ETSI TS 133 179 V13.2.0 (2016-10)

LTE;
Security of Mission Critical Push To Talk (MCPTT) over LTE
(3GPP TS 33.179 version 13.2.0 Release 13)

Reference
RTS/TSGS-0333179vd20

Keywords
LTE,SECURITY

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Intellectual Property Rights

IPRs essential or potentially essential to the present document may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (https://ipr.etsi.org/).

Pursuant to the ETSI IPR Policy, no investigation, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Foreword

This Technical Specification (TS) has been produced by ETSI 3rd Generation Partnership Project (3GPP).

The present document may refer to technical specifications or reports using their 3GPP identities, UMTS identities or GSM identities. These should be interpreted as being references to the corresponding ETSI deliverables.

The cross reference between GSM, UMTS, 3GPP and ETSI identities can be found under http://webapp.etsi.org/key/queryform.asp.

Modal verbs terminology

In the present document "**shall**", "**shall not**", "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the ETSI Drafting Rules (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

Contents

Intellectual Property Rights 2
Foreword 2
Modal verbs terminology 2
Foreword 7
1 Scope 8
2 References 8
3 Definitions and abbreviations 9
3.1 Definitions 9
3.2 Abbreviations 10
4 Overview of MCPTT security 10
4.1 General 10
4.2 Signalling plane security architecture 10
4.3 Application plane security architecture 11
4.3.1 General 11
4.3.2 User authentication and authorisation 11
4.3.3 Identity keying of users and services 12
4.3.4 Protection of application plane signalling 13
4.3.5 Media security 13
4.3.5.1 General 13
4.3.5.2 Media security for group communications. 13
4.3.5.3 Media security for private calls 15
5 Authentication and Authorization 16
5.1 General 16
5.2 LTE access authentication and security mechanism 17
5.3 Authentication for SIP core access 17
5.4 Authentication for HTTP-1 17
5.5 User Authentication 17
5.5.1 Identity Management Functional Model 17
5.5.2 User Authentication Framework 19
5.5.3 OpenID Connect (OIDC) 19
5.5.3.1 General 19
5.5.3.2 User Authentication example using Username/Password 21
5.6 MCPTT User Authorization 21
5.6.1 General 21
5.6.2 MCPTT user service authorization with MCPTT Server 23
5.6.2.0 General 23
5.6.2.1 Using SIP REGISTER 23
5.6.2.2 Using SIP PUBLISH 24
6 Signalling plane protection 25
6.1 SIP-1 interface security 25
6.2 HTTP-1 interface security 25
7 End-to-end communication security 25
7.1 Overview 25
7.2 Key provisioning and management 26
7.2.1 General 26
7.2.2 Functional model for key management 26
7.2.2.0 General 26
7.2.2.1 Reference point CSC-8 (between key management server and the key management client within the MCPTT UE) 27
7.2.2.2 Reference point CSC-9 (between the key management server and the key management client within the MCPTT Server) 27

7.2.2.3 Reference point CSC-10 (between the key management server and the key management client within a group management server) 27
7.2.3 Security procedures for key management 27
7.2.4 Provisioned key material to support end-to-end communication security 29
7.3 Group call key distribution 29
7.3.1 General 29
7.3.2 Security procedures for GMK provisioning 31
7.3.3 Key Identification and purpose tags 32
7.3.4 Group creation procedure 32
7.3.5 Dynamic group keying 32
7.3.5.1 General 32
7.3.5.2 Group regrouping procedures (within a single MCPTT system) 33
7.3.5.3 Group regrouping procedures (involving multiple MCPTT systems) 33
7.3.6 Derivation of SRTP/SRTCP master keys 34
7.4 Private call key distribution 35
7.4.1 General 35
7.4.2 Security procedures (on-network) 36
7.4.3 Security procedures (off-network) 37
7.4.4 Derivation of SRTP/SRTCP master keys 38
7.4.5 Void 39
7.5 Protection of media stream (SRTP) 39
7.5.1 General 39
7.5.2 Security procedures for media stream protection 40
7.6 Protection of offline floor control signalling (SRTCP) 41
7.6.1 General 41
7.6.2 Security procedures for offline floor control protection 42

8 Inter/Intra domain interface security 43
8.1 General 43

9 Protection of floor control and sensitive application signalling 44
9.1 Key agreement for protection of floor control and sensitive application data (Client to Server) 44
9.1.1 Identity-based key management for Client Server Key (CSK) 44
9.1.2 Creation of the CSK 45
9.1.3 Secure distribution of the CSK 45
9.1.3.0 General 45
9.1.3.1 MIKEY-SAKKE I_MESSAGE 45
9.1.3.2 Distribution of CSK during MCPTT Service Authorization and group subscription 46
9.1.3.3 Obtaining CSK from the I_MESSAGE 46
9.1.3.4 Procedure 46
9.2 Key agreement for protection of floor control and sensitive application data between servers 47
9.3 Protection of XML content 48
9.3.1 General 48
9.3.2 Protected content 48
9.3.3 Key agreement 49
9.3.4 Confidentiality protection using XML encryption (xmlenc) 49
9.3.5 Integrity protection using XML signature (xmlsig) 50
9.4 Key agreement for online floor control (SRTCP) 50
9.4.1 General 50
9.4.2 Key agreement between MCPTT client and MCPTT Server 51
9.4.3 Key agreement between MCPTT Servers 51
9.4.4 Key agreement for multicast from MCPTT Server 51
9.4.5 Derivation of SRTCP key material 51

Annex A (normative): Security requirements 53

A.0 Introduction 53

A.1 Configuration & service access 53

A.2 Group key management 53

A.3 On-network operation 53

A.4 Ambient listening 54
A.5 Data communication between MCPTT network entities 54
A.6 Key stream re-use 54
A.7 Late entry to group communication 55
A.8 Private call confidentiality 55
A.9 Off-network operation 55
A.10 Privacy of MCPTT identities 55
A.11 User authentication and authorization requirements 56

Annex B (normative): OpenID connect profile for MCPTT 57

B.0 General 57
B.1 MCPTT tokens 57
B.1.1 ID token 57
B.1.1.0 General 57
B.1.1.1 Standard claims 57
B.1.1.2 MCPTT claims 57
B.1.2 Access token 58
B.1.2.0 Introduction 58
B.1.2.1 Standard claims 58
B.1.2.2 MCPTT claims 58
B.2 MCPTT client registration 58
B.3 Obtaining tokens 59
B.3.0 General 59
B.3.1 Native MCPTT client 59
B.3.1.0 General 59
B.3.1.1 Authentication Request 59
B.3.1.2 Authentication response 60
B.3.1.3 Token request 61
B.3.1.4 Token Response 62
B.4 Refreshing an access token 62
B.4.0 General 62
B.4.1 Access token request 63
B.4.2 Access token response 63
B.5 Using the token to access MCPTT resource servers 64
B.6 Token validation 64
B.6.1 ID token validation 64
B.6.2 Access token validation 64

Annex C (informative): OpenID connect detailed flow 65

C.1 Detailed flow for MCPTT user authentication and registration using OpenID Connect 65

Annex D (Normative): KMS provisioning messages to support MCPTT 67

D.1 General aspects 67
D.2 KMS requests 67
D.3 KMS responses 68
D.3.0 General 68
D.3.1 KMS certificates 68
D.3.1.1 Description 68
D.3.1.2 Fields 69
D.3.1.3 User IDs 69
D.3.2 User Key Provision 69

D.3.2.1 Description....69
D.3.2.2 Fields....70
D.3.3 Example KMS response XML....70
D.3.3.1 Example KMSInit XML....70
D.3.3.2 Example KMSKeyProv XML....71
D.3.3.3 Example KMSCertCache XML....73
D.3.4 KMS Response XML schema....75
D.3.4.1 Base XML schema....75
D.3.4.2 Security Extension to KMS response XML schema....77

Annex E (normative): MIKEY message formats for media security....79

E.1 General aspects....79
E.1.0 Introduction....79
E.1.1 MIKEY common fields....79
E.2 MIKEY message structure for GMK distribution....79
E.3 MIKEY message structure for PCK distribution....80
E.4 MIKEY message structure for CSK distribution....81
E.5 MIKEY general extension payload to support 'SAKKE-to-self'....81
E.6 MIKEY general extension payload to encapsulate parameters associated with a GMK....82
E.6.1 General....82
E.6.2 IV....83
E.6.3 MCPTT group ID....83
E.6.4 Activation time....83
E.6.5 Text....83
E.6.6 Reserved....83
E.6.7 Random padding....83
E.6.8 Cryptography....84
E.6.9 Status....84
E.7 Hiding identities within MIKEY messages....84

Annex F (normative): Key derivation and hash functions....85

F.1 KDF interface and input parameter construction....85
F.1.1 General....85
F.1.2 FC value allocations....85
F.1.3 Calculation of the User Salt for GUK-ID generation....85
F.1.4 Calculation of keys for application data protection....85
F.2 Hash Functions....86
F.2.1 Generation of MIKEY-SAKKE UID....86

Annex G (informative): Change history....87

History....88

Foreword

This Technical Specification has been produced by the 3rd Generation Partnership Project (3GPP).

