial work in F.4.3 New Note5 in F.D.3 Reset of change log	TF-S
A-0.3.28	2018-05-18	F.4.3.1 Alternatives for MTR_TLS_CRT/KeyPair Update described. Verification Checks during Certifcate Import	M.Staubermann
A-0.3.29	2018-07-27	F.2 Add Table F..1 with terms of RFC5246 Change ref. from TR03109-3 to TR03116-3 Update Ref. to EN13757-3:2018 and EN13757-7:2018; add title “F.3.2. Transport Layer – Configuration Field Extension”	U.Pahl
A-0.3.30	2018-08-07	F.4.3.1 Change terms ActivePair / NewPair to ActiveCredential and NewCerdential	T. Blank
A-0.3.31	2018-08-10	Editorial revision, CamelCase notation, hyphen verification	M. Rac
A-0.3.32	2018-08-31	F.4.3.2 no KeyPair Renewal by the meter; Reset change log	TF-S
A-0.3.33	2018-10-05 2018-10-10 2018-11-06	Editorial change of tables in F.D Correction of examples in F.E (mainly SITP) Replace terms in Table F.1 and dokument, add definition for TLSSDULen, Update ref. in Table F.7 Correct ELL in F.D.7 Check and correction of all TLSSDULen fields in F.D	T. Blank U. Pahl T. Blank
A-0.3.34	2018-11-25 2018-11-30 2019-01-11	Editorial changes concerning CamelCase writing of special fields F.3.1 Add req. for sequence of HS-msg F.3.1 Add note for RFC 8449; F.3.4.4 usage of CMAC/HMAC Replace reference [OMSV2] by [OMS-S2] Insert F.5 TLS Certificates for meters	M. Rac TF-S U. Pahl T. Blank

A-0.4.0	19.01.2019 21.01.2019 23.01.2019 05.02.2019	F.2 Add definition TLSPlaintext and TLSCipherText; Update Table F.7 Transpose Table F.7 Correction of example F.D.4 Replace term “MasterKey” by “master key”; add list of figures; Generation of final draft	TF-S U.Pahl T. Blank U.Pahl
A-0.4.1	01.03.2019 14.03.2019 15.03.2019 19.03.2019 01.04.2019	Consider comments from BSI; F.3.1 Add two new requirement using ellipt.curves, ECDSA and Hash; F.3.1 move notes from F.D.3 F.3.4.4 double Tab.13 and 14 for GCM; F.3.6.2 editorial review. F.3.6.4 Extend Table F.19 for authentication F.4.4.1 add note; F.4.2 move note from F.4.5 F.C change key exchange to key renewal F.D.2 / 8 editorial change of address fields F.D.3 editorial change F.D.5.2 Signed Hash to 70 bytes F.E.1 insert TLS padding F.4.2.1 insert random z1 for MK’ calculation F.4.1 Add restriction for key upload Editorial changes in the Meeting F.3.5.2 add footnote for third fragment F.E.2 insert TLS padding Generation draft Tab. F14, editorial, Generation of final draft	U.Pahl T. Blank U.Pahl TF-S T. Blank U. Pahl U. Pahl
A-0.4.2	06.06.2019 14.06.2019 14.06.2019	F.F Updated Example for Meter Certificate F.A.5 Remove DSI 01 from BCF 06; F.F. Add Certificate in DER format Generation of release candidate	M. Staubermann TF-S U. Pahl
A-0.4.3	23.10.2019 25.10.2019	Editorial review, update references to OMS-S2 + EN13757-4 and internal reference errors; Add range A2h-AFh in Table F.A.6; Update source files of Appendix F.C Generation of 2nd release candidate	U. Pahl
A-1.0.0	22.11.2019	No comments from OMS-group; Reset doc. change tracking Generate 1st release	U. Pahl
B 1.0.1	2022-12	Introduction of term “OMS end-device”, as in OMS-S2; general workover on formats Editorial changes Copyright remark added to front page Term “key material” replaced by “keying material” Release	A. Bolder A. Reissinger
C 1.1.0	2023-12-06 to 2024-02-06	Consideration of changed requirements in revision 2021 of “Technische Richtlinie BSI TR-03116” Editorial improvements Release Candidate	A. Mathew P. Joppich-Dohlus A. Reissinger
C 1.1.1	2024-05-03 to 2024-05-14	Consideration of review comments Release	TF Security A. Reissinger

Contents

List of tables

List of Figures

F.1 Normative references

[OMS-S2]	OMS-Specification Volume 2 Issue 5.0.1 (the superior document)
[BSITR03109-1]	BSI Technische Richtlinie BSI TR-03109-1: Anforderungen an die Interoperabilität der Kommunikationseinheit eines intelligenten Messsystems] Version 1.1, Datum 17.09.2021 https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/TechnischeRichtlinien/TR03109/TR03109-1.pdf
[BSITR03116-3]	BSI Technische Richtlinie TR-03116-3 Kryptographische Vorgaben für Projekte der Bundesregierung; Teil 3: Intelligente Messsysteme, Datum 06.12.2022 https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/TechnischeRichtlinien/TR03116/BSI-TR-03116-3.pdf
[EN13757-3:2018]	Communication systems for meters — Part 3: Application protocols
[EN13757-7:2018]	Communication systems for meters — Part 7: Transport and security services
[EN13757-4:2019]	Communication systems for meters — Part 4: Wireless M-Bus communication;
[RFC4493]	Network Working Group Request for Comments: 4493 The AES-CMAC Algorithm; June 2006 https://tools.ietf.org/html/rfc4493
[RFC5246]	Network Working Group Request for Comments: 5246 The Transport Layer Security (TLS) Protocol Version 1.2; August 2008 https://tools.ietf.org/html/rfc5246
[RFC5289]	Network Working Group Request for Comments: 5289 TLS Elliptic Curve Cipher Suites with SHA-256/384 and AES Galois Counter Mode (GCM); August 2008 https://tools.ietf.org/html/rfc5289
[RFC6066]	Internet Engineering Task Force (IETF) Request for Comments: 6066 Transport Layer Security (TLS) Extensions: Extension Definitions; January 2011 https://tools.ietf.org/html/rfc6066
[RFC7027]	Internet Engineering Task Force (IETF) Request for Comments: 7027 Elliptic Curve Cryptography (ECC) Brainpool Curves for Transport Layer Security (TLS); October 2013 https://tools.ietf.org/html/rfc7027
[RFC7366]	Internet Engineering Task Force (IETF) Request for Comments: 7366 Encrypt-then-MAC for Transport Layer Security (TLS) and DTLS, September 2014 https://tools.ietf.org/html/rfc7366
[RFC8422]	Network Working Group Request for Comments: 8422 Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS) Versions 1.2 and Earlier; Aug. 2018 https://tools.ietf.org/html/rfc8422

F.2 Introduction

This annex is an amendment to the OMS-Specification [OMS-S2] and specifies the security mechanisms of security mode 13 for the M-Bus protocol.

Security mode 13 requires asymmetric encryption and specifies the utilization of Transport Layer Security (TLS). Since TLS is a security implementation with many configuration possibilities, this specification provides limitations, extensions and options to TLS. This includes also an applicable selection of crypto and cipher suites.

The management of security related services and the exchange of security information is handled with the SITP protocol [EN13757-7:2018]. E.g., the SITP protocol provides procedures for key and certificate updates and exchanges.

This specification comes with examples for the frame structure when using the AFL in combination with TLS, key exchanges and TLS messages. The examples are available for M-Bus and wM-Bus. Further examples for TLS channel establishment management, application data transfer and master key updates are defined utilizing the SITP protocol. Also, an example of a certificate is present.

Following special terms are used in this annex:

TLS session:	Security association between two TLS endpoints. The TLS session starts with negotiation of secrets between the TLS endpoints during a non-resumed TLS handshake. The TLS session ends either by session timeout or by replacement with a new session between the same endpoints.
TLS channel:	Reliable transport established between two TLS endpoints. The TLS channel is established with a response to a TLS ChannelRequest message. The TLS channel is closed either by end of the session or by transmission of a TLS alert. A TLS channel uses the AFL.
TLS connection:	This term is not used in this document, because neither DLL/ELL nor AFL is connection oriented.
TLS record	Is either a TLSPlaintext or TLSCiphertext.
TLSPlaintext	Contains a TLS record header and content specific data.
TLSCiphertext	Contains a TLS record header and TLS cipher suite specific header, content specific data and TLS cipher suite specific trailer.

The following table list the terms used in this specification and its corresponding name used in [RFC5246].

Table F.1 – List of used in [RFC5246]

Name in this Annex	Name in [RFC5246]
ContentType	TLSPlaintext.type TLSCiphertext.type
ProtoMajor	TLSPlaintext.version.major TLSCiphertext.version.major
ProtoMinor	TLSPlaintext.version.minor TLSCiphertext.version.minor
TLSSDULen a	TLSPlaintext.length TLSCiphertext.length
HSMsgType	Handshake.msg_type
HSMsgLength	Handshake.length
HSClientVersionMajor	Handshake.ClientHello.client_version.major
HSClientVersionMinor	Handshake.ClientHello.client_version.minor
HSServerVersionMajor	Handshake.ServerHello.server_version.major
HSServerVersionMinor	Handshake.ServerHello.server_version.minor
HSClientRandom	Handshake.ClientHello.random
HSServerRandom	Handshake.ServerHello.random
HSSessionIDLength	length(Handshake.ClientHello.session_id)
[a] Length of TLS service data unit	

F.3 Asymmetric encryption with security mode 13

F.3.1 Required TLS operation for security profile C

Asymmetric encryption is performed with security profile C (security mode 13) which allows the usage of TLS for the M-Bus protocol.

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.

- OEDs shall implement the TLS client functionality; gateways shall 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 ClientHello MaxFragmentLength extension (see [RFC6066]) with min. 512 bytes fragment size.
      NOTE 1: This has not to be confused with the AFL-fragment length.
      NOTE 2: In future version of this Annex it is intended to improve fragmentation according to RFC8449.
- Servers shall not support the ClientHello truncated-HMAC extension (see [RFC6066]) in addition to the standard HMAC size of 32 bytes (as defined in [RFC5246]) according to [BSI-TR03116-3]. Clients shall also not support the ClientHello truncated-HMAC extension (see [RFC6066]) in order to save data overhead.
- Servers shall accept the encrypt-then-MAC ClientHello extension and support encrypt-then-MAC according to [RFC7366] if AES-CBC cipher suites are offered.
- Clients should send the encrypt-then-MAC ClientHello extension and should support encrypt-then-MAC according to [RFC7366] if an AES-CBC cipher suite is requested.
- With the beginning of 2025 in newly installed meters, clients shall support the encrypt-then-MAC ClientHello extension (according to [BSI-TR03116-3]) and offer it in the ClientHello. If the encrypt-then-MAC is offered, then the server shall use either a GCM cipher or the encrypt-then-MAC.
- Clients shall accept the additional TLS ChannelRequest Message. This message shall be authenticated with at least 8 bytes of AES128 CMAC protection.
- Handshake messages (according to Table F.12) shall be transmitted/ accepted by the client (OED) only in the required sequence of RFC5246, Figure1 and Figure 2.
- TLS session renegotiation is not allowed.
- A TLS session can be resumed. This document allows a max. TLS session lifetime of 1 month (31 days) unless not more than 5 Mbytes of TLS data (according to [BSI-TR03116-3]) is transmitted in both directions within this TLS session. TLS session resumption shall be implemented in the gateway.
- A new (authenticated) TLS ChannelRequest from server (gateway) to client (OED) shall close any existing TLS channel between client (OED) and server (gateway) and establish (resume or negotiate) a new one.
- For TLS crypto suites based on elliptical curves, the uncompressed point format shall be used.
- Clients and servers shall support ECDSA for signature validation and generation with the required hash algorithms according to [BSI TR-3116-3], 6.4.1.
- Clients and servers shall support the \`supported_signatures_extension\` according to [RFC5246] 7.4.1.4.1 to indicate the supported signature algorithms (e.g. ECDSA) and hash algorithms.

TLS crypto suites based on elliptical curves shall be implemented according to [RFC8422] or [RFC5289]

The identifiers for brainpool TLS named curve IDs are defined in [RFC7027]. At least the brainpoolP256r1 and Secp256r1/NIST P256 shall be supported for ECDHE and ECDSA keys. The support of the other TLS named curves is recommended.

NOTE 1: The default FragmentSize limit in TLS is 16 kbytes (which shall be buffered by the receiver and the transmitter). For memory size restricted OEDs a max_fragment_length extension can be negotiated, in case the client also supports it. The negotiation is defined in [RFC6066].

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

NOTE 3: 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 4: After termination of the session (in an orderly manner the gateway shall send a CloseNotify message to the OED), the OED may immediately start a new TLS session by sending a ClientHello to the gateway.

Table F.2 – List of supported elliptical curves

Curve name	TLS named curve ID
brainpoolP256r1	001Ah
brainpoolP384r1	001Bh
brainpoolP512r1	001Ch
Secp256r1/NIST P256	0017h
Secp384r1/NIST P384	0018h

NOTE: The [BSITR03116-3] recommends several cipher suites and its usage period. It is recommended to consider this technical guideline for the selection of supported cipher suites. This technical report will be updated periodically based on the [BSITR03116-3]!

F.3.2 Transport layer / configuration field

The TLS security mode is declared with a 13 (0Dh) in the field “security mode” (MMMMM).

NOTE: Bits for link control are served by the Extended Link Layer (ELL), which has always been applied together with TLS.

The configuration field (CF) of the security mode 13 is defined in Table F.3:

Table F.3 – Configuration field of security mode 13

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
Content of Message	Content of Message	Reserved	Security Mode	Security Mode	Security Mode	Security Mode	Security Mode	Number of bytes for Mode 13	Number of bytes for Mode 13	Number of bytes for Mode 13	Number of bytes for Mode 13	Number of bytes for Mode 13	Number of bytes for Mode 13	Number of bytes for Mode 13	Number of bytes for Mode 13
C	C	0	M	M	M	M	M	N	N	N	N	N	N	N	N

M shall always be 13 (0Dh) to declare an encryption with TLS

NOTE: The applied TLS version has to be traced from TLS header

C	declares the content of the message and shall comply with [EN13757-3:2018].
N	The N field is not used in security mode 13 and shall always be set to FFh. The number of following bytes is taken from field TLSSDULen in the TLS record header (see Table F.9).

F.3.3 Transport layer / configuration field extension

Security mode 13 requires a Configuration Field Extension (CFE) with a size of one byte according to Table F.4.

Table F.4 – Configuration Field Extension of security mode 13

Bit 23	Bit 22	Bit 21	Bit 20	Bit 19	Bit 18	Bit 17	Bit 16
Reserved	Reserved	Reserved	Reserved	ProtocolType 3	ProtocolType 2	ProtocolType 1	ProtocolType 0
0	0	0	0	P	P	P	P

P	is the Protocol Type as defined in Table F.5

Table F.5 – Protocol types of security mode 13

Content of protocol type P	Meaning
0000	TLSCHAN (TLS ChannelRequest)
0001	TLSPROT (TLS handshake , TLS alert, TLS application (see Table F.6))
0010..1111	Reserved

F.3.4 Structure of different TLS messages

F.3.4.1 General

This section presents the structure and the applicable content types of TLS. The Table F.6 shows the different content types of the TLS protocol. The following sections show the structure of the message depending on the content type.

Table F.6 – Mapping between TLS protocol type and ContentType

Symbol name	ContentType name	ContentType (according [RFC 5246])	TLS protocol type (according to Table F.5)
TLSCHAN	TLS ChannelRequest	00_h	TLSCHAN
TLSHS	TLS Handshake	14_h, 16_h	TLSPROT
TLSALERT	TLS Alert	15_h	TLSPROT
TLSAPP	TLS Application	17_h	TLSPROT

The TLS structure follows the CF and CFE bytes in case of security mode 13. It corresponds with the “optional TPL header fields” and (depending on the ContentType) with “Encrypted/unencrypted application data” and “TPL trailer field” as shown in [EN13757-7:2018] Table 20.

Table F.7 – General structure of a TLS Message

		TLSCHAN	TLSHS, TLSALERT c	TLSHS, TLSALERT d	TLSAPP
TPL-Header	CI-field	$5F_h$	$5F_h$, $9E_h$, $9F_h$,	$5F_h$, $9E_h$, $9F_h$,	e.g. $5B_h$, $7A_h$, $C3_h$, $C4_h$
	CI-dependent fields	See EN 13757-7 Table 11 + 12	See EN 13757-7 Table 11 + 12	See EN 13757-7 Table 11 + 12	See EN 13757-7 Table 11 + 12
	Configuration field extension	See Table F.4	See Table F.4	See Table F.4	See Table F.4
	Optional TPL header fields [a]	TLSCHAN header (see Table F.8)	TLS record header [b] (see Table F.9)	TLS record header [b] (see Table F.9)	TLS record header (see Table F.9)
		No TLS cipher suite specific header	No TLS cipher suite specific header[b]	TLS cipher suite specific header [b]	TLS cipher suite specific header
APL data	optional	No data	Content specific data [b]	Content specific data [b]	APL Data
TPL-Trailer	optional	No TLS cipher suite specific trailer	No TLS cipher suite specific trailer[b]	TLS cipher suite specific trailer [b]	TLS cipher suite specific trailer

[a] The Optional TPL header fields need to be present for all message types used in this annex!
[b] To minimize data overhead several TLS records can be concatenated into one TLS message.
[c] TLSPlaintext (see F.2)
[d] TLSCiphertext (see F.2)

NOTE: This presentation only shows transport and application layer

Table F.8 – TLSCHAN header

TLS record header of TLSCHAN			
ContentType (1 byte)	reserved (1 byte) [b]	reserved (1 byte) [b]	TLSSDULen (2 byte) [a]
[a] TLSSDULen is transmitted with MSB first [b] TLSCHAN don’t use fields ProtoMajor and ProtoMinor			

Table F.9 – TLS record header according to TLS1.2 [RFC 5246]

TLS record header of TLSALERT, TLSHS, TLSAPP			
ContentType (1 byte)	ProtoMajor (1 byte)	ProtoMinor (1 byte)	TLSSDULen (2 byte) [a]
[a] TLSSDULen is transmitted with MSB first			

ContentType	shall be according to Table F.6.
ProtoMajor and ProtoMinor	shall be according to [RFC 5246] (see Table F.1)
TLSSDULen	is the length (in bytes) of the TLSPlaintext.length or TLSCiphertext.length and shall be used according to [RFC 5246] (see also Table F.1)

F.3.4.2 TLS ChannelRequest

Table F.10 – TLS ChannelRequest

CI = 5Fh	ADDR, ACC, ST, CF = 13; TLSCHAN	TLS ContentType (00h)	Reserved (00h)	Reserved (00h)	TLSSDULen MSB = 00h	TLSSDULen LSB = 00h

This content type requires always a CMAC protection by the AFL layer.

F.3.4.3 TLS Handshake

Table F.11 – TLS Handshake

CI = 5Fh/ 9Eh/ 9Fh	[ADDR,]ACC, ST,CF = 13; TLSPROT	TLS ContentType (14h, 16h)	TLS ProtoMajor (03h)	TLS ProtoMinor (03h)	TLSSDULen MSB	TLSSDULen LSB
HSMsgType	TLS MsgData					

The following table describes the used handshake message types according to RFC5246 and the support of CMAC-protection.

Table F.12 – Handshake message types

ContentType	HSMsgType		Name of HSMsgType in RFC	CMAC required
14h	01h		ChangeCipherSpec	NO
16h	01h		ClientHello	YES
16h	02h		ServerHello	NO
16h	0Bh		Certificate	NO
16h	0Ch		ServerKeyExchange	NO
16h	0Dh		CertificateRequest	NO
16h	0Eh		ServerHelloDone	NO
16h	0Fh		CertificateVerify	NO
16h	10h		ClientKeyExchange	NO
16h	14h		Finished	NO

F.3.4.4 TLS Alert

Table F.13 – TLS Alert with HMAC

CI = 5Fh/ 9Eh/ 9Fh.	[ADDR,]ACC, ST,CF = 13; TLSPROT	TLS ContentType (15h)	TLS ProtoMajor (03h)	TLS ProtoMinor (03h)	TLSSDULen MSB	TLSSDULen LSB
TLS random IV (16 bytes)	Encrypted alert data [a]				Padding [b]	TLS HMAC (32 bytes)

[a] The encrypted alert data contains a 2 byte content according to chapter 7.2 of [RFC5246].
[b] Padding according to [RFC5246]

Table F.14 – TLS Alert with GCM

CI = 5Fh/ 9Eh/ 9Fh.	[ADDR,]ACC, ST, CF = 13; TLSPROT	TLS ContentType (15h)	TLS ProtoMajor (03h)	TLS ProtoMinor (03h)	TLSSDULen MSB	TLSSDULen LSB
GCM nonce	Encrypted alert data [a, b]					

[a] The encrypted alert data contains a 2 byte content according to chapter 7.2 of [RFC5246].
[b] The padding bytes and the authentication tag are part of the encrypted alert data.

During a valid TLS session, the TLS Alert is protected by TLS. In all other cases the TLS Alert shall be protected by the CMAC in the AFL layer.

F.3.4.5 TLS Application

Table F.15 – TLS Application with HMAC

CI = 5Bh/ 7Ah/ 72h AppData	[ADDR,]ACC, ST, CF = 13; TLSPROT	TLS ContentType (17h)	TLS ProtoMajor (03h)	TLS ProtoMinor (03h)	TLSSDULen MSB	TLSSDULen LSB
TLS random IV (16 bytes)	Encrypted app. data TLS HMAC (32 bytes)					TLS HMAC (32 bytes)

This content type is protected by TLS and shall not apply a CMAC protection.

Table F.16 – TLS Application with GCM

CI = 5Bh/ 7Ah/ 72h AppData	[ADDR,]ACC, ST, CF = 13; TLSPROT	TLS ContentType (17h)	TLS ProtoMajor (03h)	TLS ProtoMinor (03h)	TLSSDULen MSB	TLSSDULen LSB
GCM nounce	Encrypted app. data					
[a] The encrypted app. data contains the padding bytes and the authentication tag beside the application data.						

NOTE: The maximum size for an AFL fragmented message is limited to 16 kbytes. Therefore the maximum TLSSDULen is less than 16kbytes, depending on the other used fields, e.g. in the TLS header. Client and server can agree to use a smaller maximum TLS fragment size (ClientHello extension).

Be aware that existing TLS 1.2 [RFC5246] with AES-CBC cipher suite applies HMAC over the plain message. The encryption is done thereafter. OED and gateway are required to support the encrypt-then-MAC according to [RFC7366] for AES-CBC cipher suites.

F.3.5 TLS message sequence

F.3.5.1 Establishment of TLS channel and prevention of ClientHello flooding

Most wireless M-Bus channels are limited in bandwidth and in duty cycle. Sending many ClientHello will rapidly reduce the transmit credits of the sender or fill the physical medium. This is true for both the master and the slave M-Bus device (gateway and OED). 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, triggered by unnecessary ClientHellos.

To take measures against these problems the TLS role “client” is assigned to the OED and the TLS role “server“ is assigned to the gateway.

The OED shall only accept TLS ChannelRequest from the gateway if the TLS ChannelRequest received is AFL CMAC authenticated and verified.

The TLS channel starts, when the client (OED) sends a TLS ClientHello handshake message to the server (gateway) (TLS 1st MsgBlock). The server (gateway) shall only accept ClientHello from the client (OED) if the received ClientHello is AFL CMAC authenticated and verified.

The server (gateway) answers with ServerHello and ServerHelloDone within a TLS message with typically more than 400 bytes length (TLS session not resumed). According to RFC[5246], the server (gateway) shall answer with ServerCertificate, ServerKeyExchange and ClientCertificateRequest, if the related requirements of the TLS configuration are fulfilled.

Any existing TLS channel between server (gateway) and client (OED) shall be silently closed in the client (OED), after receiving an authenticated and verified TLS ChannelRequest from the server (gateway). Any existing TLS channel between server and client shall be silently closed in the server, before sending an authenticated TLS ChannelRequest to the client. A TLS channel shall be identified by the Application Layer Address ADDR of the server (consists of ID, manufacturer, version, device type).

NOTE: The following message flights uses a SND-UD for data transfer to the OED. The gateway may also apply a SND-UD2 for non-fragmented wireless M-Bus messages (see [OMS-S2], 5.2.4).

F.3.5.2 TLS session with full handshake, initiated by TLS server (gateway)

Table F.17 – TLS session with full handshake, initiated by TLS server (gateway)

Gateway (LLA = COM)	**Message flight**	**OED (LLA = MTR)**
TLS ChannelRequest to enter frequent access cycle with new TLS channel	SND-UD(C = 73h); CI = 8Ch; CI = 90h; CMAC; CI = 5Fh; MTR; CF(MODE 13; TLSCHAN) → ACK(C = 00h); CI = 8Ch; CI = 8Ah ←	The client enters frequent access cycle and requests a new TLS–channel. Waits for REQ-UD2 from server to send ClientHello
The server requests to receive	REQ-UD2(C = 5Bh); CI = 8Ch; CI = 80h; MTR → RSP-UD(C = 08h); CI = 8Ch; CI = 90h; CMAC; CI = 9Eh; ← Flight 1	Client sends ClientHello (without SessionID)
Server sends ServerHello, Certificate, ServerKeyExchange, CertificateRequest and ServerHelloDone in fragmented messages.	SND-UD(C = 73h); CI = 8Eh; CI = 90h; FragID = 1; TLS RecLen; CI = 5Fh; CF(MODE 13; TLSHS) → ACK(C = 00h); CI = 8Eh ← SND-UD(C = 53h); CI = 8Eh; CI = 90h; FragID = 2; → ACK (C = 08h); CI = 8Eh; CI = 8Ah ← Flight 2 [a]	
Server requests to receive	REQ-UD2(C = 7Bh); CI = 8Ch; CI = 80h → RSP-UD(C = 08h); CI = 8Eh; CI = 90h; FragID = 1; TLS RecLen; CI = 9Eh; CF(MODE 13; TLSHS) ← REQ-UD2(C = 5Bh); CI = 8Eh → RSP-UD(C = 08h); CI = 8Eh; CI = 90h; FragID = 2 ← Flight 3 [a]	Client sends Certificate, ClientKeyExchange, CertificateVerify ChangeCipherSpec, Finished
Server sends ChangeCipherSpec, Finished.	SND-UD(C = 73h) ; CI = 8Ch ; CF(MODE 13; TLSHS) → ACK(C = 00h); CI = 8Ch; CI = 8Ah ← Flight 4	

[a] Due to the length of the certificate it might be necessary to use a third fragment.

All numbers are hexadecimal.

F.3.5.3 Resumed TLS session, initiated by TLS server (gateway)

Table F.18 – Resumed TLS session, initiated by TLS server (gateway

Gateway (LLA = COM)	**Message flight**	**OED (LLA = MTR)**
Optional immediate SND-UD (for OEDs with receiver always on) The gateway sends TLS ChannelRequest	SND-UD(C = 53h); CI = 8Ch; CI = 90h; CMAC; CI = 5Fh; CF(MODE 13; TLSCHAN)	Since the receiver is not always open, the message is not received.
	SND-NR(C = 44h); CI = 7Ah	OED sends periodic, encrypted data to gateway. Payload AES encrypted and opens receiver according to radio mode timings e.g. after 100 to 1000 ms for Mode C.
SND-UD to enter the frequent access cycle with a TLS ChannelRequest	SND-UD(C = 73h); CI = 8Ch; CI = 90h; CI = 5Fh; CF(MODE 13; TLSCHAN) ACK(C = 00h); CI = 8Ah	The OED enters frequent access cycle and requests a new TLS channel. Waits for REQ-UD2 from gateway to send ClientHello
	REQ-UD2(C = 5Bh); CI = 8Ch; CI = 80h RSP-UD(C = 08h); CI = 8Ch; CI = 8Eh CF(MODE 13; TLSHS) Flight 1	OED sends ClientHello with SessionID
Gateway sends a ServerHello with SessionID (reflects the SessionID received in the ClienHello message) and ChangeCipherSpec, finished in one message.	SND-UD(C = 73h); CI = 8Ch; CI = 5Fh; CF(MODE 13; TLSHS) ACK(C = 00h); CI = 8Ah Flight 2	
Gateway requests to receive	REQ-UD2(C = 5Bh); CI = 8Ch; CI = 80h RSP-UD(C = 08h); CI = 8Ch; CI = 9Eh; CF(MODE 13; TLSHS) Flight 3	OED sends ChangeCipherSpec, Finished

All numbers are hexadecimal.

In case of TLS session resumption, the server shall ignore the supported elliptic curves extension and the supported point formats extension appearing in the current ClientHello message.

F.3.5.4 Exchange of application data

Table F.19 – Exchange of application data

Gateway (LLA = COM)	**Message flight**	**OED (LLA = MTR)**
TLS encapsulated request	SND-UD(C = 73h); CI = 8Ch; CI = 5Bh; CF(MODE 13; TLSAPP) → ACK(C=00h); CI=8Ch; CI=8Ah; CF=0 ← Flight *x*	
	or REQ-UD2(C = 53h); CI = 8Ch; CI = 80h → RSP-UD(C = 08h); CI = 8Ch; CI = 72h CF(MODE 13; TLSAPP) ← Flight *y*	Send TLS encapsulated response

All numbers are hexadecimal.

F.3.5.5 Terminate TLS channel

Because the TLS channel may be silently terminated (closed) by either client or server (for example with a reboot and power-loss) the client shall accept an (authenticated) TLS ChannelRequest to close an existing TLS channel to the sender of the TLS ChannelRequest and initiate a new TLS channel by sending a ClientHello to the sender of the TLS ChannelRequest.

The wireless M-Bus does not define a connection-oriented layer. Therefore, termination of the TLS channel cannot close the underlying connection layer. TLS channel terminated by server:

Table F.20 – TLS channel terminated by server

Gateway (LLA = COM)	**Message flight**	**OED (LLA = MTR)**
Sends TLS CloseNotify	SND-UD(C = 73h); CI = 8Ch; CI=5Fh; CF(MODE 13; TLSALERT) → ACK(C = 00h); CI = 8Ch; CI = 8Ah ← Flight *n*-3	
Requests close notify from client	REQ-UD2(C = 5Bh); CI = 8Ch; CI = 80h → RSP-UD(C = 08h); CI = 8Ch; CI = 9Eh; CF(MODE 13; TLSALERT) ← Flight *n*-2	Send TLS CloseNotify
At the end of communication, the DLL may terminate the Frequent Access Cycle	SND-NKE(C = 40h); CI = 8Ch; CI=80h → Flight *n*-1	

All numbers are hexadecimal.

F.3.6 OED behaviour for different TLS channel states

F.3.6.1 OED messages outside the TLS channel

The OED needs to transmit periodically messages to allow an access to the OED (see [OMS-S2], 4.3.3). This can be message type SND-NR or ACC-NR (see [OMS-S2], 5.2.4). Both message types are transmitted unsolicited with no acknowledgement and may be lost. For that reason, the unsolicited OED message shall not be transmitted with security profile C (not within the TLS channel).

F.3.6.2 TLS channel usage for OEDs conforming to Annex E.1

If an OED or gateway complies with [OMS-S2], Annex E clause E.1 it need to transfer all application data via the TLS channel (security profile C, see [OMS-S2], 9.1).

Under certain conditions a gateway may also accept unidirectional messages from OEDs with security profile B (see “LKS2” in [BSITR03109-1]) when the TLS channel is not available. For that reason, the OED shall transmit a SND-NR or ACC-NR periodically in synchronised transmission slots (according to [OMS-S2], 4.3.2.1). The SND-NR shall contain simple consumption data (at least the current consumption values). It shall be protected with security profile B (according to [OMS-S2], 9.1). The alternative ACC-NR (see [OMS-S2], 5.2.4) contains no application data and applies no security profile.

NOTE 1: If the OED applies only the ACC-NR the gateway will never receive consumption data in case of disturbed TLS channel.

All other services like selection and request of data or the acceptance of commands shall be rejected in the case the message was not protected with security profile C (TLS). The OED shall respond an application error in the concerning response (see F.3.6.4).

NOTE 2: This rule requires that a client supports a key pair and a valid certificate by default to enable a bidirectional data transfer with the server. If the security information is not available, they need to be provided using the local service interface of the OED (see F.4).

F.3.6.3 TLS channel usage for OEDs not conforming to Annex E.1

For OEDs which do not need to comply with Annex E, E.1 the use of a TLS channel (security profile C, see [OMS-S2], 9.1) is optional.

F.3.6.4 Behaviour in case of application error

If the client receives a command via the TLS channel and an error has happened, it has to respond an application error. Depending on the error condition, the error code is transmitted either encrypted (within the TLS channel) or plain (outside the TLS channel). The conditions are defined in Table F.21.

Table F.21 – Encryption and authentication of application errors

Error condition	Application error	Encryption of application error	Authentication of application error
Application command has been successfully received via TLS channel but not executed (e.g. command unknown or missing parameters)	Application error according to [EN13757-3:2018], Tab.21	TLS [a]	TLS [a]
Application command has not been successfully received, because decryption or authentication has failed (e.g. wrong key, invalid MessageCounter or TLS handshake has not been finished)	Application error 20h	No [c]	No [c]
Application command has not been successfully received, because the selected security mode is not supported (e.g. wrong security profile)	Application error 21h	No	AFL CMAC [b]
Application command has been successfully received but it was rejected, because it has been received outside the TLS channel.	Application error 22h	No	AFL CMAC [b]

[a] Security profile C
[b] Security profile B
[c] No Security profile

An application error shall not cause a finish of an established TLS session.

NOTE 1: The application errors 20h to 22h describes failure in the transport layer in reason of incorrect keying material or wrong security method. For that reason the security method is omitted when transmitting the command's denial.

NOTE 2: [OMS-S2], 4.3.4 describes how an OED protects against unauthorized communication requests.

NOTE 3: A gateway according [BSITR03109-1] does not accept any message without authentication being checked. The application error may be analysed by a radio logger.