The contents of the present document are subject to continuing work within the TSG and may change following formal TSG approval. Should the TSG modify the contents of the present document, it will be re-released by the TSG with an identifying change of release date and an increase in version number as follows:

Version x.y.z

where:

- x the first digit:
  - 1 presented to TSG for information;
  - 2 presented to TSG for approval;
  - 3 or greater indicates TSG approved document under change control.
- y the second digit is incremented for all changes of substance, i.e. technical enhancements, corrections, updates, etc.
- z the third digit is incremented when editorial only changes have been incorporated in the document.

1 Scope

The present document specifies the security architecture, procedures and information flows needed to protect the mission critical push to talk (MCPTT) service. The architecture includes mechanisms for authentication, protection of MCPTT signalling and protection of MCPTT media. Security for both MCPTT group calls and MCPTT private calls operating in on-network and off-network modes of operation is specified.

The functional architecture for MCPTT is defined in 3GPP TS 23.179 [2], the corresponding service requirements are defined in 3GPP TS 22.179 [3].

The MCPTT service can be used for public safety applications and also for general commercial applications e.g. utility companies and railways. As the security model is based on the public safety environment, some security features may not be applicable to MCPTT for commercial purposes.

2 References

The following documents contain provisions which, through reference in this text, constitute provisions of the present document.

- References are either specific (identified by date of publication, edition number, version number, etc.) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies. In the case of a reference to a 3GPP document (including a GSM document), a non-specific reference implicitly refers to the latest version of that document *in the same Release as the present document.*

[1] 3GPP TR 21.905: "Vocabulary for 3GPP Specifications".

[2] 3GPP TS 23.179: "Functional architecture and information flows to support mission critical communication services; Stage 2".

[3] 3GPP TS 22.179: "Mission Critical Push To Talk (MCPTT) over LTE; Stage 1".

[4] 3GPP TS 33.210: "3G security; Network Domain Security (NDS); IP network layer security".

[5] 3GPP TS 33.310: "Network Domain Security (NDS); Authentication Framework (AF)".

[6] 3GPP TS 33.203: "3G security; Access security for IP-based services".

[7] Void.

[8] 3GPP TS 33.328: "IP Multimedia Subsystem (IMS) media plane security".

[9] IETF RFC 6507: "Elliptic Curve-Based Certificateless Signatures for Identity-Based Encryption (ECCSI)".

[10] IETF RFC 6508: "Sakai-Kasahara Key Encryption (SAKKE)".

[11] IETF RFC 6509: "MIKEY-SAKKE: Sakai-Kasahara Key Encryption in Multimedia Internet KEYing (MIKEY)".

[12] IETF RFC 3550: "RTP: A Transport Protocol for Real-Time Applications".

[13] IETF RFC 3711: "The Secure Real-time Transport Protocol (SRTP)".

[14] 3GPP TS 33.401: "3GPP System Architecture Evolution (SAE); Security architecture".

[15] 3GPP TS 23.228: "IP Multimedia Subsystem (IMS); Stage 2".

[16] 3GPP TS 33.222: "Generic Authentication Architecture (GAA); Access to network application functions using Hypertext Transfer Protocol over Transport Layer Security (HTTPS)".

[17] 3GPP TS 33.220: "Generic Authentication Architecture (GAA); Generic Bootstrapping Architecture (GBA)".

[18] NIST FIPS 180-4: "Secure Hash Standard (SHS)".

[19] IETF RFC 6749: "The OAuth 2.0 Authorization Framework".

[20] IETF RFC 6750: "The OAuth 2.0 Authorization Framework: Bearer Token Usage".

[21] OpenID Connect 1.0: "OpenID Connect Core 1.0 incorporating errata set 1", http://openid.net/specs/openid-connect-core-1_0.html.

[22] IETF RFC 3830: "MIKEY: Multimedia Internet KEYing".

[23] IETF RFC 3602: "The AES-CBC Cipher Algorithm and Its Use with IPsec".

[24] IETF RFC 4771: "Integrity Transform Carrying Roll-Over Counter for the Secure Real-time Transport Protocol (SRTP)".

[25] IETF RFC 6043: "MIKEY-TICKET: Ticket-Based Modes of Key Distribution in Multimedia Internet KEYing (MIKEY)".

[26] IETF RFC 7714: "AES-GCM Authenticated Encryption in the Secure Real-time Transport Protocol (SRTP)".

[27] W3C: "XML Encryption Syntax and Processing Version 1.1", https://www.w3.org/TR/xmlenc-core1/.

[28] W3C: "XML Signature Syntax and Processing (Second Edition)", http://www.w3.org/TR/xmldsig-core/.

[29] IETF RFC 5905: "Network Time Protocol Version 4: Protocol and Algorithms Specification".

[30] IETF RFC 5480: "Elliptic Curve Cryptography Subject Public Key Information".

[31] IETF RFC 6090: "Fundamental Elliptic Curve Cryptography Algorithms".

[32] IETF RFC 7519: "JSON Web Token (JWT)".

[33] IETF RFC 7662: "OAuth 2.0 Token Introspection".

[34] IETF RFC 3394: "Advanced Encryption Standard (AES) Key Wrap Algorithm".

[35] IETF RFC 7515: "JSON Web Signature (JWS)".

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, the terms and definitions given in 3GPP TR 21.905 [1] and the following apply. A term defined in the present document takes precedence over the definition of the same term, if any, in 3GPP TR 21.905 [1].

Floor: Floor(x) is the largest integer smaller than or equal to x.

3.2 Abbreviations

For the purposes of the present document, the abbreviations given in 3GPP TR 21.905 [1] and the following apply. An abbreviation defined in the present document takes precedence over the definition of the same abbreviation, if any, in 3GPP TR 21.905 [1].

CSC	Common Services Core
GMK	Group Master Key
GMK-ID	Group Master Key Identifier
GMS	Group Management Server
GUK-ID	Group User Key Identifier
IdM	Identity Management
IdMS	Identity Management Server
KMS	Key Management Server
MBCP	Media Burst Control Protocol
MCPTT	Mission Critical Push to Talk
MKI	Master Key Identifier
OIDC	OpenID Connect
PCK	Private Call Key
PCK-ID	Private Call Key Identifier
PSK	Pre-Shared Key
SRTCP	Secure Real-Time Transport Control Protocol
SRTP	Secure Real-Time Transport Protocol
SSRC	Synchronization Source
TBCP	Talk Burst Control Protocol
TrK	KMS Transport Key
UID	User Identifier for MIKEY-SAKKE (referred to as the 'Identifier' in RFC 6509 [11])

4 Overview of MCPTT security

4.1 General

The MCPTT security architecture defined in this document is designed to meet the security requirements defined in Annex A. The MCPTT security architecture provides signalling and application plane security mechanisms to protect metadata and communications used as part of the MCPTT service. The following signalling plane security mechanisms are used by the MCPTT service:

- Protection of the signalling plane used by the MCPTT Service, defined in clause 6.
- Protection of inter/intra domain interfaces, defined in clause 8.