F.4 Update of security information

F.4.1 General

This section describes how to update the security information in the OED and the gateway. It describes how to read and write new security information or how to trigger the generation of new security information in the OED.

The reading of security information from the OED is limited. Private keys and symmetric keys shall not be readable over any external interface of the OED.

The SITP allows transmitting of several blocks within one message. For this Annex the execution shall be applied as a combined transaction (either for all blocks or for none). This means the receiver needs to check all blocks first before executing the first block. If an error is detected in any block then the transaction (of all blocks) shall be rejected.

F.4.2 Master key update by the gateway

The master key update shall be performed using the SITP protocol (see Appendix F.A) over TLS (security mode 13). The new master key will be calculated by the OED with a random z1 (a 128 bit random number) which has to be provided by the gateway.

NOTE: The installation of an already used MK shall be avoided.

F.4.2.1 SITP parameters during Master key update

KeyID

According to [OMS-S2], 9.2.3 the master key applies KeyID = 00h.

KeyVersion

The Initial KeyVersion of the Master key MK_0 after Reset to MK_0 is 00h. For each master key update the new KeyVersion shall be different to the currently active KeyVersion. It is recommended to increment it by 1. The KeyVersion FF_h is used as a wildcard and shall not be assigned as a discrete value.

In the following section MK is the current master key (version is n), MK’ is the new master key (version after increment is n’).

1. Before transferring the random z1, the gateway initiates a TLS channel between OED (TLS client) and gateway (TLS server). The AFL CMAC is required for TLS ChannelRequest and TLS ClientHello is computed using the current MK.
2. The following SITP commands are sent within TLS application records (security mode = 13h)
3. The gateway generates a new random z1 for the OED and stores MK’ without overwriting MK. In case of a reboot of the gateway between step 7 and 10, the MK’ is required.

4. Transfer of the 16 byte random z1 into the OED is done using SITP with the following parameters:

Table F.22 – SITP parameters for master key update, step 4

Parameter	Value	Description
BCF	00_h	Command “Transfer security information”
DSI	01_h	Wrapped data structure according to NIST SP 800-38F type KWP
DSH1	FF_h	KeyID not used (no wrapping applied)
DSH2	FF_h	KeyVersion not used (no wrapping applied)
Key	z1	128 bit random z1
TargetTime	3080000000_h	set to “invalid” as separate activation service is used
KeyID	00_h	master key
KeyVersion	n’ [a]	according MK’

[a] In case, the OED receives a KeyVersion = FFh (unused) the OED computes the new KeyVersion n’ on its own by incrementing n.

5. In case, the OED receives a KeyVersion equal to its current KeyVersion (n’ = n), an SITP status response is sent by the OED and the KeyUpdate is aborted with a permanent failure. The current master key MK is still in use.

Table F.23 – SITP parameters for master key update, step 5

Parameter	Value	Description
BCF	80_h	Response “Transfer security information”
DSI	22_h	Status response structure
DSH1	FF_h	KeyID not used
DSH2	FF_h	KeyVersion not used
Status response byte	21_h	“Unknown or invalid KeyID/KeyVersion”

6. In case, the gateway does not receive a successful SITP status response, the KeyUpdate is aborted with a permanent failure. The current master key MK is still in use.

Table F.24 – SITP parameters for master key update, step 6

Parameter	Value	Description
BCF	80_h	Response “Transfer security information”
DSI	22_h	Status Response structure
DSH1	FF_h	KeyID not used
DSH2	FF_h	KeyVersion not used
Status response byte	00_h	“Successful SITP command”

7. If the transfer has been successful, the OED calculates MK' = CMAC(MK, z1) according to TR-03116-3, 7.1.1 and stores the new master key MK’ under KeyVersion n’, but does not activate the key.

8. Activation of the new master key and deactivation of the old master key is done using SITP with the following parameters:

Table F.25 – SITP parameters for master key update, step 8

Parameter	Value	Description
BCF	04_h	Command “Combined activation/deactivation of security information”
DSI	03_h	Data structure for combined activation/deactivation
DSH1	FF_h	KeyID not used (no wrapping applied)
DSH2	FF_h	KeyVersion not used (no wrapping applied)
TargetTime	3000000000_h	set to zero for an instant activation/deactivation
Activated KeyID	00_h	master key
Activated KeyVersion	n’ [a]	according MK’
Deactivated KeyID	00_h	master key
Deactivated KeyVersion	FF_h	Unused
Option	01_h	perform a MessageCounter reset

[a] If the gateway sets the activated KeyVersion to FFh (unused), the detection of the new master key MK’ and the according KeyVersion n’ has to be done by the OED.

9. The OED tries to activate KeyID = 00h with version n’ and in an atomic operation deactivates KeyID = 00h with version n.
10. In case the gateway does receive a failure, the KeyUpdate is aborted with a permanent failure. The current master key MK is still in use.
11. In case the gateway does receive a successful response, the activation has been successful and gateway and OED immediately use MK’ as the new master key for the next TLS ChannelRequestChannelRequest. If KeyVersion of deactivated MK is >0, MK can be deleted. If KeyVersion is = 0, the initial MK_0 is preserved in the OED and its associated current MessageCounter is stored persistent in the OED.

F.4.2.2 Synchronization of state

In case the gateway does not receive any response after the activation/deactivation or if the update process has been interrupted between Step 7 and Step 10, the gateway may probe, if communication with the OED is possible by using a message with AFL CMAC using KMAC derived from MK or MK’. If communication using MK is not possible, the probe is done with MK’, which must succeed.

F.4.3 Update of OED certificate and KeyPair

F.4.3.1 General

The OED holds an individual certificate (MTR_TLS_CRT) with associated KeyPair (MTR_TLS_PUB/MTR_TLS_PRV) for TLS authentication and (ECDHE) key agreement. The initial OED certificate and KeyPair is generated and stored in the OED by the manufacturer. OMS declares the assignment of the MTR_TLS_CRT to the KeyID = 90h and the associated KeyPair to the KeyID = A0h. That means the PublicKey in MTR_TLS_CRT is identical to the PublicKey of the KeyPair (MTR_TLS_PUB). This association is fix and cannot be released, i.e. a kind of rekeying of a certificate is not allowed. For an activation always both components shall be available and valid.

Each OED certificate, either stored in the OED by the manufacturer (initial certificate) or generated by the gateway and transferred to the OED over a confidential TLS channel, shall contain a valid date of issue. The date of issue is identified by the “Validity.NotBefore” parameter (see an example in Appendix F.F). The certificate may be self-signed or Issued by a CA (this case is not described here).

The OED shall be able to manage exact two pairs of credentials - each contains the MTR_TLS_CRT and the associated KeyPair. They are defined as:

- ActiveCredential - the one currently in use
- NewCredential – the one which is intended to replace the active one

There can only be one ActiveCredential and one NewCredential in the OED. If a new certificate or the KeyPair is transmitted or generated it shall be stored in the NewCredential. During the activation process the NewCredential shall be validated in terms of:

- Syntactical correctness of the data structures
- PublicKey of MTR_TLS_CRT and KeyPair shall be equal
- The date of issue of the NewCredential (contained in MTR_TLS_CRT, see above) shall be newer than the one of the ActiveCredential (if ActiveCredential present).
- The Subject CommonName of the NewCredential shall be the same as the Subject CommonName of the ActiveCredential.
- On initial installation (no ActiveCredential present), the Subject CommonName of the NewCredential shall unambiguously identify the OED.
- If the NewCredential issuer name is identical with subject name, the certificate is a (self-signed) root Certificate and the signature shall be verified with the PublicKey of the NewCredential. The BasicConstraints extension shall contain cA = 1 and pathLenConstraint = 0
- If the NewCredential issuer name is different from the subject name, the BasicConstrains extension shall contain cA = 0 and the issuer should be a trusted certificate with BasicConstraint cA = 1 installed in the OED.
- If the extension KeyUsage is present in the certificate, the bits for KeyEncipherment and KeyAgreement should be set for MTR_TLS_CRT.

The installation or renewal of the NewCredential can be realized in three different ways:

- Both are generated by the gateway and are transferred to the OED over a confidential and authentic TLS channel using the SITP-Protocol (see F.4.3.3, with security mode 13)
- If no TLS channel can be established (initial installation) the SITP commands are sent with authentic symmetrical encryption according to OMS security profile B (see F.4.3.3).
- If the OED generates a public/private KeyPair based on a cryptographically strong random number and the private key is kept confidential in the OED, the OED may generate the NewCredential. The gateway retrieves the certificate from the OED over an authentic channel. The private key does not leave the OED (see F.4.3.2).

KeyID

According to OMS definition the MTR_TLS_CRT has KeyID = 90h. Associated to MTR_TLS_CRT is an Elliptic-Curve KeyPair (MTR_TLS_PUB, MTR_TLS_PRV) using KeyID = A0h

KeyVersion

Neither the (MTR_TLS_PUB, MTR_TLS_PRV) KeyPair nor the MTR_TLS_CRT are using a KeyVersion. As explained above only the date of issue of the certificate is relevant. Therefore the KeyVersion parameter of SITP shall be set to FF_h and ignored by the OED.

F.4.3.2 Renewal of KeyPair and certificate by the OED

The renewal of KeyPair and certificate by the OED is not supported in this version.

F.4.3.3 Installation of a new KeyPair and new OED certificate by the gateway

1. Transfer of the KeyPair from the gateway to the OED is done using SITP with the following parameters:

Table F.26 – SITP parameters – Command key transfer

Parameter	Value	Description
BCF	00_h	Command “Transfer security information”
DSI	11_h	Data structure for KeyPair transport which contains MTR_TLS_PUB and MTR_TLS_PRV
DSH1	$A0_h$	KeyID for KeyPair assigned to MTR_TLS_CRT
DSH2	FF_h	KeyVersion not used
Data structure content		KeyPair ASN.1 DER coded according to RFC5958 chapter 2

2. The OED confirms the transfer of the new key (see Table F.27). In case the gateway does not receive a successful SITP status response, the KeyPair update is aborted with a permanent failure. The ActiveCredential is still in use.

Table F.27 – SITP parameters – Key transfer status ok

Parameter	Value	Description
BCF	80_h	Response “Transfer security information”
DSI	22_h	Status response structure
DSH1	$A0_h$	KeyID for KeyPair assigned to MTR_TLS_CRT
DSH2	FF_h	KeyVersion not used
Status response byte	00_h	Successful SITP command

3. The OED stores the new KeyPair into NewCredential, but does not activate it immediately. Activation is done with a separate command after the associated certificate MTR_TLS_CRT is available in the OED.
4. Transfer of the certificate (MTR_TLS_CRT) into the OED is done using SITP with the following parameters:

Table F.28 – SITP parameters – Command certificate transfer

Parameter	Value	Description
BCF	00_h	Command “Transfer security information”
DSI	10_h	Data structure for transporting certificates
DSH1	90_h	KeyID for MTR_TLS_CRT
DSH2	FF_h	KeyVersion not used
Data structure content		Certificates ASN.1 DER coded according to RFC5280 chapter 4

5. The OED confirms the transfer of the new key (see Table F.29). In case the gateway does not receive a successful SITP status response, the certificate update is aborted with a permanent failure. The ActiveCredential is still in use.

Table F.29 – SITP parameters – Certificate status ok

Parameter	Value	Description
BCF	80_h	Response “Transfer security information”
DSI	22_h	Status response structure
DSH1	90_h	KeyID for MTR_TLS_CRT
DSH2	FF_h	KeyVersion not used
Status response byte	00_h	Successful SITP command

6. Activation of the NewCredential is done using SITP with the following parameters:

Table F.30 – SITP parameters – Command activation

Parameter	Value	Description
BCF	01_h	Command “Activate security information”
DSI	02_h	Data structure for activation of security information
DSH1	FF_h	KeyID not used (no wrapping applied)
DSH2	FF_h	KeyVersion not used (no wrapping applied)
TargetTime	3000000000_h	set to zero for an instant activation/deactivation
KeyID	$A0_h$	KeyID for KeyPair assigned to MTR_TLS_CRT
KeyVersion	FF_h	KeyVersion not used - only the NewCredential can be activated

7. If the OED receives an activation command it shall validate the NewCredential (described in F.4.3.1).
    a. If there are any problems the activation is aborted with a permanent failure. In that case the ActiveCredential is still in use.
    If this happen e.g. if the transferred certificate does not match to the transferred KeyPair a SITP status response is sent by the OED.

Table F.31 – SITP parameters – Activation status with failure

Parameter	Value	Description
BCF	84_h	Response “Combined activation/deactivation of security information”
DSI	22_h	Status Response structure
DSH1	FF_h	KeyID not used (no wrapping applied)
DSH2	FF_h	KeyVersion not used (no wrapping applied)
Status response byte	81_h	Mismatch KeyPair and certificate

    b. If the NewCredential is validated correctly the OED transfers the NewCredential to the ActiveCredential, deletes the former ActiveCredential and sends a successful SITP response.

Table F.32 – SITP parameters – Activation status with success

Parameter	Value	Description
BCF	84_h	Response “Combined activation/deactivation of security information”
DSI	22_h	Status response structure
DSH1	FF_h	KeyID not used (no wrapping applied)
DSH2	FF_h	KeyVersion not used (no wrapping applied)
Status response byte	00_h	Successful SITP command

8. In case the gateway receives a negative response, the update process is aborted with a permanent failure. The current ActiveCredential is still in use.
9. If the activation has been successful, gateway and OED use the new ActiveCredential for the next TLS handshake.

F.4.3.4 Synchronization of state

Synchronization could be used in case of doubt about communication state between gateway and OED e.g. after the reboot of the gateway.

The gateway initiates a TLS handshake (Send TLS ChannelRequest, Receive ClientHello, Send ServerHello + ServerCertificate + ServerKeyExchange + ClientCertificateRequest + ServerHelloDone) and waits for the ClientCertificate (MTR_TLS_CRT). If the OED sends the MTR_TLS_CRT that corresponds to the activated LMN_TLS_PUB/PRV KeyPair the synchronization has been successful.

F.4.4 Update of gateway certificate (used for TLS ServerTrust)

F.4.4.1 Transfer of gateway certificate

The gateway holds an individual certificate with associated KeyPair for TLS authentication and (ECDHE) key derivation. An OED should only send data via TLS, after the TLS channel is mutual authenticated, i.e. the presented ServerCertificate matches one of the trust anchors stored in the OED. As there is no initial gateway certificate in the OED, the gateway certificate for mutual authentication has to be installed with the SITP protocol using OMS security profile B.

According to OMS definition the self-signed certificate of the gateway (GWLMN_TLS_CRT) is assigned the KeyID 81h or 82h. The public key for use with TLS (GWLMN_TLS_PUB) is derived by the OED from the gateway certificate GWLMN_TLS_CRT.tbs.subjectPublicKeyInfo. No KeyID is assigned for GWLMN_TLS_PUB.

During the TLS handshake the OED (TLS client) processes the ServerCertificateMessage sent by the gateway. The OED shall terminate the TLS handshake, if the presented ServerCertificate (the first in the ServerCertificateMessage) is not equal to one of the certificates stored under KeyID 81h or 82h.

Update of the OED certificate and gateway certificate is independent from each other and independent from master key update MK.

After changing the OED certificate or gateway certificate, the TLS session keys are still valid. The new certificates come into effect, when the next full TLS handshake is started.

For installation of the initial GWLMN_TLS_CRT no TLS channel can be used. In this case the following SITP commands are sent with symmetrical encryption using the current master key MK according to OMS security profile B. For update of subsequent GWLMN_TLS_CRT a TLS channel shall be used. In this case the following SITP commands are sent according to OMS security profile C (with CF. security mode 13).

NOTE: Typically, the MK0 is active when transferring the initial GWLMN_TLS_CRT.

1. Transfer of the certificate (GWLMN_TLS_CRT) into the OED is done using SITP with the following parameters:

Table F.33 – SITP parameters – Transfer certificate

Parameter	Value	Description
BCF	00_h	Command “Transfer security information”
DSI	10_h	Data structure for transporting certificates
DSH1	81_h or 82_h	KeyID for GWLMN_TLS_CRT
DSH2	FF_h	KeyVersion not used for certificates
Data structure content		Certificates ASN.1 DER coded according to RFC5280 chapter 4

2. In case the gateway does not receive a successful SITP response, the certificate update is aborted with a permanent failure. The currently active GWLMN_TLS_CRT is still in use.
3. Activation of the GWLMN_TLS_CRT is done using SITP with the following parameters:

Table F.34 – SITP parameters – Command activation

Parameter	Value	Description
BCF	01_h	Command “Activate security information”
DSI	02_h	Data structure for activation of security information
DSH1	FF_h	KeyID not used (no wrapping applied)
DSH2	FF_h	KeyVersion not used (no wrapping applied)
TargetTime	3000000000_h	set to zero for an instant activation/deactivation
KeyID	81_h or 82_h	KeyID for KeyPair assigned to MTR_TLS_CRT
KeyVersion	FF_h	KeyVersion not used - only the NewCredential can be activated

4. For an activation of the new GWLMN_TLS_CRT the validations (syntactical correctness and date-of-issue check) shall be done successfully. If there are any problems the activation is aborted and a SITP status response (containing the error) is sent by the OED.
5. In case the activation was successful the new certificate replaces the current one with the identical KeyID in the OED (e.g. both have 81_h). Gateway and OED use the certificate for verification of the ServerCertificate in the next (full) TLS handshake. A positive SITP status response is sent by the OED.

F.4.4.2 Synchronization of state

Upon TLS ChannelRequest from gateway to OED, the OED sends ClientHello to the gateway and receives a ServerCertificateMessage containing the GWLMN_TLS_CRT of the gateway. If the first certificate of the ServerCertificateMessage is not found in the OED under KeyID 81h-82h, no communication with the gateway is allowed and a reset to MK_0 is required.

Alternative: The gateway uses SITP with security profile B to read the certificates of KeyID = 81h or 82h with the following parameters:

Table F.35 – SITP parameters – Command “Get security information”

Parameter	Value	Description
BCF	06h	Command “Get security information”
DSI	00h	Empty structure used for requesting security information
DSH1	81h or 82h	KeyID for GWLMN_TLS_CRT
DSH2	FFh	KeyVersion not used for certificates

The response will deliver the currently active certificates in the OED, e.g.:

Table F.36 – SITP parameters – Response ‘Get security information’

Parameter	Value	Description
BCF	86_h	Response “Get security information”
DSI	10_h	Data structure for transporting certificates
DSH1	81_h	KeyID for GWLMN_TLS_CRT
DSH2	FF_h	KeyVersion not used for certificates
Data structure content		Certificates ASN.1 DER coded according to RFC5280 chapter 4

F.4.5 Reset to MK_0

Using physical access to the OED, the master key may be restored to MK_0 (KeyID = 0, KeyVersion = 0). Such a reset of MK_0 shall lead to the deletion of all other master keys with KeyID = 0 and KeyVersion>0 and any GWLMN_TLS_CRT with KeyID 81h-82h. Restoring the master key MK_0 also restores the associated initial MessageCounter C_{M0} (see [OMS-S2], 9.3.2.3).

NOTE: As the initial MessageCounter C_{M0} is not resettable, the MK_0 should be replaced by a new MK after establishing a TLS-communication to the OED (see also [OMS-S2], 9.3.2.3).

F.5 TLS-Certificates for OEDs

An OED shall be able to handle (i.e. process and store) X.509v3 Certificates (according to RFC5280) DER-encoded. Details are shown in Table F.37. See also an example in Appendix F.F.

NOTE: The [BSITR03109-1] can request additional parameters than shown here.

Table F.37 – TLS certificate details

Parameter	Description
Certificate Size	Certificate Size up to 500 Bytes minimum sufficient for EC key sizes up to brainpoolP512r1 without further certificate extensions
PublicKey	Uncompressed PublicKey of Type id-ecPublicKey according to RFC5480 Chap. 2.2
Curves [a]	• Required: brainpoolP256r1, NIST P256, brainpoolP384r1 • Recommended: brainpoolP512r1, NIST P384
Digest Algorithm [a]	Required: SHA256, SHA384, SHA512
Signature Algorithm [a]	Required: ECDSA
SerialNumber	Length 4..20 bytes
BasicConstraintsExtension (critical)	• cA = 1 • pathLenConstraint = 0
KeyUsage Extension (critical)	DigitalSignature Required
Distinguished Name (DN)	• Selfsigned, i.e. issuerDN = subjectDN • commonName (case Insensitive): accept 1..64 characters of ASN.1 type “Printable string”. Currently national format empty or according to DIN43863-5:2012-04 without spaces, appended by .mtr/.MTR. Example “7mfc0100001234.mtr”. • serialNumber attribute (OID 1.3.6.1.4.1.1466.115.121.1.44) accept Sequence-Number of certificate • Currently no other DN-attributes are present, but certificate should not be rejected, if they are present. • For a transition period also certificates with empty issuerDN, subjectDN (no attributes) must be accepted.
Validity	• notBefore: UTCTIME and GENERALIZED TIME • notAfter: UTCTIME and GENERALIZED TIME
Other extensions	Currently no other extensions are present, but certificate should not be rejected, if they are present.

[a] Deprecation and announcement of supported Algorithms and key sizes follows annually updated BSI TR-03116-3.

Appendix F.A (normative): Security Information Transfer Protocol (SITP)

F.A.1 Introduction

This specific application protocol is intended for all kind of security information handling and management of security relevant services in a metering system. The main use case is to update key information. Handling of security information requires bidirectional access to the OED. This annex extends the SITP in [EN13757-7:2018], Annex A by services for asymmetric crypto methods such as TLS.

In detail the SITP provides the following services:

- Transfer security information (to OED)
- Activate security information
- Deactivate security information
- Combined activation/deactivation of security information
- Generate security information
- Get security information (from OED)
- Get list of all security information (from OED)
- Get list of active security information (from OED)
- Transfer end to end secured application data

The usage of this upper layer protocol is introduced with a specific sub-range of CI-fields. It is therefore independent from the M-Bus application layer protocol.

Two examples which show the usage of the SITP are provided in the CEN TR 17167; F.7.

F.A.2 SITP services

F.A.2.1 Transfer security information

The purpose of this service is to transfer security information (e.g. security keys) from a communication partner to an OED in a secure way. The security information itself is often generated by a backend system and not by the communication partner. This service provides, in addition to security service from the lower layers a wrapping mechanism. The wrapping mechanism ensures a strong binding between the key and the set of data specifying the use of the key. The transferred security information contains a target time that gives an absolute or relative timestamp of usage. The target time may as well be given by the separate activation or deactivation service described below.

NOTE: This service gives no advice on how to handle existing security information when new information is received. Such handling is a part of the security policy of the whole system.

F.A.2.2 Activate security information

This service shall not be used for the OMS. Use “Combined activation/deactivation of security information” instead.

F.A.2.3 Deactivate security information

This service shall not be used for the OMS. Use “Combined activation/deactivation of security information” instead.

F.A.2.4 Destroy security information

The purpose of this service is to destroy (and delete) specific security information formerly used by the OED. It avoids possible security issues with old security information material. The OED shall not allow destroying active security information.

F.A.2.5 Combined activation/deactivation of security information

The purpose of this service is to combine the action of activation and deactivation. It is needed in case of a “message counter sharing” between an old and a new key. As a new key requires a reset of the MessageCounter the old key cannot be used with this counter value zero anymore. To avoid security issues in that situation both actions have to take place simultaneously.

F.A.2.6 Generate security information

The purpose of this service is to instruct the OED to create security information (e.g. a new key pair, a certificate or a certificate signing request “CSR”). The security information can be requested by a separate command.

F.A.2.7 Get security information

The purpose of this service is to get single security information from the OED. This can be used to ask for a certificate, a CSR, the public key or applied key id, key version of this OED.

F.A.2.8 Get list of all key information

The purpose of this service is to get a list of all used key id and the available key versions stored in an OED.

F.A.2.9 Get list of active key information

The purpose of this service is to get a list of all active key id, their key versions and the key counter, which are currently valid.

F.A.2.10 Transfer end to end secured application data

The purpose of this service is to securely transfer an application layer protocol message between a remote command initiator and an OED. The application data are integrity and maybe privacy protected by usage of a key which is only known to the OED and the command initiator. An intermediary will not be able to alter this message as the end parties are able to detect this modification. This allows an end to end secured transfer of critical commands or critical data.

F.A.3 CI-fields

Commands and responses for the Security Information Transfer Protocol (SITP) are identified by specific values of the CI-field. The applicable CI-fields are listed in the Table A.1:

Table F.A.1 — SITP CI-fields

CI-field	Designation	Header	Remarks
C3h	Command to device	Long	Security Information Transfer Protocol
C4h	Response from device	Short	Security Information Transfer Protocol
C5h	Response from device	Long	Security Information Transfer Protocol

F.A.4 SITP structure

The SITP structure is implemented as APL data (see EN 13757-7:2018, Table 10 and [OMS-S2], 8.1). It provides the possibility to transfer multiple blocks of security information in one message. Each block contains a single SITP command or SITP response (see Table F.A.3). The block is identified by a block ID which shall be used for a correlation between commands and responses. The function of each block is specified by the block control field. Each command and each response shall have a principal structure as depicted in the table below:

Table F.A.2 - Internal block structure of SITP

Block length (BL)	Block ID field (BID)	Block control field (BCF)	Block parameters
2 bytes	1 byte	1 byte	*n* bytes

The elements are:

Block length: BL, Length of this block including BID, BCF and all block parameters (LSB first). BL = 0 signals no more blocks to follow

Block ID field: BID, Identifier of this block, start with first block BID = 0

Block control field: BCF, coding of the usage of this command or response, see Table F.A.5

Block parameters: Variable parameters depending on the BCF, see F.A.6

Block length is a multi-byte field. Block parameters may also contain multi byte fields. If nothing else is declared then multi byte fields shall be transmitted with the least significant byte first (little endian).

F.A.5 Block control field

The block control field gives the function of the block and shall have one of the listed values for command or response as shown in the table below. There is a fixed relation between a certain command and its assigned response, e.g. command 03_h requires a response with 83_h.

Table F.A.3 - Block control field

Command	Applicable DSI	Function	Response	Applicable DSI
00_h	01_h, 10_h, 11_h	Transfer security information (see F.A.2.1)	80_h	22_h
01_h	02_h	Activate security information (see F.A.2.2)	81_h	22_h
02_h	02_h	Deactivate security information (see F.A.2.3)	82_h	22_h
03_h	02_h	Destroy security information (see F.A.2.4)	83_h	22_h
04_h	03_h	Combined activation/deactivation of security information (see F.A.2.5)	84_h	22_h
05_h	02_h	Generate security information (see F.A.2.6)	85h	22_h
06_h	00_h	Get security information (see F.A.2.7)	86h	10_h, 12_h, 13_h, 22_h,
07_h	00_h	Get list of all key information (see F.A.2.8)	87h	20_h, 22_h,

Command	Applicable DSI	Function	Response	Applicable DSI
08_h	00_h	Get list of active key information (see F.A.2.9)	88h	21_h, 22_h,
09_h [b]	00_h	Get list of active keys and key counter information (see F.A.2.9)	89h	22_h, 23_h [b],
$0A_h$ to $1F_h$		Reserved for future use	$8A_h$ to $9F_h$	
20_h	30_h to 37_h	Transfer end to end secured application data (see F.A.2.10)	$A0_h$	22_h, 30_h to 37_h,
21_h to $6F_h$		Reserved for future use	$A1_h$ to EF_h	
70_h to $7F_h$	$F0_h$ to FF_h [a]	Manufacturer specific usage	$F0_h$ to FF_h	$F0_h$ to FF_h [a]

[a] All other DSI can be used as well
[b] These values are not yet covered by [EN13757-7:2018]

F.A.6 Block parameters

The structure of the block parameter is defined in Table F.A.4. It is applied for all SITP commands and responses. In case of status responses this structure is also used but the data structure content will be different (see Table F.A.5).

The structure is designed to transport applicative data in the “Data structure content”. The other fields serve as organisational and introductive information. The RecipientID and the DSH shall have the same values in a command and the respective response. This is essential for a correct assignment of the response by the command initiator.

Table F.A.4 - Block parameter structure

RecipientID	Data structure identifier (DSI)	Data structure header (DSH)		Data structure content
		DSH1	DSH2	
1 byte	1 byte	1 byte	1 byte	*n* bytes

Recipient ID: Identifier of the application. Set to 00h if no dedicated application is addressed.

NOTE: For example to address the gateway function in a combined electricity meter. In the TCP world this would be the port number.

Data structure identifier: DSI, identifying the applied data structure declared in Table F.A.5

Data structure header: DSH introduces the data structure. Its meaning depends on the data structure identifier. DSH1 and DSH2 are autonomous 1-byte fields as shown in Table F.A.5.

Data structure content: Structured security information according to F.A.7

F.A.7 Overview about data structures / mechanisms

This chapter informs about the data structures, the applied mechanisms and the usage of the DSH. The “data structure content” as shown in Table F.A.4 is provided in the dedicated sub clauses (e.g. data structure 01_h) or defined by the data structure itself (e.g. CSR according to RFC4211 chapter 3). In this case no further explanations about the data structure content are given in concerning section.

There are two different groups of data structures defined which can be seen in Table F.A.5. One group is mainly intended for handling and management of security information and their relevant services (DSI 00_h to $2F_h$, see F.A.8). The other group is defined for the use case of secured application data transfer (DSI $3x_h$, see F.A.9). Table F.A.5 shows the list of data structures / mechanisms, the usage of DSH and the applicable BCF values. Details about the DSH values can be found below the table.

Table F.A.5 – List of SITP data structures / mechanisms

Data Structure Identifier (DSI)	Data structure / Mechanism	Data structure header (DSH)	
		DSH1	DSH2
00_h	Empty structure / no wrapping (see [EN13757-7:2018], Annex A, A.8.2)	KeyID	KeyVersion
01_h	Wrapping AES128 according NIST SP 800-38F type KWP (see [EN13757-7:2018], Annex A, A.8.3)	WrapperKey ID	WrapperKey Version
02_h	Wrapping AES128 according NIST SP 800-38F type KWP (see [EN13757-7:2018], Annex A, A.8.4)	WrapperKey ID	WrapperKey Version
03_h	Wrapping AES128 according NIST SP 800-38F type KWP (see [EN13757-7:2018], Annex A, A.8.5)	WrapperKey ID	WrapperKey Version
04_h to $0F_h$	Reserved for future use	n/a	n/a
10_h	Certificates ASN.1 DER coded according to RFC5280 chapter 4	KeyID	unused [a]
11_h	KeyPair ASN.1 DER coded according to RFC5958 chapter 2	KeyID	unused [a]
12_h	Public keys ASN.1 DER coded according to subject public key info (RFC5480 chapter 2) [c]	KeyID	unused [a]
13_h	CSR according to RFC4211 chapter 3	KeyID	unused [a]
14_h to $1F_h$	Reserved for asymmetric technologies	n/a	n/a
20_h	All keys structure wrapped with AES128 according NIST SP 800-38F type KWP (see [EN13757-7:2018], Annex A, A.8.6.)	WrapperKey ID	WrapperKey Version
21_h	Active keys structure ([EN13757-7:2018], Annex A, A.8.7)	unused [a]	unused [a]
22_h	Status response structure (see [EN13757-7:2018], Annex A, A.8.8)	KeyID [b]	KeyVersion [b]
23_h [d]	Active key counter structure (F.A.8)	unused [a]	unused [a]
24_h to $2F_h$	Reserved for future use	n/a	n/a
30_h	Wrapping AES128 according to NIST SP 800-38F type KWP ([EN13757-7:2018], Annex A, A.9.2)	WrapperKey ID	WrapperKey Version
31_h	Authentication with HMAC SHA256 (see [EN13757-7:2018], Annex A, A.9.3)	WrapperKey ID	WrapperKey Version
32_h	Authentication with AES128 CMAC (8 byte MAC, see [EN13757-7:2018], Annex A, A.9.4)	WrapperKey ID	WrapperKey Version
33_h	Authentication with AES128 CMAC (16 byte MAC, see [EN13757-7:2018], Annex A, A.9.4)	WrapperKey ID	WrapperKey Version
34_h	Authenticated encryption with AES128-GCM (see [EN13757-7:2018], Annex A, A.9.5)	WrapperKey ID	WrapperKey Version
35_h	Authentication with AES128 GMAC (see [EN13757-7:2018], Annex A, A.9.6)	WrapperKey ID	WrapperKey Version
36_h	Authenticated encryption with AES128 CCM (8 byte MAC, see [EN13757-7:2018], Annex A, A.9.7)	WrapperKey ID	WrapperKey Version
37_h	Authenticated encryption with AES128-CCM (16 byte MAC, see [EN13757-7:2018], Annex A, A.9.7)	WrapperKey ID	WrapperKey Version
38_h to EF_h	Reserved for future use	N/A	NA
$F0_h$ to FF_h	Manufacturer specific	N/A	N/A

[a] Set unused fields to FF_h
[b] The same DSH value than used in command
[c] This chapter extends RFC5280 4.1.2.7
[d] This value are not yet covered by [EN13757-7:2018]

Contents of data structure header (DSH):

WrapperKeyID: ID of the key used for wrapping or authentication according the KeyID definition. If not used (no wrapping applied) set to FFh. Note that only KeyID in the range from 00h to 0Fh are used for TPL security (see Table 24). WrapperKeys used for the APL security shall use a KeyID larger than 0Fh.

WrapperKeyVersion: Version of the key used for wrapping. If not used set to FFh

KeyID: The ID of the intended key. If not used set to FFh.
OMS declares specific meaning for certain KeyID.

Table F.A.6 - Predefined OMS KeyID

KeyID	Type of Security information
00_h to $7F_h$	see [OMS-S2]; 9.2.2
80_h	n/a
81_h to 82_h [a]	Trust anchor certificate (GWLMN_TLS_CRT)
83_h to $8F_h$	Reserved for Trust anchor certificate
90_h	Device specific TLS server authentication certificate (MTR_TLS_CRT)
91_h	Application data signature certificate (MTR_SIG_CRT)
92_h to $9F_h$	n/a
$A0_h$	KeyPair assigned to MTR_TLS_CRT
$A1_h$	KeyPair assigned to MTR_SIG_CRT
$A2_h$ to AF_h	n/a
$B0_h$ to FF_h	see [OMS-S2], 9.2.2

[a] A second trust anchor may be necessary for a fail-safe transfer of authority from one SMGW to another SMGW with a direct trust concept.