The following application plane security mechanisms are used by the MCPTT service:

- Authentication and authorisation of users to the MCPTT Service, defined in clause 5.
- Protection of sensitive application signalling within the MCPTT Service, defined in clause 9.
- Security of floor control within the MCPTT Service, defined in clause 7.
- End-to-end security of user media within the MCPTT Service, also defined in clause 7.

Security mechanisms in the signalling and application plane are independent of each other, but are both required for a secure MCPTT system.

4.2 Signalling plane security architecture

Within MCPTT, signalling plane security protects the interfaces used by the MCPTT application. Figure 4.2-1 provides an overview of these interfaces.

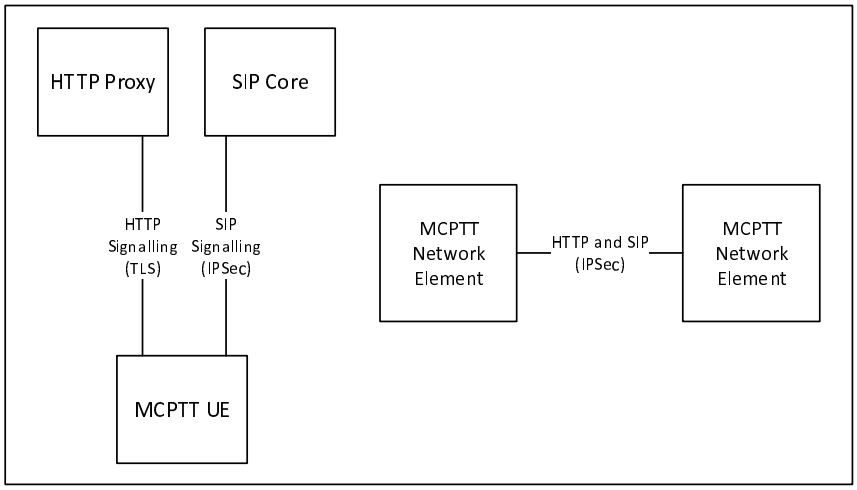


Figure 4.2-1: Signalling plane security architecture

MCPTT signalling from the UE is passed over both HTTP and SIP. The signalling plane security mechanisms for UE to Server interfaces are defined in clause 6. Additionally, MCPTT data is passed between MCPTT network elements, either inter or intra MCPTT domain. The security mechanism for protecting data between MCPTT network elements is defined in clause 8.

4.3 Application plane security architecture

4.3.1 General

Application plane security provides protection both between MCPTT clients, between the MCPTT client and the MCPTT domain, and also between MCPTT domains. Application plane security on the client is bound to the MCPTT user associated with the client and not to the MCPTT UE. Consequently, user authentication and authorisation to the MCPTT domain is required prior to access to the majority of MCPTT services.

Application plane signalling security allows protection of MCPTT-specific signalling from non-MCPTT entities (including the SIP core). Application plane signalling security is applied from the MCPTT client to the client's primary MCPTT domain. It may also be applied between MCPTT domains.

Media security allow protection of MCPTT media within the MCPTT system. It is applied end-to-end between MCPTT clients. It is a configuration option whether MCPTT network entities, including the MCPTT Server is able to access the content of MCPTT media.

Additionally, signalling plane protection is applied to all HTTP and SIP connections into the MCPTT domain. While signalling plane protection and signalling plane entities are not shown in this subclause, including the SIP core and HTTP proxy, it is assumed that signalling plane protection mechanisms are in use.

4.3.2 User authentication and authorisation

Prior to connecting to the MCPTT domain, the MCPTT user application requires a 'token' authorising its access to MCPTT services. To obtain authorisation token(s), the MCPTT user application authenticates the MCPTT user to an Identity Management Server which provides the authorisation token.

The authorisation token is provided to MCPTT network entities, such as the MCPTT Server, over an MCPTT signalling interface (either a HTTP interface or SIP interface). The MCPTT network entity will provide access to MCPTT services based upon the token provided.

The architecture for user authentication and authorisation is shown in Figure 4.3.2-1.

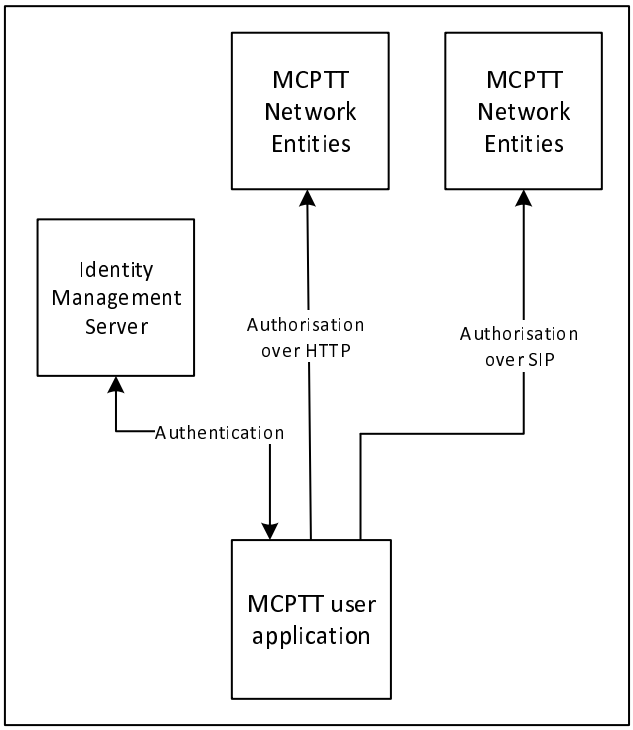


Figure 4.3.2-1: User authentication and authorisation

While not shown in Figure 4.3.2-1, authorisation occurs over HTTP or SIP and hence uses signalling plane protection to encrypt HTTP to a HTTP proxy and to encrypt SIP to a SIP core.

The mechanism to perform user authentication and authorisation is defined in clause 5.

4.3.3 Identity keying of users and services

Once a MCPTT client has obtained user authorisation to access the MCPTT domain, the client may obtain key material associated with the user's identity using the authorisation token. Identity keys are required to support key distribution for application signalling, floor control and media. Identity key material is obtained via an HTTP request to a Key Management Server as shown in Figure 4.3.3-1.

Identitiy keying is repeated periodically (e.g. monthly). This ensures that user identities are regularly verified and that users that are no longer part of the MCPTT domain are removed from the system.

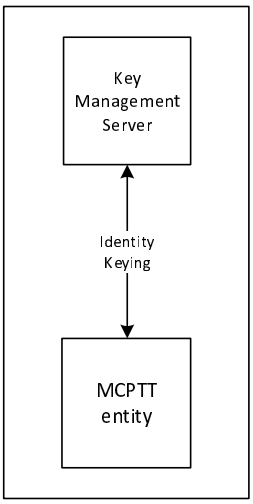


Figure 4.3.3-1: Identity keying of MCPTT entities

While not shown in Figure 4.3.3-1, the connection to the KMS is over HTTP and hence is secured up to the HTTP proxy. Additionally, key material may be wrapped using a transport key distributed out-of-band.

A number of MCPTT network entities also require identity key material including the MCPTT server and Group Management Server. This key material is obtained via the same HTTP interface.

The mechanism to perform identity keying is defined in clause 7.2.

4.3.4 Protection of application plane signalling

Application plane signalling security protects application signalling between the MCPTT user application and the MCPTT server. Key distribution for application signalling is performed by sending a key material from the MCPTT client to the MCPTT Server over the SIP interface. The key is secured using the identity key material provisioned by the Key Management Server. The security architecture is shown in Figure 4.3.4-1.

The mechanism to provide application plane signalling is defined in clause 9.

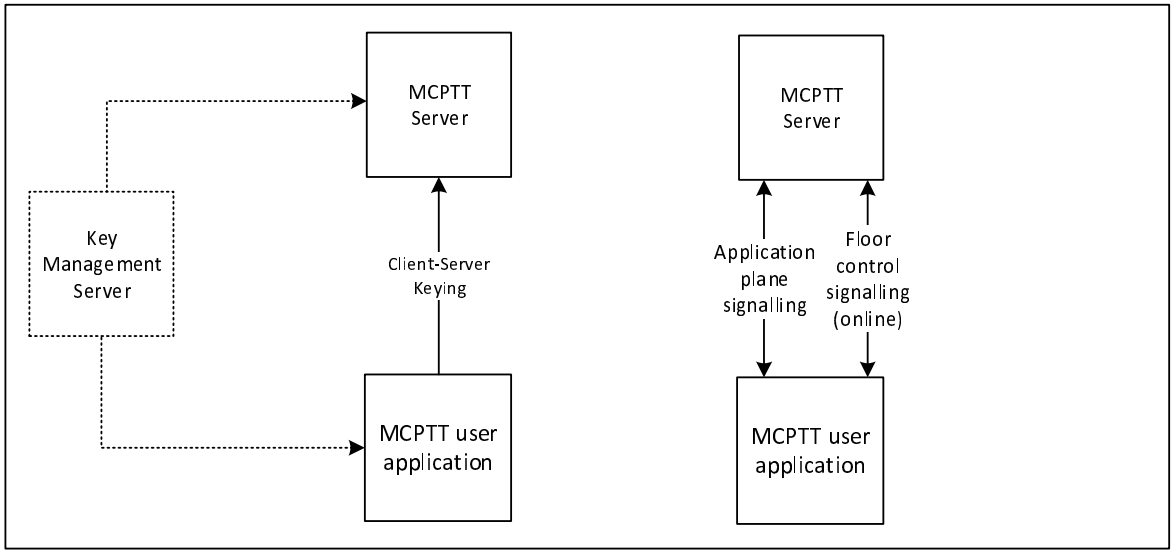


Figure 4.3.4-1: Application plane signalling security

Application plane signalling security can also be applied between MCPTT servers. In this case the MCPTT servers are keyed manually. While not shown in Figure 4.3.4-1, application plane signalling uses SIP and HTTP and hence is also secured up to the SIP core and HTTP proxy respectively.

4.3.5 Media security

4.3.5.1 General

Media security establishes an end-to-end security context between MCPTT users to support group communications and private calls. The intention is for media to be able to be encrypted end-to-end between MCPTT clients, irrespective of whether the media is routed unicast via the media distribution server, multicast via the media distribution server, or transmitted over a direct connection.

Key distribution for groups is performed by the Group Management Server. Key distribution for private calls is performed by the initiating MCPTT client. Once a security context is established, the method for encrypting media for groups and private calls is the same. Aditionally, when MCPTT UEs are offline, the security context that is used to protect media security is also used to protect floor control signalling.

4.3.5.2 Media security for group communications.

Media security for groups is secured by establishing a shared group security context between group members. Key distribution for the group security context is performed by a Group Management Server. The Group Management Server sends a group keys and group security parameters over SIP as part of group management.

Group keys and security parameters are encrypted by the Group Management Server to individual MCPTT users that are members of the group. The Group Management Server may choose to distribute the group key to MCPTT Server(s) to allow the media mixing function within the MCPTT Server(s) to be used. MCPTT users and MCPTT services require identity keying by a KMS prior to performing group management.

Figure 4.3.5.2-1 provides an overview of the group keying process. Details of the process may be found in clause 7.3.

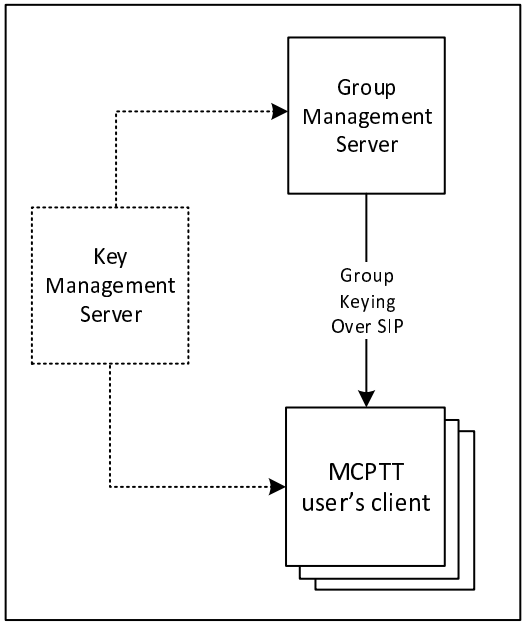


Figure 4.3.5.2-1: Group keying for media security

Once a group key has been shared with MCPTT users, keys are derived from that group key to protect media (and floor control when the UE is offline). Key derivation is based on the MCPTT users' identity, hence every member of the group encrypts media using a different key.

Media is encrypted using the SRTP protocol. When the MCPTT UE has a network connection the encrypted media is routed to other MCPTT clients via the media distribution function in the MCPTT Server. Media may be distributed over unicast (MCPTT-7) or multicast (MCPTT-8). When the MCPTT UE is offline, the encrypted media is routed directly to MCPTT clients on other MCPTT UEs. The security procedure for protecting media is the same in either case. Details of media encryption is provided in clause 7.5.

Floor control is encrypted using the SRTCP protocol. Unlike media, floor control is protected differently when the UE has a network connection and when it is offline. When the UE has a network connection, floor control traffic is encrypted to the floor control server within the MCPTT Server. When it is offline, floor control is encrypted directly to group UEs using a key derived from the group security context. Details of floor control encryption is provided in clause 7.6.

Figure 4.3.5.2-2 provides an overview of how media is protected for group communications.

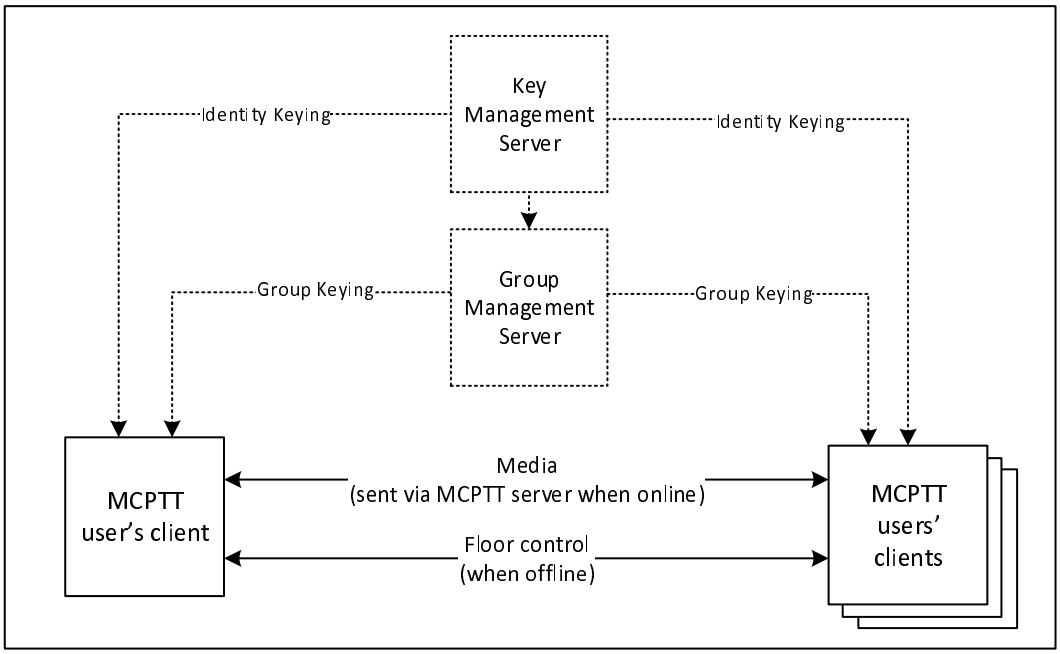


Figure 4.3.5.2-2: Group media protection

4.3.5.3 Media security for private calls

As part setting up a private call, the call initiator provides a key to the terminating client. The key is encrypted to the MCPTT user that is currently registered on the terminating client. As a result, MCPTT users require identity keying by a KMS prior to performing group management.