KeyVersion: The version of the intended KeyID. If not used set to FFh.

F.A.8 Data structures for security information

The OED shall apply data structures for security information according [EN13757-7:2018], Annex A, A.8.

For the Data structure 22_h (status response) the values of [EN13757-7:2018], Annex A, table A.13 are extended by following table.

Table F.A.7 – Extension of status response byte definition

Status response byte	Explanation
09_h	This SITP block was not executed due to an error in another SITP block
$0A_h$ to $0F_h$	Reserved for status information (no error)
…	…
11_h	Unknown or invalid command (BCF error)
12_h to 13_h	Reserved for data structure header errors
…	…
80_h	Unspecified certificate error
81_h	Mismatch KeyPair and certificate
82_h to $8F_h$	Reserved for asymmetric usage

Data structure 23_h

The Data structure 23_h extend the SITP in [EN13757-7:2018], with a missing functionality. This data structure is used to report the current KeyCounter values used in 0. The KeyID, the KeyVersions and the stored KeyCounter will be transferred, not the keys itself. This structure provides the possibility to ask for the current KeyCounter value per KeyID. It is not wrapped/protected as the KeyID and KeyCounter are no secret information. The length of the whole structure is variable, as the number of used KeyID is not defined as shown in Table F.A.8 (the maximum length is 256 * 6 bytes).

Table F.A.8 – Active key counter structure 23_h

KeyID	KeyVersion	KeyCounter
1 byte	1 byte	4 byte
…	…	…
1 byte	1 byte	4 byte

KeyID: 1 byte value of the reported key id. Only key id which has an activekey verison shall be given here.

KeyVersion: 1 byte value for the active key version of this key id. If no key versioning used set to FFh

KeyCounter: 4 byte KeyCounter (last stored one) of the respective key id with LSB first.

F.A.9 Data structures for secured application data

For the secured transfer of application protocols with critical commands or critical data the OED may apply data structures according to [EN13757-7:2018], Annex A, A.9:

- Data Structure 30h – AES key wrap (encryption and authentication) or
- Data Structure 32h and 33h – CMAC (authentication only).

The use of other data structures for secured application data is optional and will not be checked by the OMS-CT.

Appendix F.B (informative): Examples for the usage of AFL and TLS

This chapter shows several examples of messages using AFL and TLS.

Table F.B.1 – Simple secured unfragmented M-Bus message

DLL header (EN 13757-4): Len,MsgType Addr	Long Transport App header CI = 72h, ADDR, ACC, ST, CF = MODE05	Application data [OMS-S2] M-Bus DIF/VIFs

Table F.B.2 – Non-fragmented authenticated M-Bus message

DLL header (EN 13757-4): Len,MsgType Addr	Extended Link Layer CI = 8Ch CC, ACC [OMS-S2]	Authentication and Fragmentation Layer CI = 90h AFLLen, AFLFlags, CTR, MAC	Long Transport header CI = 72h, ADDR, ACC, ST, CF = MODE07	Application data [OMS-S2]

Table F.B.3 – Fragmented authenticated application message (1st fragment)

DLL header (EN 13757-4): Len,MsgType, Addr	Extended Link Layer CI = 8Eh, CC, ACC, ADDR [OMS-S2]	Authentication and Fragmentation Layer CI = 90h, AFLLen, AFLFlags(MF = 1), TLSSDULen, CTR	Short Transport header CI = 7Ah, ACC, ST, CF = MODE07	Application data [OMS-S2]

Table F.B.4 – Fragmented authenticated application message (last fragment)

DLL header (EN 13757-4): Len,MsgType, Addr	Extended Link Layer CI = 8Eh, CC, ACC, ADDR [OMS-S2]	Authentication and Fragmentation Layer CI = 90h, AFLLen, AFLFlags(MF = 0), MAC	Application data [OMS-S2]

Table F.B.5 – Unfragmented TLS secured application command message

<table>
<tr><td rowspan="2">DLL header
(EN 13757-4):
Len,MsgType,
Addr</td><td rowspan="2">Extended Link Layer
CI = 8Ch CC, ACC
[OMS-S2]</td><td rowspan="2">Long Transport header
CI = 5Bh, ADDR, ACC, ST, CF = MODE13
[OMS-S2]</td><td colspan="3">TLS record
(includes application data)</td></tr>
<tr><td>TLS HDR, IV</td><td>M-Bus application data
[OMS-S2]</td><td>HMAC/
Padding</td></tr>
</table>

Appendix F.C (informative): Master key renewal

This Appendix shows the key renewal procedure as described in F.4.2.

Figure F.C.1 – Key renewal procedure – Overview

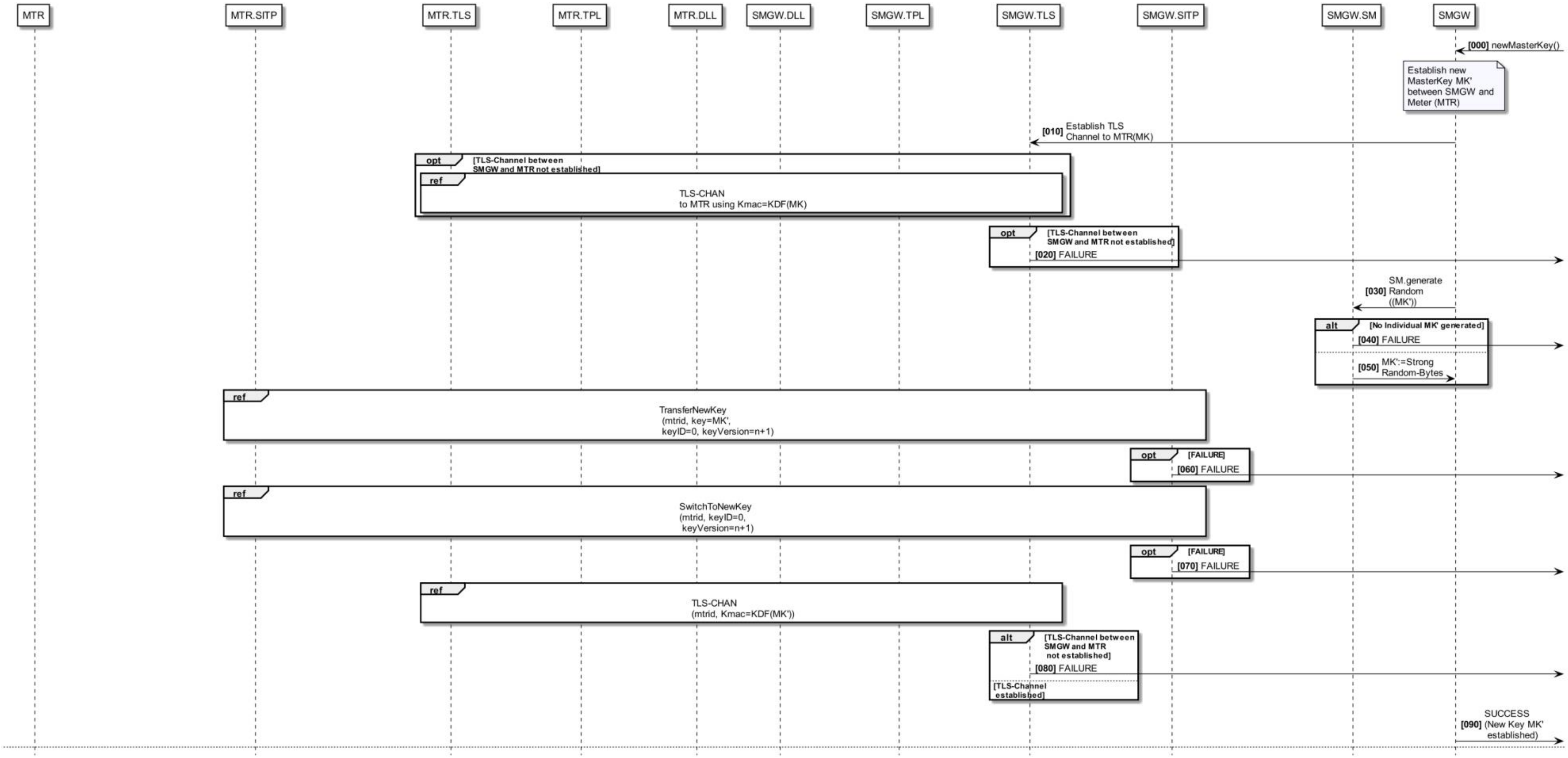


Figure F.C.2 – Key renewal procedure – Key transfer

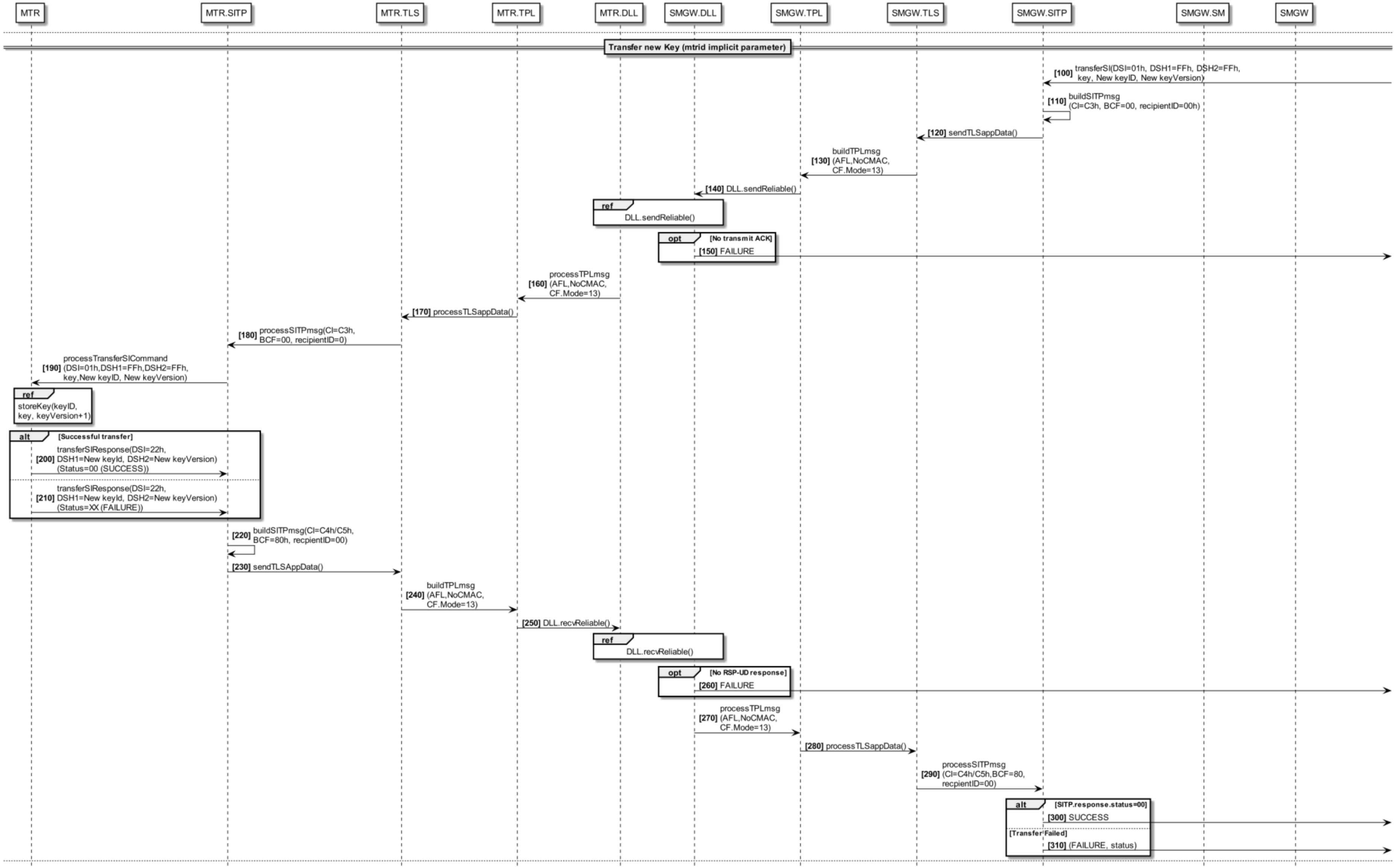


Figure F.C.3 – Key renewal procedure – Key activation/deactivation

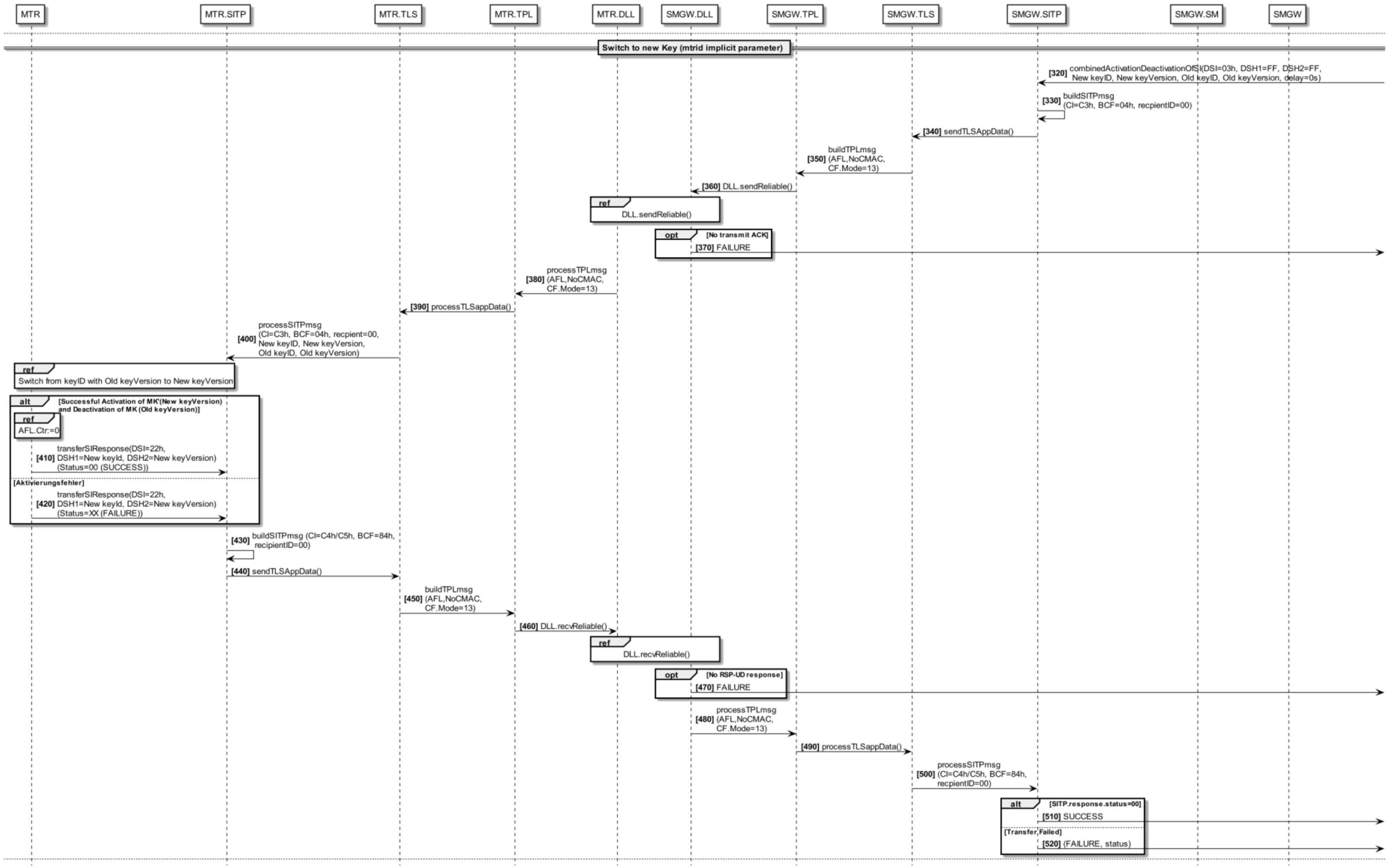
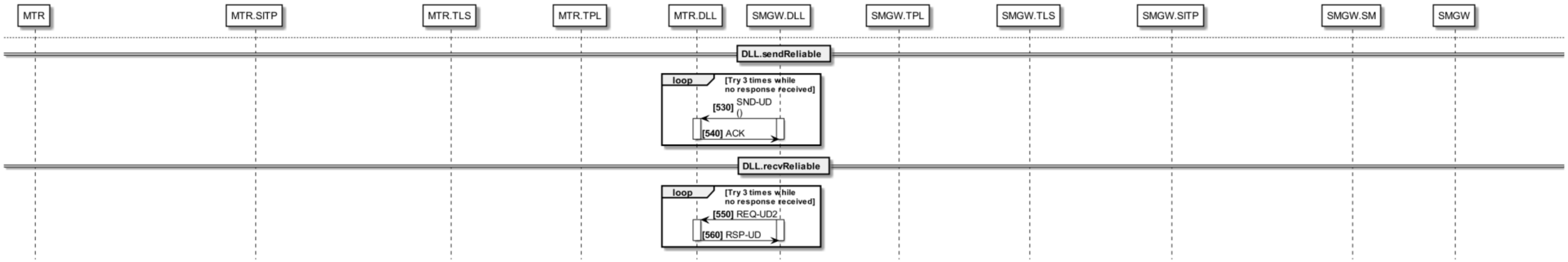


Figure F.C.4 – Key renewal procedure – Send and receive of datagrams



MK: Current MasterKey (Key Version n)
MK': New MasterKey (Key Version n+1)
SITP: Security Information Transport Protocol
AFL: Authentication and Fragmentation Layer
DLL: Data Link Layer
TPL: Transport Layer
TLS: Transport Layer Security
MTR: Meter
SMGW: Smart Meter Gateway
SM: Security Module in SMGW
KDF: Key Derivation Function
mtrid: Meter-Address
gwid: Gateway-Address

Appendix F.D (informative): Message examples of TLS

F.D.1 General

The colours are applied for the distinction of the different layers. They are used according to figure 1 of the superior document [OMS-S2].