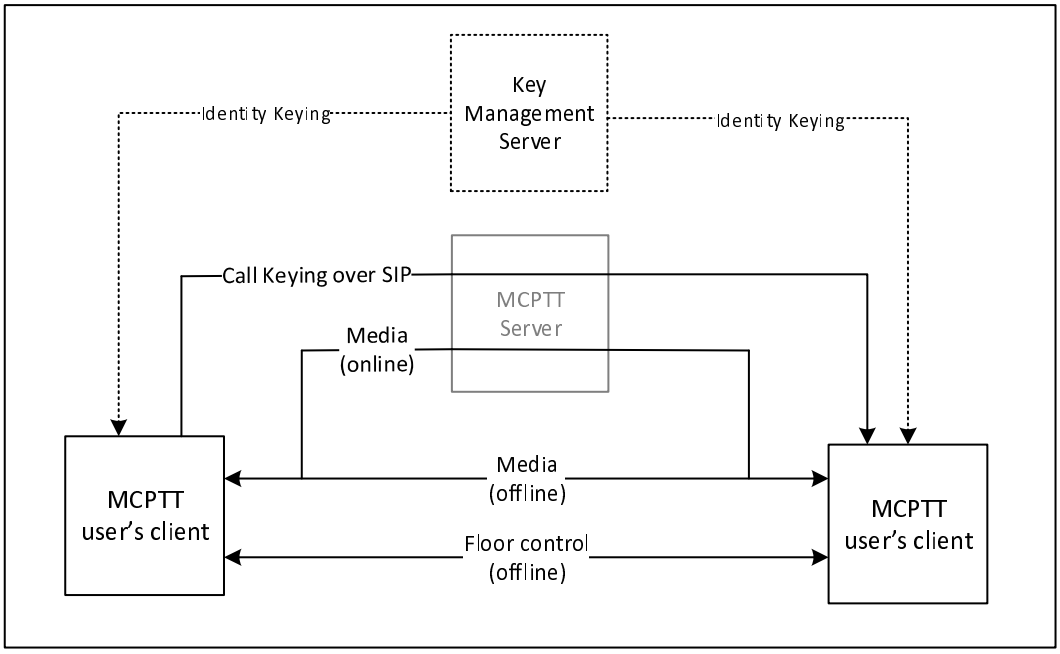


Figure 4.3.5.3-1: Media security for private calls

Figure 4.3.5.3-1 provides an overview of media protection for private calls. For clarity, MCPTT network entities will not have the private call key material and hence will not be able to decrypt the media for the private call communication (unless the monitoring function is specifically authorised for either user).

Details of private call key distribution are provided in clause 7.4.

Once private call key distribution has been completed, media and floor control is protected as described for group communications in clause 4.3.5.2. Media will be routed via the media distribution function in the MCPTT Server when

the UE is online, and directly when the UE is offline. The media security context shall also be used to protect floor control when the MCPTT UE is offline. Details of media and floor control protection may be found in clauses 7.5 and 7.6 respectively.

5 Authentication and Authorization

5.1 General

The generic steps for MCPTT authentication is shown in figure 5.1-1.

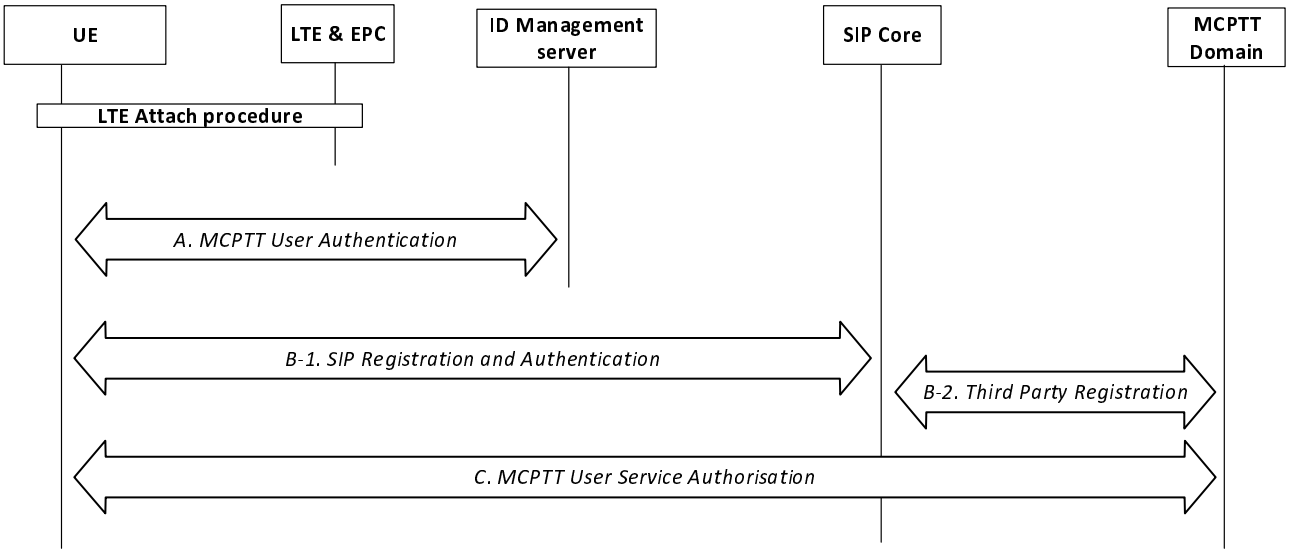


Figure 5.1-1: MCPTT Authentication

At UE power-on, the MCPTT UE performs LTE authentication as specified in TS 33.401 [14]. The MCPTT UE then performs the following authentication procedures to successfully complete the MCPTT service registration and identity binding between signalling layer identities and the MCPTT user identities.

- A: MCPTT user authentication.
- B: SIP Registration and Authentication.
- C: MCPTT Service Authorization.

These procedures are described in more detail in subsequent clauses.

Steps A and B may be performed in either order or in parallel. For scenarios where this order has an impact on the identity bindings between signalling layer identities and the MCPTT user identities, a re-registration (Step B) to the SIP Core may be performed to update the registered signalling layer identity.

If an MCPTT UE completes SIP registration in Step B prior to performing MCPTT user authentication in Step A and MCPTT user service authorization in Step C, the MCPTT UE shall be able to enter a 'limited service' state. In this limited state, where the MCPTT user is not authorized with the MCPTT service, the MCPTT UE shall be able to use limited MCPTT services (e.g. an anonymous MCPTT emergency call). The MCPTT Server is informed of the registration of the MCPTT UE with the SIP core though Step B-2.

Additionally, an HTTP-1 authentication mechanism is used.

NOTE: Mechanisms for confidentiality and integrity protection (not defined in this clause) may be combined only with certain authentication procedures.

5.2 LTE access authentication and security mechanism

MCPTT UE shall perform the authentication and security mechanisms as specified in 3GPP TS 33.401 [14] for LTE network access security.

5.3 Authentication for SIP core access

This clause specifies the mutual authentication between the UE and the SIP core.

IMS AKA authentication shall be performed as specified in 3GPP TS 33.203 [6] for SIP core access. IMS AKA authentication mechanism as specified in 3GPP TS 33.203 [6] shall be performed irrespective of whether SIP core architecture is compliant with 3GPP TS 23.228 [15] or not.

Authentication related information shall be provided by SIP database that may be part of the HSS or may be part of the MCPTT service provider's SIP database depending on the SIP core deployment scenarios specified in 3GPP TS 23.179 [2].

Implementation options and requirements on the ISIM or USIM application to support SIP core access security are specified in 3GPP TS 33.203 [6].