The column "Offset" shows the length of the application data calculated from the beginning of the application layer.

NOTE 1: In this example all messages from the OED use a short header (no radio adapter on the OED).

NOTE 2: In example of radio frames are preamble, synchronisation word and CRC-fields not presented to get a better readability.

F.D.2 TLS ChannelRequest (from server to client)

F.D.2.1 Example for wireless M-Bus

Table F.D.1 – TLS-ChannelRequest

Layer	0 (first byte)	1	2	3
DLL	L = nn	C = 53h/73h (SND-UD)	MFCT_0	MFCT_1
DLL	ID_0(GW)	ID_1(GW)	ID_2(GW)	ID_3(GW)
DLL	VER(GW)	DEVTYPE(31h)		
ELL	CI = 8Ch(ELL)	CC	ACC	
AFL	CI = 90(AFL)	AFL.AFLL (0Fh)	AFL.FCL(01h) LSB	AFL.FCL(2Ch) MSB MF = 0,CTR, MCL, MAC
AFL	AFL.MCL(25h), CMAC8,CTR	AFL.MCR[7..0]	AFL.MCR[15..8]	AFL.MCR[23..16]
AFL	AFL.MCR[31..24]	AFL.MAC	AFL.MAC	AFL.MAC
AFL	AFL.MAC	AFL.MAC	AFL.MAC	AFL.MAC
AFL	AFL.MAC			
TPL	CI = 5Fh	ID_0(MTR)	ID_1(MTR)	ID_2(MTR)
TPL	ID_3(MTR)	MFCT_0(MTR)	MFCT_1(MTR)	VER(MTR)
TPL	DEVTYPE(MTR)	ACC	ST	CF[7..0]
TPL	CF[15..8](Mode13)	CFE[7..0] (TLSCHAN)		
TPL	ContentType (00h, ChannelRequest)	Reserved (00h)	Reserved (00h)	TLSSDULen MSB (00h)
TPL	TLSSDULen LSB (00h)			

The OED answers with ACK (no data) and waits for a REQ-UD2 from the gateway.

F.D.2.2 Example for wired M-Bus

Table F.D.2 – TLS ChannelRequest

Layer	0 (first byte)	1	2	3
DLL	68h	L = nn	L = nn	68h
DLL	C = 53h/73h (SND-UD)	PAddr (Primary M-Bus addr.)		
AFL	CI = 90h (AFL)	AFL.AFLL (0Fh)	AFL.FCL (00h) LSB FID = 0 (no fragments)	AFL.FCL (2Ch) MSB MF = 0 MCLP = 1 MLP = 0 MCRP = 1 MACP = 1
AFL	AFL.MCL (25h) MLMP = 0 MCMP = 1 AT = CMAC AES 128 length 8 bytes	AFL.MCR [7..0] (xxh)	AFL.MCR [15..8] (xxh)	AFL.MCR [23..16] (xxh)
AFL	AFL.MCR [31..32] (xxh)	AFL.MAC [63..56] (xxh)	AFL.MAC [55..48] (xxh)	AFL.MAC [47..40] (xxh)
AFL	AFL.MAC [39..32] (xxh)	AFL.MAC [31..24] (xxh)	AFL.MAC [23..16] (xxh)	AFL.MAC [15..8] (xxh)
AFL	AFL.MAC [7..0] (xxh)			
TPL	CI = 5Fh	ID_0(MTR)	ID_1(MTR)	ID_2(MTR)
TPL	ID_3(MTR)	MFCT_0(MTR)	MFCT_1(MTR)	VER(MTR)
TPL	DEVTYPE(MTR)	ACC	STS	CF[7..0]
TPL	CF[15..8](MODE 13)	CFE[7..0](TLSCHAN)		
TPL	ContentType (00h, Application Data)	Reserved (00h)	Reserved (00h)	TLSSDULen (MSB, 00h)
TPL	TLSSDULen (LSB, 00h)			
DLL	Chksum	0x16		

F.D.3 First message flight – ClientHello (from client to server)

Table F.D.3 – First message flight (TLS session initiate - Full handshake)

Layer	0 (first byte)	1	2	3
DLL	L = nn	C = 08h/28h (RSP-UD)	MFCT_0(MTR)	MFCT_1(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	
AFL	CI = 90h (AFL)	AFL.AFLL (0Fh)	AFL.FCL [7..0] (00h) FID = 0 (no fragments)	AFL.FCL [15..8] (2Ch) MF = 0 MCLP = 1 MLP = 0 MCRP = 1 MACP = 1
AFL	AFL.MCL (25h) MLMP = 0 MCMP = 1 AT = CMAC AES 128 length 8 bytes	AFL.MCR [7..0] (xxh)	AFL.MCR [15..8] (xxh)	AFL.MCR [23..16] (xxh)
AFL	AFL.MCR [31..32] (xxh)	AFL.MAC [63..56] (xxh)	AFL.MAC [55..48] (xxh)	AFL.MAC [47..40] (xxh)
AFL	AFL.MAC [39..32] (xxh)	AFL.MAC [31..24] (xxh)	AFL.MAC [23..16] (xxh)	AFL.MAC [15..8] (xxh)
AFL	AFL.MAC [7..0] (xxh)			
TPL	CI = 9Eh	ACC	STS	CF[7..0]
TPL	CF[15..8] (MODE 13,)	CFE[7..0] (TLSPROT)		
TPL	ContentType (16h, Handshake)	ProtoMajor (03h) TLS1.2	ProtoMinor(03h) TLS1.2	TLSSDULen (MSB, 00h)
TPL	TLSSDULen (LSB, 4Ch)	HSMsgType (01h ClientHello)	HSMsgLength (MSB, 00h)	HSMsgLength (00h)
TPL	HSMsgLength (LSB)	HSClientVersionMajor (03h)	HSClientVersionMinor (03h) TLS1.2	HSClientRandom 32
TPL	HSClientRandom 31	HSClientRandom	HSClientRandom	HSClientRandom
TPL	HSClientRandom 27	HSClientRandom	HSClientRandom	HSClientRandom
TPL	HSClientRandom 23	HSClientRandom	HSClientRandom	HSClientRandom
TPL	HSClientRandom 19	HSClientRandom	HSClientRandom	HSClientRandom
TPL	HSClientRandom 15	HSClientRandom	HSClientRandom	HSClientRandom
TPL	HSClientRandom 11	HSClientRandom	HSClientRandom	HSClientRandom
TPL	HSClientRandom 7	HSClientRandom	HSClientRandom	HSClientRandom
TPL	HSClientRandom 3	HSClientRandom 2	HSClientRandom 1	HSSessionIDLength (00h No Resume)[1]
TPL	CipherSuitesLength (MSB, 00h)	CipherSuitesLength (LSB, 08h)	CipherSuite Prio1 (MSB, C0h) TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256	CipherSuitePrio1 (LSB, 23h) TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256

[1] **NOTE**: If no TLS session resumption is requested, the SessionID length is 00h. With TLS session resumption the maximum length is 32 bytes.

Layer	0 (first byte)	1	2	3
TPL	CipherSuitePrio2 (MSB, C0h) TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SH384	CipherSuitePrio2 (LSB, 24h) TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384	CipherSuitePrio3 (MSB, C0h) TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	CipherSuitePrio3 (LSB, 2Bh) TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
TPL	CipherSuitePrio4 (MSB, C0h) TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384	CipherSuitePrio4 (LSB, 2Ch) TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384	CompressionMethods Length (01h)	NULL Compression (00h)
TPL	ExtensionsLength (MSB, 00h)	ExtensionsLength (LSB)	Extension elliptical_curves (MSB,00h)	Extension elliptical_curves (LSB, 0Ah)
TPL	Elliptical_curves_length (MSB,00h)	Elliptical_curves_length (LSB)	secp256r1 (NIST P-256) MSB (00h)	secp256r1(NIST P-256) LSB. (17h)
TPL	brainpoolp256r1 MSB (00h)	brainpoolp256r1LSB (1Ah)	brainpoolp384r1 MSB (00h)	brainpoolp384r1 LSB (1Bh)
TPL	Extension max.fragment_length (MSB 00h)	Extension max.fragment_length (LSB 01h)	Extension length (MSB 00h)	Extension Length (01h)
TPL	Extension Data (02h) Max.Fragment Size 1 kbyte	Extension encrypt-then-mac (MSB 00h)	Extension encrypt-then-mac (LSB 16h)	Extension length (MSB 00h)
TPL	Extension Length (LSB 00h)			

NOTE 1: This example contains 4 cipher suites to show possible applications but in practice at least one cipher suite is sufficient.

NOTE 2: The example does not contain the truncated-HMAC extension.

F.D.4 Second message flight – ServerHello, Certificate, ServerKeyExchange, CertificateRequest, ServerHelloDone (from server to client)

F.D.4.1 First TLS fragment

Table F.D.4 – Second message flight, First TLS fragment

Layer	0 (first byte)	1	2	3
DLL	L = nn	C = 53h/73h (SND-UD)	MFCT_0(GW)	MFCT_1(GW)
DLL	ID_0(GW)	ID_1(GW)	ID_2(GW)	ID_3(GW)
DLL	VER(GW)	DEVTYPE(31h)		
ELL	CI = 8Eh(ELL)	CC	ACC	MFCT_0 (OED)
ELL	MFCT_1(OED)	ID_0(OED)	ID_1(OED)	ID_2(OED)
ELL	ID_3(OED)	VER(OED)	DEVTYPE(OED)	
	CI = 90(AFL)	AFL.AFLL(05h)	AFL.FCL(01h) LSB	AFL.FCL(70h)MSB MF = 1, ML
AFL	AFL.MCL(40h) ML	AFL.ML (8Fh, LSB)	AFL.ML (01h, MSB)	
TPL	CI = 5Fh	ID_0(MTR)	ID_1(MTR)	ID_2(MTR)
TPL	ID_3(MTR)	MFCT_0(MTR)	MFCT_1(MTR)	VER(MTR)
TPL	DEVTYPE(MTR)	ACC	STS	CF[7..0]
TPL	CF[15..8](MODE 13)	CFE[7..0](TLSPROT)		
TPL	ContentType (16h, Handshake)	ProtoMajor (03h)	ProtoMinor (03h) TLS1.2	TLSSDULen (MSB)
TPL	TLSSDULen (LSB)	HSMsgType (02h ServerHello)	HSMsgLength (MSB, 00h)	HSMsgLength (00h)
TPL	HSMsgLength (LSB 26h)	HSServerVersionMajor (03h)	HSServerVersionMinor (03h) TLS1.2	HSServerRandom 32 (MSB)
TPL	HSServerRandom 31	HSServerRandom 30	HSServerRandom 29	HSServerRandom 28
TPL	HSServerRandom 27	HSServerRandom	HSServerRandom	HSServerRandom
TPL	HSServerRandom 23	HSServerRandom	HSServerRandom	HSServerRandom
TPL	HSServerRandom 19	HSServerRandom	HSServerRandom	HSServerRandom
TPL	HSServerRandom 15	HSServerRandom	HSServerRandom	HSServerRandom
TPL	HSServerRandom 11	HSServerRandom	HSServerRandom	HSServerRandom
TPL	HSServerRandom 7	HSServerRandom	HSServerRandom	HSServerRandom
TPL	HSServerRandom 3	HSServerRandom 2	HSServerRandom 1	HSSessionIDLength (00h)[2]
TPL	CipherSuite (MSB, C0h) TLS_ECDHE_ ECDSA_WITH_AES_ 128_CBC_SHA256	CipherSuite (LSB, 23h) TLS_ECDHE_ ECDSA_WITH_AES_ 128_CBC_SHA256	CompressionSelected (NONE, 00h)	ExtensionsLength (MSB 00h)
TPL	ExtensionsLength (LSB)	Extension elliptical_curves (MSB,00h)	Extension elliptical_curves (LSB, 0Ah)	Elliptical_curves_ length (MSB,00h)
TPL	Elliptical_curves_ length (LSB 02)	secp256r1 (NIST P-256) MSB (00h)	secp256r1(NIST P-256) LSB. (17h)	Extension max.fragment_length (MSB 00h)
TPL	Extension max.fragment_length (LSB 01h)	Extension length (MSB 00h)	Extension Length (01h)	Extension Data (02h) Max.Fragment Size 1 kbyte

2 **NOTE**: If TLS session resumption is used, this field contains the SessionID (up to 32 bytes)

Layer	0 (first byte)	1	2	3
TPL	Extension encrypt-then-mac (MSB 00h)	Extension encrypt-then-mac (LSB 16h)	Extension length (MSB 00h)	Extension Length (LSB 00h)
TPL	HSMsgType (0Bh Certificate)	HSMsgLength (MSB,00h)	HSMsgLength (00h)	HSMsgLength (LSB,F9h)
TPL	ASN.1 CertData (SEQUENCE 30h)	ASN.1 CertData (82h)	ASN.1 CertData	ASN.1 CertData
TPL	…	…	…	…
TPL	ASN.1 CertData	ASN.1 CertData	ASN.1 CertData	ASN.1 CertData

The OED answers with ACK.

F.D.4.2 Second TLS fragment

Table F.D.5 – Second message flight, second TLS fragment

Layer	0 (first byte)	1	2	3
DLL	L = nn	C = 53h/73h (SND-UD)	MFCT_0(GW)	MFCT_1(GW)
DLL	ID_0(GW)	ID_1(GW)	ID_2(GW)	ID_3(GW)
DLL	VER(GW)	DEVTYPE(31h)		
ELL	CI = 8Eh(ELL)	CC	ACC	MFCT_0 (OED)
ELL	MFCT_1(OED)	ID_0(OED)	ID_1(OED)	ID_2(OED)
ELL	ID_3(OED)	VER(OED)	DEVTYPE(OED)	
AFL	CI = 90h(AFL)	AFL.AFLL(02h)	AFL.FCL(02h) LSB	AFL.FCL(00h) MSB
TPL	ASN1.CertData	ASN1.CertData	ASN1.CertData	ASN1.CertData
TPL	…	…	…	…
TPL	ASN.1 CertData	ASN.1 CertData	ASN.1 CertData	ASN.1 CertData (CertDataEnd)
TPL	HSMsgType (0Ch) ServerKeyExchange	HSMsgLength (MSB, 00h)	HSMsgLength (00h)	HSMsgLength (LSB, 25h)
TPL	CurveType = namedCurve (03h)	00h (MSB)	17h (LSB, secp256r1)	Public ECC Key Point Len uncompress.(41h)
TPL	ECC Point format (04h)	ECC Point Public (2*32 bytes)	ECCPoint Public	ECCPoint Public
TPL	ECCPoint Public	ECCPoint Public	ECCPoint Public	ECCPoint Public
TPL	…	…	…	…
TPL	ECCPoint Public	HSMsgType (0Dh) CertificateRequest	HSMsgLength (MSB, 00h)	HSMsgLength (00h)
TPL	HSMsgLength (LSB 08h)	CertificateTypeList Len (01h)	40h ECDSA_Sign	00h (MSB Signature HashAlgoListLen)
TPL	02h (LSB Signature HashAlgoListLen)	04h (SHA256)	04h (SHA256) 03h (ECDSA)	CAListLen (MSB 00h)
TPL	CAListLen (LSB 00h)	HSMsgType (0Eh) ServerHelloDone	HSMsgLength (MSB, 00h)	HSMsgLength (00h)
TPL	HSMsgLength (LSB, 00h)			

The OED answers with ACK. The OED waits for REQ-UD2 from the gateway.