5.4 Authentication for HTTP-1

Based on the configuration provided with the MCPTT client in the UE, one of the following authentication mechanisms shall be performed between the HTTP Client in the MCPTT UE and the HTTP Proxy:

- one-way authentication of the HTTP Proxy based on the server certificate;
- mutual authentication based on certificates;
- mutual authentication based on pre-shared key.

If authentication based on certificate is performed, then the profiles as given in 3GPP TS 33.310 [5], clauses 6.1.3a and 6.1.4a shall be used. The structure of the PKI used for the certificate is out of scope of the present document. Guidance on certificate based mutual authentication is provided in 3GPP TS 33.222 [16], annex B.

The usage of Pre-Shared Key Ciphersuites for Transport Layer Security (TLS) is specified in the TLS profile given in 3GPP TS 33.310 [5], annex E.

Editor's Note: 3GPP TS 23.179 specifies that HTTP-1 reference point shall use the Ut reference point as defined in 3GPP TS 23.002 [5]. Security for Ut reference point allows also other authentication options as specified above, which is ffs whether they would be suitable for MCPTT.

5.5 User Authentication

5.5.1 Identity Management Functional Model

The Identity Management functional model for MCPTT is shown in figure 5.5.1-1 and consists of the identity management server located in the MCPTT common services core and the identity management client located in the MCPTT UE. The IdM server and the IdM client in the MCPTT UE establish the foundation for MCPTT user authentication and user authorization.

The CSC-1 reference point, between the MCPTT IdM client in the UE and the Identity Management server, provides the interface for user authentication. CSC-1 is a direct HTTP interface between the IdM client in the UE and the IdM server and shall support OpenID Connect 1.0 ([19], [20] and [21]).

The OpenID Connect profile for MCPTT shall be implemented as defined in annex B. MCPTT user authentication, MCPTT user authorization, OpenID Connect 1.0, and the OpenID Connect profile for MCPTT shall form the basis of the MCPTT identity management architecture.

In alignment with the OpenID Connect 1.0 [21] and OAuth 2.0 standards [19] and [20], CSC-1 shall consist of two identity management interfaces; the authorization endpoint and the token endpoint. These endpoints are separate and independent from each other, requiring separate and independent IP addressing. The authorization endpoint server and the token endpoint server may be collectively referred to as the IdM server in this document.

The HTTP connection between the Identity Management client and the Identity management server shall be protected using HTTPS.

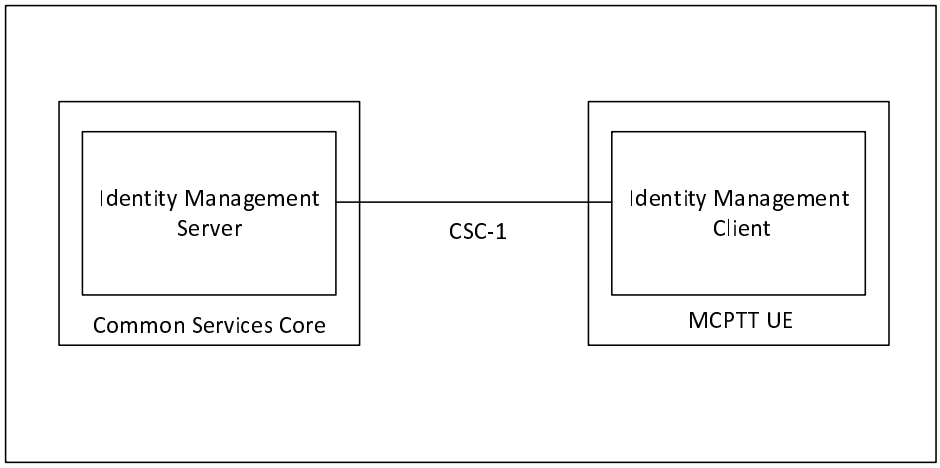


Figure 5.5.1-1: Functional Model for MCPTT Identity Management

To support MCPTT user authentication, the IdM server (IdMS) shall be provisioned with the user's MC ID and MCPTT ID (the MCPTT ID may be the same as the MC ID). A mapping between the MC ID and MCPTT ID shall be created and maintained in the IdMS. When an MCPTT user wishes to authenticate with the MCPTT service, the user is directed to provide their MC ID and credentials via the UE IdM client to the IdMS (note that the primary authentication method used to obtain the MC ID and credentials is out of scope of the present document). The IdMS receives and verifies the user's MC ID and credentials, and if valid returns an id token, refresh token, and access token to the UE IdM client specific to that user. The IdM client learns the user's MCPTT ID from the id token. Table 5.5.1-1 shows the MCPTT tokens and their usage.

Table 5.5.1-1: MCPTT tokens

Token Type	Consumer of the Token	Description (See Annex B for details)
Id token	UE client(s)	Contains MCPTT ID and whatever info related to the user that is useful to the client.
Access token	KMS, MCPTT server, etc. (Resource Server)	Short-lived token (definable in the IdMS) that conveys the user's identity. This token contains the MCPTT ID.
Refresh token	IdM server (Authorization Server)	Allows UE to obtain a new access token without forcing user to log in again.

In support of MCPTT user authorization, the access token(s) obtained during user authentication is used to gain MCPTT services for the user.

The KM client in the UE uses an access token to obtain key management service authorization from the KMS. The KMS verifies the access token, and if valid, authorizes the user for key management services and returns identity specific media and signalling key information to the KM client in the UE.

The MCPTT client in the UE uses an access token to obtain MCPTT service authorization from the MCPTT server. The MCPTT server verifies the access token, and if valid, authorizes the user for MCPTT services and returns an acknowledgement to the MCPTT client in the UE.

Upon successful MCPTT service authorization, the CM client in the UE uses an access token for configuration management authorization with the CM server to obtain the user profile. The CM server verifies the access token, and if valid, obtains the user's profile from the MCPTT user database and returns it to the CM client in the UE.

From the user's profile, the UE obtains the group memberships of the user (both on-network and off-network) and then using an access token, requests group configuration data from the Group Management server. The GM server verifies

the access token, and if valid, returns the user's group configuration data (and associated group keys) to the GM client in the UE.

Editor's Note: This assumes there is one MCPTT id per user; this may be revisited depending on SA6 decision.

To support the MCPTT identity functional model, the MCPTT ID shall be:

- Provisioned into the IdM database and mapped to MC IDs.
- Provisioned into the KMS and mapped to identity associated keys.
- Provisioned into the MCPTT user database and mapped to the user profile; and
- Provisioned into the GMS and mapped to Group IDs.

Further details of the user authorization architecture are found in clause 5.6.

5.5.2 User Authentication Framework

The framework utilises the MCPTT CSC-1 reference point as depicted in Figure 5.5.2-1:.

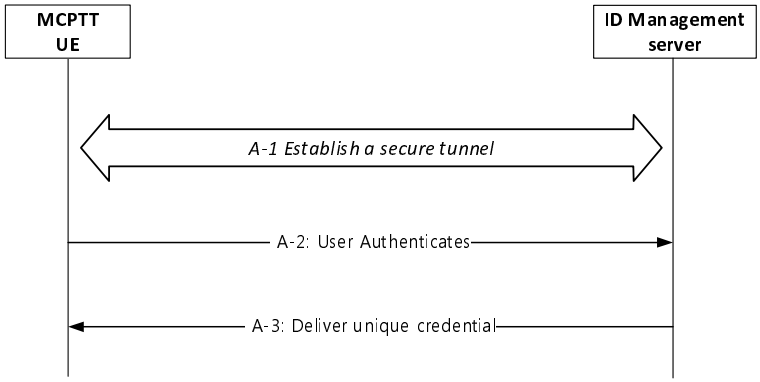


Figure 5.5.2-1: MCPTT User Authentication Framework

The User Authentication procedure in Step A of Figure 5.1-1 is further detailed into 3 sub steps that comprise the MCPTT user authentication framework:

- A-1 - Establish a secure tunnel between the MCPTT UE and Identity Management (IdM) server. Subsequent steps make use of this tunnel.
- A-2 - Perform the User Authentication Process (User proves their identity).
- A-3 - Deliver the credential, that uniquely identifies the MCPTT user, to the MCPTT client.