F.D.5 Third message flight, Certificate, ClientKeyExchange, CertificateVerify, ChangeCipherSpec, Finished (from client to server)

F.D.5.1 First TLS fragment

Table F.D. 6 – Third message flight, first TLS fragment

Layer	0 (first byte)	1	2	3
DLL	L = nnh	C = 08h/28h (RSP-UD)	MFCT_0(MTR)	MFCT_1(MTR)
DLL	ID_0(MTR)	ID_1(MTR)	ID_2(MTR)	ID_3(MTR)
DLL	VER(MTR)	DEVTYPE(MTR)		
ELL	CI = 8Eh(ELL)	CC	ACC	MFCT_0 (GW)
ELL	MFCT_1(GW)	ID_0(GW)	ID_1(GW)	ID_2(GW)
ELL	ID_3(GW)	VER(GW)	DEVTYPE(GW)	
AFL	CI = 90h(AFL)	AFL.AFLL(05h)	AFL.FCL(01h) LSB	AFL.FCL(70h) MSB MF = 1, MCL, ML
AFL	AFL.MCL(40h), ML	AFL.ML (B8h, LSB)	AFL.ML (01h, MSB)	
TPL	CI = 9Eh	ACC	STS	CF[7..0]
TPL	CF[15..8] (MODE 13)	CFE[7..0] (TLSPROT)		
TPL	ContentType (16h, Handshake)	ProtoMajor (03h)	ProtoMinor (03h) TLS1.2	TLSSDULen (MSB)
TPL	TLSSDULen (LSB)	HSMsgType (0Bh) Certificate	HSMsgLength (MSB)	HSMsgLength
TPL	HSMsgLength (LSB)	ASN.1 CertData (SEQUENCE 30h)	ASN.1 CertData (82h)	ASN.1 CertData
TPL	ASN1.CertData	ASN1.CertData	ASN1.CertData	ASN1.CertData
TPL	…	…	…	…
TPL	ASN.1 CertData	ASN.1 CertData	ASN.1 CertData	ASN.1 CertData

The OED waits for REQ-UD2 from the gateway.

F.D.5.2 Second TLS fragment

Table F.D.7 – Third message flight, second TLS fragment

Layer	0 (first byte)	1	2	3
DLL	L = nn	C = 08h/28h (RSP-UD)	MFCT_0(MTR)	MFCT_1(MTR)
DLL	ID_0(MTR)	ID_1(MTR)	ID_2(MTR)	ID_3(MTR)
DLL	VER(MTR)	DEVTYPE(MTR)		
ELL	CI = 8Eh(ELL)	CC	ACC	MFCT_0 (GW)
ELL	MFCT_1(GW)	ID_0(GW)	ID_1(GW)	ID_2(GW)
ELL	ID_3(GW)	VER(GW)	DEVTYPE(GW)	
AFL	CI = 90h(AFL)	AFL.AFLL(0Ah)	AFL.FCL(02h) LSB	AFL.FCL(00h) MSB MF = 0
TPL	ASN.1 CertData	ASN.1 CertData	ASN.1 CertData	ASN.1 CertData
TPL	...			
TPL	ASN.1 CertData	ASN.1 CertData(End)	HSMsgType (10h) ClientKey Exchange	HSMsgLength (MSB, 00h)
TPL	HSMsgLength (00h)	HSMsgLength (LSB 42h)	Public ECC Key Point Len uncompressed (41h)	ECC Point format (04h)
TPL	ECC Point Public (2*32 bytes)	ECCPoint Public	ECCPoint Public	ECCPoint Public
TPL	…	…	…	…
TPL	ECCPoint Public	ECCPoint Public	ECCPoint Public	ECCPoint Public
TPL	HSMsgType (0Fh) CertificateVerify	HSMsgLength MSB (00h)	HSMsgLength	HSMsgLength LSB (46h)
TPL	SignedHash	SignedHash	SignedHash	SignedHash
TPL	SignedHash	SignedHash	SignedHash	SignedHash
TPL	SignedHash	SignedHash	SignedHash	SignedHash
TPL	SignedHash	SignedHash	SignedHash	SignedHash
TPL	SignedHash	SignedHash	SignedHash	SignedHash
TPL	SignedHash	SignedHash	SignedHash	SignedHash
TPL	SignedHash	SignedHash	SignedHash	SignedHash
TPL	SignedHash	SignedHash	SignedHash	SignedHash
TPL	SignedHash	SignedHash	SignedHash	SignedHash
TPL	SignedHash	SignedHash	SignedHash	SignedHash
TPL	SignedHash	SignedHash	SignedHash	SignedHash
TPL	SignedHash	SignedHash	SignedHash	SignedHash
TPL	SignedHash	SignedHash	SignedHash	SignedHash
TPL	SignedHash	SignedHash	SignedHash	SignedHash
TPL	SignedHash	SignedHash	SignedHash	SignedHash
TPL	SignedHash	SignedHash	SignedHash	SignedHash
TPL	SignedHash	SignedHash	SignedHash	SignedHash
TPL	SignedHash	SignedHash	SignedHash	SignedHash
TPL	ContentType (14h, ChangeCipherSpec)	ProtoMajor (03h)	ProtoMinor (03h) TLS1.2	TLSSDULen (MSB, 00h)
TPL	TLSSDULen (LSB, 05h)	HSMsgType (01h) ChangeCipherSpec	HSMsgLength (MSB, 00h)	HSMsgLength
TPL	HSMsgLength (LSB, 01h)	Fix parameter (01h)	ContentType (16h, Handshake)	ProtoMajor (03h)

Layer	0 (first byte)	1	2	3
TPL	ProtoMinor (03h) TLS1.2	TLSSDULen (MSB, 00h)	TLSSDULen (LSB, 34h)	HSMsgType (14h) Finished
TPL	HSMsgLength (MSB, 00h)	HSMsgLength	HSMsgLength (LSB, 30h)	Encrypted HSHash
TPL	Encrypted HSHash	Encrypted HSHash	Encrypted HSHash	Encrypted HSHash
TPL	Encrypted HSHash	Encrypted HSHash	Encrypted HSHash	Encrypted HSHash
TPL	Encrypted HSHash	Encrypted HSHash	Encrypted HSHash	Encrypted Padding(03h)
TPL	Encrypted Padding(03h)	Encrypted Padding(03h)	Encrypted PadLength(03h)	HMAC
TPL	HMAC	HMAC	HMAC	HMAC
TPL	HMAC	HMAC	HMAC	HMAC
TPL	HMAC	HMAC	HMAC	HMAC
TPL	HMAC	HMAC	HMAC	HMAC
TPL	HMAC	HMAC	HMAC	HMAC
TPL	HMAC	HMAC	HMAC	HMAC
TPL	HMAC	HMAC	HMAC	HMAC
TPL	HMAC	HMAC	HMAC	

The OED waits for SND-UD from the gateway.

NOTE: Due to the length of the certificate it might be necessary to use a third fragment.

F.D.6 Fourth message flight – ChangeCipherSpec, finished (from server to client)

Using cipher suite AES128-CBC SHA256 with encrypt-then-MAC, without truncated-HMAC extensions.

Table F.D.8 – Fourth message flight

Layer	0 (first byte)	1	2	3
DLL	L = nn	C = 53h/73h (SND-UD)	MFCT_0(GW)	MFCT_1(GW)
DLL	ID_0(GW)	ID_1(GW)	ID_2(GW)	ID_3(GW)
DLL	VER(GW)	DEVTYPE(31h)		
ELL	CI = 8Eh(ELL)	CC	ACC	MFCT_0(OED)
ELL	MFCT_1(OED)	ID_0(OED)	ID_1(OED)	ID_2 (OED)
ELL	ID_3 (OED)	VER(OED)	DEVTYPE(OED)	
TPL	CI = 5Fh	ID_0(MTR)	ID_1(MTR)	ID_2(MTR)
TPL	ID_3(MTR)	MFCT_0(MTR)	MFCT_1(MTR)	VER(MTR)
TPL	DEVTYPE(MTR)	ACC	STS	CF[7..0]
TPL	CF[15..8](MODE 13,)	CF [23..16](TLSHS)		
TPL	ContentType (14h, ChangeCipherSpec)	ProtoMajor (03h)	ProtoMinor (03h) TLS1.2	TLSSDULen (MSB, 00h)
TPL	TLSSDULen (LSB, 05h)	HSMsgType (01h) ChangeCipherSpec	HSMsgLength (MSB, 00h)	HSMsgLength
TPL	HSMsgLength (LSB, 01h)	Fix parameter (01h)	ContentType (16h, Handshake)	ProtoMajor (03h)
TPL	ProtoMinor (03h) TLS1.2	TLSSDULen (MSB, 00h)	TLSSDULen (LSB, 34h)	HSMsgType (14h) Finished
TPL	HSMsgLength (MSB, 00h)	HSMsgLength	HSMsgLength (LSB, 30h)	Encrypted HSHash
TPL	Encrypted HSHash	Encrypted HSHash	Encrypted HSHash	Encrypted HSHash
TPL	Encrypted HSHash	Encrypted HSHash	Encrypted HSHash	Encrypted HSHash
TPL	Encrypted HSHash	Encrypted HSHash	Encrypted HSHash	Encrypted Padding(03h)
TPL	Encrypted Padding(03h)	Encrypted Padding(03h)	Encrypted PadLength(03h)	HMAC
TPL	HMAC	HMAC	HMAC	HMAC
TPL	HMAC	HMAC	HMAC	HMAC
TPL	HMAC	HMAC	HMAC	HMAC
TPL	HMAC	HMAC	HMAC	HMAC
TPL	HMAC	HMAC	HMAC	HMAC
TPL	HMAC	HMAC	HMAC	HMAC
TPL	HMAC	HMAC	HMAC	HMAC
TPL	HMAC	HMAC	HMAC	

The OED answers with ACK.

F.D.7 Xth message flight on wireless M-Bus – M-Bus application data transfer (from server to client)

This is an example of an M-Bus application command protected by AES128-CBC SHA256 transferred via the wireless M-Bus.

Table F.D.9 – Application data (wireless M-Bus, SND-UD)

Layer	0 (first byte)	1	2	3
DLL	L = nn	C = 53h/73h (SND-UD)	MFCT_0(GW)	MFCT_1(GW)
DLL	ID_0(GW)	ID_1(GW)	ID_2(GW)	ID_3(GW)
DLL	VER(GW)	DEVTYPE(31h)		
ELL	CI = 8Ch(ELL)	CC	ACC	
TPL	CI = 5Bh	ID_0(MTR)	ID_1(MTR)	ID_2(MTR)
TPL	ID_3(MTR)	MFCT_0(MTR)	MFCT_1(MTR)	VER(MTR)
TPL	DEVTYPE(MTR)	ACC	STS	CF[7..0]
TPL	CF[15..8](MODE 13)	CFE[7..0](TLSPROT)		
TPL	ContentType (17h, Application Data)	ProtoMajor (03h)	ProtoMinor (03h) TLS1.2	TLSSDULen (MSB, 00h)
TPL	TLSSDULen (LSB, 40h) incl. IV, HMAC and padding	IV (random)	IV (random)	IV (random)
TPL	IV (random)	IV (random)	IV (random)	IV (random)
TPL	IV (random)	IV (random)	IV (random)	IV (random)
TPL	IV (random)	IV (random)	IV (random)	IV (random)
TPL	IV (random)			
APL	Encrypted AppData DIF	Encrypted AppData VIF	Encrypted AppData Value	Encrypted AppData Value
APL	Encrypted AppData Value	Encrypted AppData Value	Encrypted AppData DIF2	Encrypted AppData VIF2
APL	Encrypted AppData VIFE	Encrypted AppData Value	Encrypted AppData Value	Encrypted AppData Value
APL	Encrypted AppData Value	Encrypted Padding(02h)	Encrypted Padding(02h)	Encrypted PadLength(02h)
TPL	SHA256 HMAC (first byte; shown without truncation)	HMAC	HMAC	HMAC
TPL	HMAC	HMAC	HMAC	HMAC
TPL	HMAC	HMAC	HMAC	HMAC
TPL	HMAC	HMAC	HMAC	HMAC
TPL	HMAC	HMAC	HMAC	HMAC
TPL	HMAC	HMAC	HMAC	HMAC
TPL	HMAC	HMAC	HMAC	HMAC
TPL	HMAC	HMAC	HMAC	HMAC 32 (last byte)

The OED answers with ACK. The gateway has to send a REQ-UD2 to receive application data from the OED.

The HMAC size for SHA256 is 32 bytes as the truncated-HMAC extension (see RFC6066) is not supported according to [BSI-TR03116-3].

The padding value and PadLen is calculated according to [RFC5246] 3.2.3.2.

Due to the 16 byte block length requirement for AES128 a minimum of 0-15 padding bytes is added before the PadLen byte. To hide the actual payload size, a multiple of 16 bytes padding can be additionally added (up to a total of 255 padding bytes).

Example: AppData Length = 13, HMAC = 32, PadLen = 1, IVLen = 16: (13+32+16+1) MOD 16 = 14. Thus the padding value and PadLen can have the values (16-14) = 0x02, 0x12, 0x22, 0x32 … 0xF2.

Because the link layer (12 byte), extended link layer (3 or 11 byte), the optional AFL header (7 byte) and the TPL header (0,5 or13 byte) adds 19 to 43 bytes, the APL data size is typically around 200 bytes. It is the task of the application layer to discover the maximum unfragmented TPL+APL size.

NOTE: 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.

F.D.8 X[th] message flight on wired M-Bus – M-Bus application data transfer (from client to server)

This is an example of an M-Bus application data protected by AES128-CBC SHA256 transferred via wired M-Bus.

The server sends a request of data.

Table F.D.10 – Application data (wired M-Bus, REQ-UD2)

Layer	0 (first byte)	1	2	3
DLL	10h	C = 5B/7Bh (REQ-UD)	PAddr (Primary M-Bus addr.)	Chksum
DLL	0x16			

The client responds the currently selected application data.

Table F.D.11 – Application data (wired M-Bus, RSP-UD)

Layer	0 (first byte)	1	2	3
DLL	68h	L = nn	L = nn	68h
DLL	C = 08h (RSP-UD)	PAddr (Primary M-Bus addr.)		
TPL	CI = 72h	ID_0(MTR)	ID_1(MTR)	ID_2(MTR)
TPL	ID_3(MTR)	MFCT_0(MTR)	MFCT_1(MTR)	VER(MTR)
TPL	DEVTYPE(MTR)	ACC	STS	CF[7..0]
TPL	CF[15..8](MODE 13)	CFE[7..0](TLSPROT)		
TPL	ContentType (17h, Application Data)	ProtoMajor (03h)	ProtoMinor (03h) TLS1.2	TLSSDULen (MSB, 00h)
TPL	TLSSDULen (LSB, 40h) incl. IV, HMAC and padding			
TPL	IV (random)	IV (random)	IV (random)	IV (random)
TPL	IV (random)	IV (random)	IV (random)	IV (random)
TPL	IV (random)	IV (random)	IV (random)	IV (random)
TPL	IV (random)	IV (random)	IV (random)	IV (random)
TPL	Encrypted AppData DIF	Encrypted AppData VIF	Encrypted AppData Value	Encrypted AppData Value
APL	Encrypted AppData Value	Encrypted AppData Value	Encrypted AppData DIF2	Encrypted AppData VIF2
APL	Encrypted AppData VIFE	Encrypted AppData Value	Encrypted AppData Value	Encrypted AppData Value
APL	Encrypted AppData Value	Encrypted Padding(02h)	Encrypted Padding(02h)	Encrypted PadLength(02h)
TPL	SHA256 HMAC (first byte; shown without truncation)	HMAC	HMAC	HMAC
TPL	HMAC	HMAC	HMAC	HMAC
TPL	HMAC	HMAC	HMAC	HMAC
TPL	HMAC	HMAC	HMAC	HMAC
TPL	HMAC	HMAC	HMAC	HMAC
TPL	HMAC	HMAC	HMAC	HMAC
TPL	HMAC	HMAC	HMAC	HMAC
TPL	HMAC	HMAC	HMAC	HMAC 32(LastByte)
DLL	Chksum	0x16		

Appendix F.E (informative): Examples of the Security Information Transfer Protocol

This appendix shows example datagrams between a gateway and an OED transporting the Security Information Transport Protocol (SITP).

F.E.1 Example Master key renewal with security mode 13 (bidirectional communication) – key transfer

The following table shows a SND-UD datagram sent by the gateway after successful reception of a SND-NR datagram by the OED. Thereby, it starts a bidirectional communication between both devices. Goal of the bidirectional communication is to transport and activate a new AES128 encryption key (actually the master key) to the OED.