Following step A-3, the MCPTT client uses the credential(s) obtained from step A-3 to perform MCPTT service authorization as per procedure C in figure 5.1-1.

The framework supporting steps A-2 and A-3 shall be implemented using OpenID Connect 1.0 ([19], [20] and [21]).

NOTE: MCPTT service authorization in step C of Figure 5.1-1 is outside the scope of the User Authentication framework.

5.5.3 OpenID Connect (OIDC)

5.5.3.1 General

Figure 5.5.3.1-1 describes the MCPTT User Authentication Framework using the OpenID Connect protocol. Specifically, it describes the steps by which an MCPTT user authenticates to the Identity Management server (IdMS),

resulting in a set of credentials delivered to the UE uniquely identifying the MCPTT user's identity. The means by which these credentials are sent from the UE to the MCPTT services are out of scope of this authentication framework. The authentication framework supports extensible user authentication solutions based on MCPTT service provider policy (shown in step 3), with username/password-based user authentication as a mandatory supported method. Other user authentication methods (in step 3; e.g. biometrics, secureID, etc.) are possible but not defined here. A detailed OpenID Connect flow can be found in annex C.

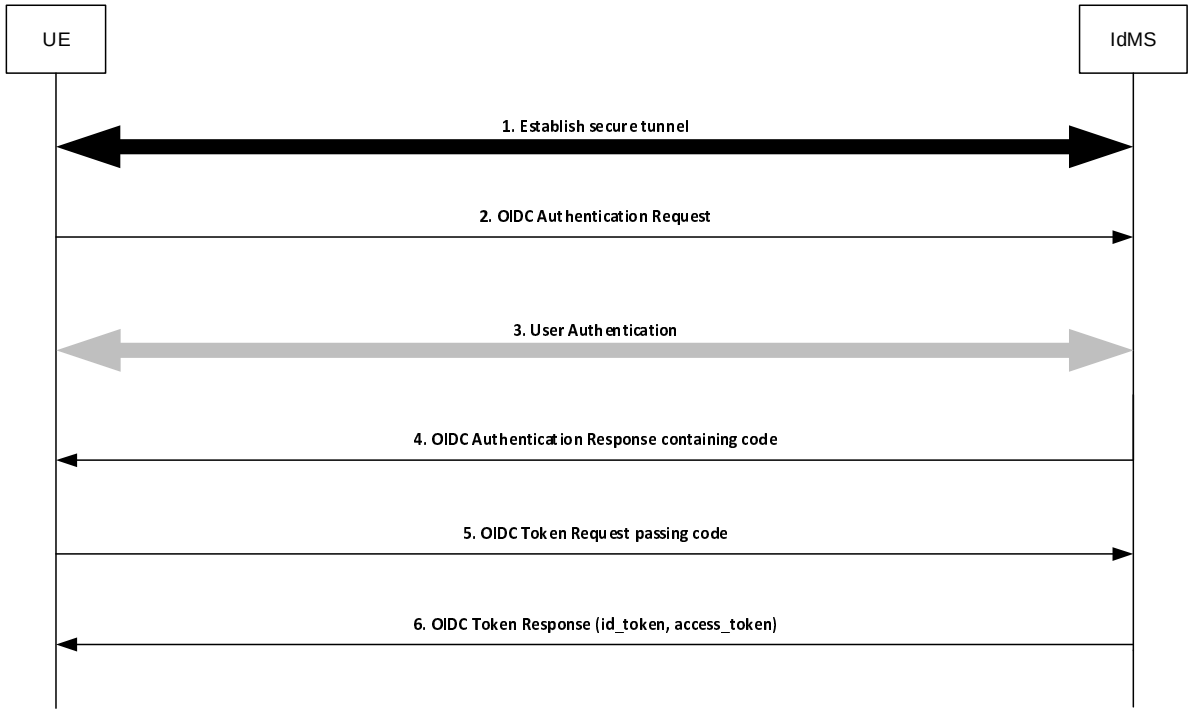


Figure 5.5.3.1-1: OpenID Connect (OIDC) flow supporting MCPTT user authentication

Step 1: UE establishes a secure tunnel with the Identity Management server (IdMS).

Step 2: UE sends an OpenID Connect Authentication Request to the IdMS. The request may contain an indication of authentication methods supported by the UE.

Step 3: User Authentication is performed.

NOTE: The primary credentials for user authentication (e.g. biometrics, secureID, OTP, username/password) are based on MCPTT service provider policy. The method chosen by the MCPTT service provider is not defined nor limited by the present document.

Step 4: IdMS sends an OpenID Connect Authentication Response to the UE containing an authorization code.

Step 5: UE sends an OpenID Connect Token Request to the IdMS, passing the authorization code.

Step 6: IdMS sends an OpenID Connect Token Response to the UE containing an id_token and an access_token (each which uniquely identify the user of the MCPTT service). The id_token is consumed by the UE to personalize the MCPTT client for the MCPTT user, and the access_token is used by the UE to communicate the identity of the MCPTT user to the MCPTT server(s).

5.5.3.2 User Authentication example using Username/Password

Figure 5.5.3.2-1 shows the OIDC MCPTT flow when Username/Password is used as the user authentication method.

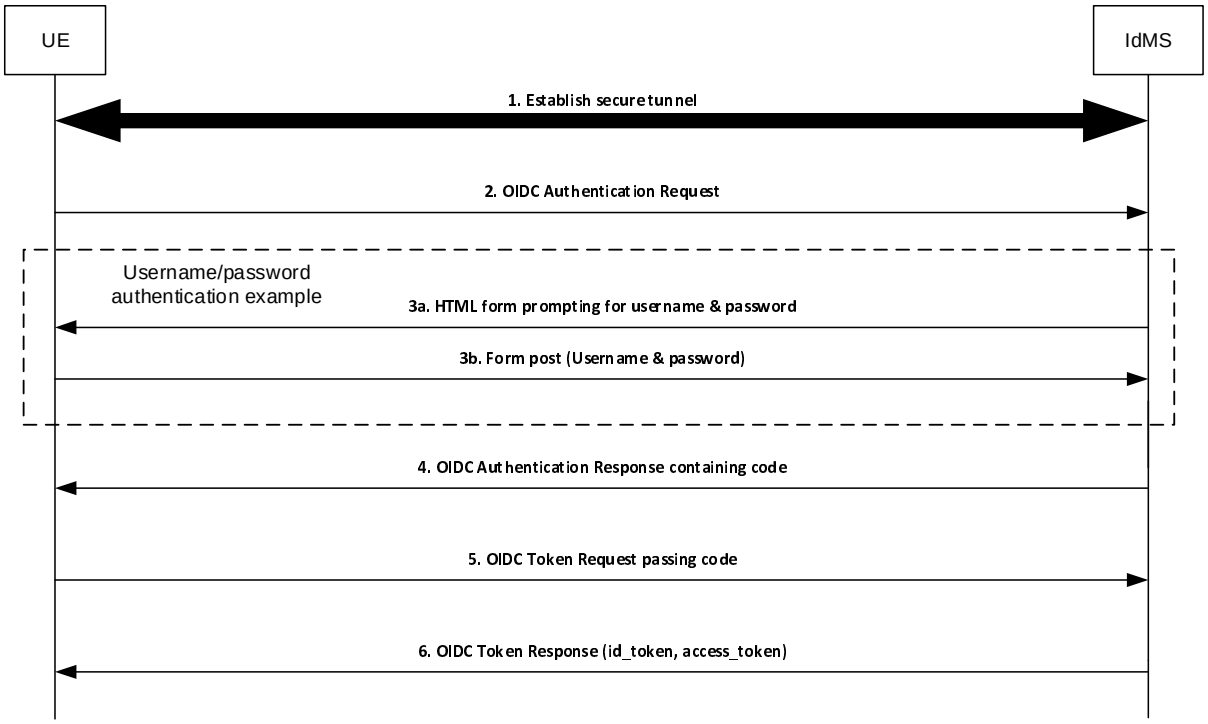


Figure 5.5.3.2-1: OpenID Connect (OIDC) Example Using Username/Password

Step 1: UE establishes a secure tunnel with the Identity Management server (IdMS).