Table F.E.1 – SITP transfer key command

Layer	0 (first byte)	1	2	3
DLL	L = nn	C = 43h (SND-UD2)	MFCT_0(GW)	MFCT_1(GW)
DLL	ID_0(GW)	ID_1(GW)	ID_2(GW)	ID_3(GW)
DLL	VER(GW)	DEVTYPE(31h		
ELL	CI = 8Ch(ELL)	CC	ACC	
TPL	CI = C3h	ID_0(MTR)	ID_1(MTR)	ID_2(MTR)
TPL	ID_3(MTR)	MFCT_0(MTR)	MFCT_1(MTR)	VER(MTR)
TPL	DEVTYPE(MTR)	ACC	STS	CF[7..0]
TPL	CF[15..8](MODE 13)	CFE[7..0](TLSPROT)		
TPL	ContentType(17h, Application Data)	ProtoMajor (03h)	ProtoMinor(03h) TLS1.2	TLSSDULen (MSB, 00h)
TPL	TLSSDULen (LSB, 60h) incl. IV, HMAC and padding			
TPL	IV (random)	IV (random)	IV (random)	IV (random)
TPL	IV (random)	IV (random)	IV (random)	IV (random)
TPL	IV (random)	IV (random)	IV (random)	IV (random)
TPL	IV (random)	IV (random)	IV (random)	IV (random)
APL-SITP	**Block length (BL)** LSB = 26h (38 bytes)	**Block length (BL)** MSB = 00h	**Block ID (BID)** = 00h	**Block Control Field (BCF)** = 00h (Transfer security information)
APL-SITP	**Recipient ID** = 00h (no dedicated application)	**Data Structure Identifier (DSI)** = 01h (Wrapping AES128 according NIST SP 800-38F type KWP)	**Data structure header (DSH1)** = FFh (WrapperKey ID = FFh identifies no wrapping in the TLS channel)	**Data structure header (DSH2)** = FFh (WrapperKeyVersion = FFh identifies no WrapperKey)
APL-SITP	**Wrapped struct. 01 (ICV)** = A6h	**Wrapped struct. 01 (ICV)** = 59h	**Wrapped struct. 01 (ICV)** = 59h	**Wrapped struct. 01 (ICV)** = A6h
APL-SITP	**Wrapped struct. 01 (MLI)** = 00h (MSB)	**Wrapped struct. 01 (MLI)** = 00h	**Wrapped struct. 01 (MLI)** = 00h	**Wrapped struct. 01 (MLI)** = 17h (LSB)
APL-SITP	**Key/random z1** (MSB)	**Key/random z1**	**Key/random z1**	**Key/random z1**
APL-SITP	**Key/random z1**	**Key/random z1**	**Key/random z1**	**Key/random z1**
APL-SITP	**Key/random z1**	**Key/random z1**	**Key/random z1**	**Key/random z1**

Layer	0 (first byte)	1	2	3
APL-SITP	**Key/random z1**	**Key/random z1**	**Key/random z1**	**Key/random z1** (LSB)
APL-SITP	**Target Time** = 00h (LSB)	**Target Time** = 00h	**Target Time** = 00h	**Target Time** = 80h
APL-SITP	**Target Time** = 30h (MSB)	**KeyID** = 00h (master key)	**KeyVersion** = 01h (First replaced key)	**Padding** = 00h
TPL	Encrypted Padding(07h)	Encrypted Padding(07h)	Encrypted Padding(07h)	Encrypted Padding(07h)
TPL	Encrypted Padding(07h)	Encrypted Padding(07h)	Encrypted Padding(07h)	Encrypted PadLength(07h)
TPL	SHA256 HMAC (first byte; shown without truncation)	HMAC	HMAC	HMAC
TPL	HMAC	HMAC	HMAC	HMAC
TPL	HMAC	HMAC	HMAC	HMAC
TPL	HMAC	HMAC	HMAC	HMAC
TPL	HMAC	HMAC	HMAC	HMAC
TPL	HMAC	HMAC	HMAC	HMAC
TPL	HMAC	HMAC	HMAC	HMAC
TPL	HMAC	HMAC	HMAC	HMAC 32 (last byte)

Assuming the OED receives the datagrams and processes them successfully, the corresponding response by the OED is as follows:

Table F.E.2 – SITP transfer key command response

Layer	0 (first byte)	1	2	3
DLL	L = nn	C = 08h (RSP-UD)	MFCT_0(MTR)	MFCT_1(MTR)
DLL	ID_0(MTR)	ID_1(MTR)	ID_2(MTR)	ID_3(MTR)
DLL	VER(MTR)	DEVTYPE(MTR)		
ELL	CI = 8Ch(ELL)	CC	ACC	
TPL	CI = C4h	ACC	STS	CF[7..0]
TPL	CF[15..8](MODE 13)	CFE[7..0](TLSPROT)		
TPL	ContentType(17h, Application Data)	ProtoMajor (03h)	ProtoMinor(03h) TLS1.2	TLSSDULen (MSB, 00h)
TPL	TLSSDULen (LSB, 40h) incl. IV, HMAC and padding			
TPL	IV (random)	IV (random)	IV (random)	IV (random)
TPL	IV (random)	IV (random)	IV (random)	IV (random)
TPL	IV (random)	IV (random)	IV (random)	IV (random)
TPL	IV (random)	IV (random)	IV (random)	IV (random)
APL-SITP	**Block length (BL)** LSB = 07h (7 bytes)	**Block length (BL)** MSB = 00h	**Block ID (BID)** = 00h	**Block Control Field (BCF)** = 80h (Status Response to Transfer security information)
APL-SITP	**Recipient ID** = 00h (no dedicated application)	**Data Structure Identifier (DSI)** = 22h (Status Response Structure)	**Data structure header (DSH1)** = FFh (KeyID = FFh identifies no wrapping in the TLS channel)	**Data structure header (DSH2)** = FFh (KeyVersion = FFh identifies no wrapper key)
APL-SITP	**Block parameters** = 00h (Successful command)	Encrypted Padding(06h)	Encrypted Padding(06h)	Encrypted Padding(06h)
TPL	Encrypted Padding(06h)	Encrypted Padding(06h)	Encrypted Padding(06h)	Encrypted PadLength(06h)
TPL	SHA256 HMAC (first byte; shown without truncation)	HMAC	HMAC	HMAC
TPL	HMAC	HMAC	HMAC	HMAC
TPL	HMAC	HMAC	HMAC	HMAC
TPL	HMAC	HMAC	HMAC	HMAC
TPL	HMAC	HMAC	HMAC	HMAC
TPL	HMAC	HMAC	HMAC	HMAC
TPL	HMAC	HMAC	HMAC	HMAC
TPL	HMAC	HMAC	HMAC	HMAC 32 (last byte)

F.E.2 Example Master key renewal with security mode 13 (bidirectional communication) – key activation/deactivation

At some point in time, the gateway activates the already transferred key by using the "combined activate/deactivate key" command. The corresponding datagram is as follows:

Table F.E.3 – SITP combined activate/deactivate key command

Layer	0 (first byte)	1	2	3
DLL	L = nn	C = 43h (SND-UD2)	MFCT_0(GW)	MFCT_1(GW)
DLL	ID_0(GW)	ID_1(GW)	ID_2(GW)	ID_3(GW)
DLL	VER(GW)	DEVTYPE(31h)		
ELL	CI = 8Ch(ELL)	CC	ACC	
TPL	CI = C3h	ID_0(MTR)	ID_1(MTR)	ID_2(MTR)
TPL	ID_3(MTR)	MFCT_0(MTR)	MFCT_1(MTR)	VER(MTR)
TPL	DEVTYPE(MTR)	ACC	STS	CF[7..0]
TPL	CF[15..8](MODE 13)	CFE[7..0](TLSPROT)		
TPL	ContentType(17h, Application Data)	ProtoMajor (03h)	ProtoMinor(03h) TLS1.2	TLSSDULen (MSB, 00h)
TPL	TLSSDULen (LSB, 60h) incl. IV, HMAC and padding			
TPL	IV (random)	IV (random)	IV (random)	IV (random)
TPL	IV (random)	IV (random)	IV (random)	IV (random)
TPL	IV (random)	IV (random)	IV (random)	IV (random)
TPL	IV (random)	IV (random)	IV (random)	IV (random)
APL-SITP	**Block length (BL)** LSB = 1Eh (30 bytes)	**Block length (BL)** MSB = 00h	**Block ID (BID)** = 00h	**Block Control Field (BCF)** = 04h (Combined activation/deactivation security information)
APL-SITP	**Recipient ID** = 00h (no dedicated application)	**Data Structure Identifier (DSI)** = 03h (Wrapping AES128 according NIST SP 800-38F type KWP)	**Data structure header (DSH1)** = FFh (WrapperKey ID = FFh identifies no wrapping in the TLS channel)	**Data structure header (DSH2)** = FFh (WrapperKeyVersion = FFh identifies no wrapper key)
APL-SITP	**Wrapped struct. 03 (ICV)** = A6h (MSB)	**Wrapped struct. 03 (ICV)** = 59h	**Wrapped struct. 03 (ICV)** = 59h	**Wrapped struct. 03 (ICV)** = A6h (LSB)
APL-SITP	**Wrapped struct. 03 (MLI)** = 00h (MSB)	**Wrapped struct. 03 (MLI)** = 00h	**Wrapped struct. 03 (MLI)** = 00h	**Wrapped struct. 03 (MLI)** = 0Ah (LSB)
APL-SITP	**Target Time** = 00h (LSB)	**Target Time** = 00h	**Target Time** = 00h	**Target Time** = 00h
APL-SITP	**Target Time** = 30h (MSB)	**Activated KeyID** = 00h (master key)	**Activated KeyVersion** = 01h (Activate previously transferred new master key version)	**Deactivated KeyID** = 00h (master key)
APL-SITP	**Deactivated KeyVersion** = 00h (Deactivate old master key version)	**Option** = 01h (no MessageCounter reset)	**Padding** = 00h	**Padding** = 00h
APL-SITP	**Padding** = 00h	**Padding** = 00h	**Padding** = 00h	**Padding** = 00h
TPL	Encrypted Padding(0Fh)	Encrypted Padding(0Fh)	Encrypted Padding(0Fh)	Encrypted Padding(0Fh)
TPL	Encrypted Padding(0Fh)	Encrypted Padding(0Fh)	Encrypted Padding(0Fh)	Encrypted Padding(0Fh)
TPL	Encrypted Padding(0Fh)	Encrypted Padding(0Fh)	Encrypted Padding(0Fh)	Encrypted Padding(0Fh)
TPL	Encrypted Padding(0Fh)	Encrypted Padding(0Fh)	Encrypted Padding(0Fh)	Encrypted PadLength(0Fh)

Layer	0 (first byte)	1	2	3
TPL	SHA256-HMAC (first byte; shown without truncation)	HMAC	HMAC	HMAC
TPL	HMAC	HMAC	HMAC	HMAC
TPL	HMAC	HMAC	HMAC	HMAC
TPL	HMAC	HMAC	HMAC	HMAC
TPL	HMAC	HMAC	HMAC	HMAC
TPL	HMAC	HMAC	HMAC	HMAC
TPL	HMAC	HMAC	HMAC	HMAC
TPL	HMAC	HMAC	HMAC	HMAC 32 (last byte)

The corresponding response of the OED to the "combined activate/deactivate key" command is as follows, assuming that the command is processed successfully.

Table F.E.4 – SITP combined activate/deactivate key command response

Layer	0 (first byte)	1	2	3
DLL	L = nn	C = 08h (RSP-UD)	MFCT_0(MTR)	MFCT_1(MTR)
DLL	ID_0(MTR)	ID_1(MTR)	ID_2(MTR)	ID_3(MTR)
DLL	VER(MTR)	DEVTYPE(MTR)		
ELL	CI = 8Ch(ELL)	CC	ACC	
TPL	CI = C4h	ACC	STS	CF[7..0]
TPL	CF[15..8](MODE 13)	CFE[7..0](TLSPROT)		
TPL	ContentType(17h, Application Data)	ProtoMajor (03h)	ProtoMinor(03h) TLS1.2	TLSSDULen (MSB, 00h)
TPL	TLSSDULen (LSB, 40h) incl. IV, HMAC and padding			
TPL	IV (random)	IV (random)	IV (random)	IV (random)
TPL	IV (random)	IV (random)	IV (random)	IV (random)
TPL	IV (random)	IV (random)	IV (random)	IV (random)
TPL	IV (random)	IV (random)	IV (random)	IV (random)
APL-SITP	**Block length (BL)** LSB = 07h	**Block length (BL)** MSB = 00h	**Block ID (BID)** = 00h	**Block Control Field (BCF)** = 84h (Status Response to Combined activation/deactivation security information)
APL-SITP	**Recipient ID** = 00h (no dedicated application)	**Data Structure Identifier (DSI)** = 22h (Status Response Structure)	**Data structure header (DSH1)** = FFh (KeyID = FFh identifies no wrapping in the TLS channel)	**Data structure header (DSH2)** = FFh (KeyVersion = FFh identifies no wrapper key)
APL-SITP	**Block parameters** = 00h (Successful command)	Encrypted Padding(06h)	Encrypted Padding(06h)	Encrypted Padding(06h)
TPL	Encrypted Padding(06h)	Encrypted Padding(06h)	Encrypted Padding(06h)	Encrypted PadLength(06h)
TPL	SHA256-HMAC (first byte; shown without truncation)	HMAC	HMAC	HMAC
TPL	HMAC	HMAC	HMAC	HMAC
TPL	HMAC	HMAC	HMAC	HMAC
TPL	HMAC	HMAC	HMAC	HMAC
TPL	HMAC	HMAC	HMAC	HMAC
TPL	HMAC	HMAC	HMAC	HMAC
TPL	HMAC	HMAC	HMAC	HMAC
TPL	HMAC	HMAC	HMAC	HMAC 32 (last byte)

Appendix F.F (informative): Example certificate

An example of a self-signed certificate for a gas meter is shown below.

NOTE 1: The size of the certificate below is 339 bytes.

NOTE 2: An X509v3 Key Usage "Digital Signature" is sufficient for use with ECDHE cipher suites, because TLS ephemeral encryption key pairs are generated independent from the certificate keys.

Certificate Data (TEXT with data fields):

```
Version: 3 (0x2)
Serial Number: c9:e7:c5:e1:44:ab:20:ba
Signature Algorithm: ecdsa-with-SHA256
    Issuer: CN = 7mfc0100001234.mtr
    Validity
        Not Before: Aug 1 00:00:00 2019 GMT
        Not After : Aug 1 00:00:00 2026 GMT
    Subject: CN = 7mfc0100001234.mtr
    Subject Public Key Info:
        Public Key Algorithm: id-ecPublicKey
            Public-Key: (256 bit)
            pub:
                04:8a:43:29:95:15:fa:fb:9f:45:6a:73:7d:b7:05:be:ee:15:
                d7:1e:cc:68:22:aa:c2:f4:ca:12:71:e4:58:8e:cc:76:35:5b:
                b4:90:06:35:d2:ee:56:29:49:12:72:06:23:75:f7:90:79:f6:
                48:74:07:92:dc:96:f0:ac:92:90:d4
            ASN1 OID: brainpoolP256r1
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:TRUE
        X509v3 Key Usage: Digital Signature
Signature Algorithm: ecdsa-with-SHA256
    30:46:02:21:00:82:83:d2:87:08:69:a1:c8:20:96:8a:96:66:3d:c9:15:9b:81:82:9f:
    96:d5:fc:41:e1:27:22:cc:71:4d:e1:97:02:21:00:88:99:82:c0:17:03:15:1c:a2:11:
    1c:9b:34:a6:0b:9d:d2:a6:0f:2b:32:79:b3:57:41:9e:5a:e9:1a:be:79:e2
```

Certificate Data (DER):

```
0000000: 30 82 01 4f 30 81 f5 a0 03 02 01 02 02 09 00 c9
0000010: e7 c5 e1 44 ab 20 ba 30 0a 06 08 2a 86 48 ce 3d
0000020: 04 03 02 30 1d 31 1b 30 19 06 03 55 04 03 0c 12
0000030: 37 6f 6d 73 31 61 30 30 30 30 30 34 30 30 2e 6d
0000040: 74 72 30 1e 17 0d 31 39 30 36 30 36 30 37 32 36
0000050: 32 32 5a 17 0d 32 36 30 36 30 34 30 37 32 36 32
0000060: 32 5a 30 1d 31 1b 30 19 06 03 55 04 03 0c 12 37
0000070: 6f 6d 73 31 61 30 30 30 30 30 34 30 30 2e 6d 74
0000080: 72 30 5a 30 14 06 07 2a 86 48 ce 3d 02 01 06 09
0000090: 2b 24 03 03 02 08 01 01 07 03 42 00 04 8a 43 29
00000a0: 95 15 fa fb 9f 45 6a 73 7d b7 05 be ee 15 d7 1e
00000b0: cc 68 22 aa c2 f4 ca 12 71 e4 58 8e cc 76 35 5b
00000c0: b4 90 06 35 d2 ee 56 29 49 12 72 06 23 75 f7 90
00000d0: 79 f6 48 74 07 92 dc 96 f0 ac 92 90 d4 a3 1d 30
00000e0: 1b 30 0c 06 03 55 1d 13 04 05 30 03 01 01 ff 30
00000f0: 0b 06 03 55 1d 0f 04 04 03 02 07 80 30 0a 06 08
0000100: 2a 86 48 ce 3d 04 03 02 03 49 00 30 46 02 21 00
0000110: 82 83 d2 87 08 69 a1 c8 20 96 8a 96 66 3d c9 15
0000120: 9b 81 82 9f 96 d5 fc 41 e1 27 22 cc 71 4d e1 97
0000130: 02 21 00 88 99 82 c0 17 03 15 1c a2 11 1c 9b 34
0000140: a6 0b 9d d2 a6 0f 2b 32 79 b3 57 41 9e 5a e9 1a
0000150: be 79 e2
```