Step 2: UE sends an OpenID Connect Authentication Request to the IdMS. The request may contain an indication of authentication methods supported by the UE.

Step 3a: IdMS sends an HTML form to UE prompting the user for their username & password.

Step 3b: UE sends the username & password (as provided by the user) to the IdMS.

Step 4: IdMS sends an OpenID Connect Authentication Response to the UE containing an authorization code.

Step 5: UE sends an OpenID Connect Token Request to the IdMS, passing the authorization code.

Step 6: IdMS sends an OpenID Connect Token Response to the UE containing an id_token and an access_token (each which uniquely identify the user of the MCPTT service). The id_token is consumed by the UE to personalize the MCPTT client for the MCPTT user, and the access_token is used by the UE to communicate the identity of the MCPTT user to the network entities.

5.6 MCPTT User Authorization

5.6.1 General

This clause expands on the MCPTT User Service Authorization step shown in figure 5.1-1 step C.

MCPTT User Service Authorization is the MCPTT function that validates whether or not a MCPTT user has the authority to access certain MCPTT services. In order to gain access to MCPTT services, the MCPTT client in the UE presents an access token (acquired during user authentication as described in subclause 5.5) to each service of interest (i.e. MCPTT Key Management, MCPTT user services, etc.). If the access token is valid, then the user is granted the use of that service. Figure 5.6.1-1 shows the flow for user authorization, which covers key management authorization, MCPTT user service authorization, configuration management authorization, and group management authorization.

NOTE: All HTTP traffic between the UE and KMS, and all HTTP traffic between the UE and HTTP proxy is protected using HTTPS.

For key management authorization, the KM client in the UE presents an access token to the KMS over HTTP. The KMS validates the access token and if successful, provides user specific key material back to the UE KM client based on the MCPTT ID of the user. This includes identity based key information used for media and signalling protection.

For user service authorization, the MCPTT client in the UE presents an access token to the MCPTT server over SIP. The MCPTT server validates the access token and if successful, authorizes the user for full MCPTT services and sends an acknowledgement back to the MCPTT client. The MCPTT server then maps and maintains the IMPU to MCPTT ID association. The MCPTT ID to IMPU association shall only be known to the application layer. The SIP message used to convey the access token from the MCPTT client to the MCPTT server may be either a SIP REGISTER or SIP PUBLISH message.

The UE can now perform configuration management authorization and download the user profile. Following the flow described in subclause 10.1.4.2 of 3GPP TS 23.179 [2] "MCPTT user obtains the user profile (UE initiated)", the Configuration Management (CM) client in the UE sends an access token in the user profile query to the Configuration Management server over HTTP. The CM server receives the request and validates the access token, and if valid, the CM server uses the MCPTT ID to obtain the user profile from the MCPTT user database. The CM server then sends the user profile back to the CM client over HTTP.

Upon receiving the user's profile, the Group Management (GM) client in the UE can now perform group management authorization. The GM client obtains the user's group membership information from the user's profile, and following the flow shown in clause 10.1.5.2 of 3GPP TS 23.179 [2] "Retrieve group configurations at the group management client", the Group Management (GM) client in the UE sends an access token in the Get group configuration request to the host GM server of the group membership over HTTP. The GM server validates the access token, and if valid, completes the flow. As part of group management authorization, group key information is provided as per subclause 7.3.2 of the present document.

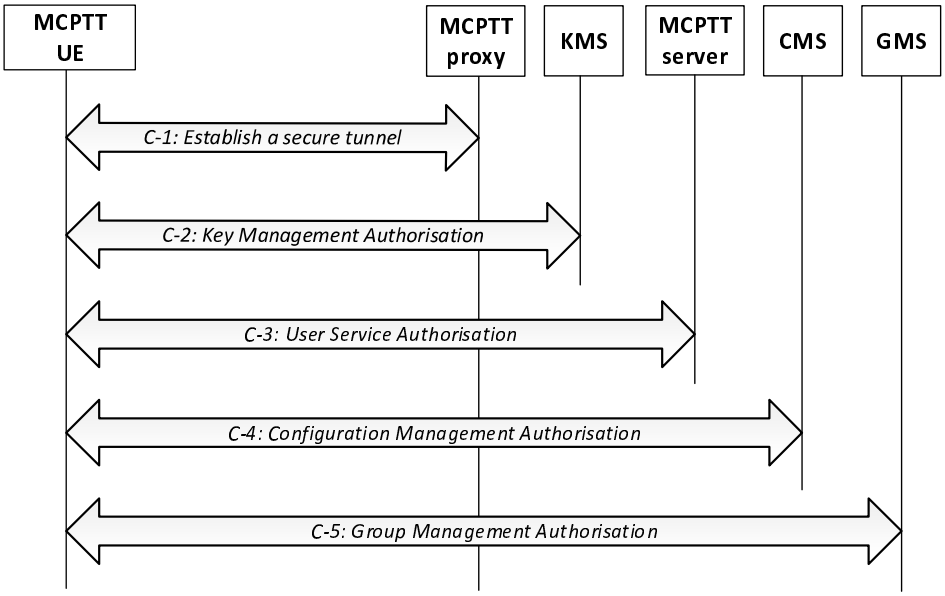


Figure 5.6.1-1: MCPTT user authorization

The user authorization procedure in Step C of Figure 5.1-1 is further detailed into 5 sub steps that comprise the MCPTT user service authorization process:

Step C-1: If not already done, establish a secure HTTP tunnel using HTTPS between the MCPTT UE and MCPTT proxy server. Subsequent HTTP messaging makes use of this tunnel.

Step C-2: The KMS client in the UE presents an access token to the KMS over HTTP. The KMS authorizes the user for key management services and replies to the client with identity specific key information.

Step C-3: The MCPTT client in the UE presents an access token to the MCPTT server over SIP as defined in clause 5.6.2 of the present document.

Step C-4: The CM client in the UE follows the "MCPTT user obtains the user profile (UE initiated)" flow from clause 10.1.4.2 of 3GPP TS 23.179 [2], presenting an access token in the Get MCPTT user profile request over HTTP. If the token is valid, then the CM server authorizes the user for configuration management services. Completion of this step results in the CM server providing the user's profile to the CM client.

Step C-5: The GM client in the UE follows the "Retrieve group configurations at the group management client" flow as shown in clause 10.1.2 of 3GPP TS 23.179 [2], presenting an access token in the Get group configuration request over HTTP. If the token is valid, the CM server authorizes the user for group management services. Completion of this step results in the GMS sending the user's group policy information and group key information to the GM client.

5.6.2 MCPTT user service authorization with MCPTT Server

5.6.2.0 General

Depending on implementation, MCPTT user service authorization may be performed by sending the access token to the MCPTT server over the SIP-1 and SIP-2 reference points using either a SIP REGISTER message or a SIP PUBLISH message. Clause 5.6.2.1 describes how to use the SIP REGISTER message to transport the access token to the MCPTT server and clause 5.6.2.2 describes how to use the SIP PUBLISH message to transport the access token to the MCPTT server.

During initial SIP registration, the SIP REGISTER message shall not be delayed for lack of an access token. If an access token is not available then SIP registration shall proceed without the inclusion of the access token and the access token shall be transmitted to the MCPTT server as per Step C-3 in figure 5.6.1-1.

If an access token is available before SIP registration, or if the UE becomes de-registered and a SIP re-registration is required, the SIP REGISTER message may include the access token without requiring the user to re-authenticate.

The access token may be sent over SIP to the MCPTT server to re-bind an IMPU and MCPTT ID if either have changed (e.g. IMPU is different due to SIP deregistration/SIP re-registration, or user logs out and another user logs onto the same UE).

5.6.2.1 Using SIP REGISTER

The use of a SIP REGISTER message to provide the access token to the MCPTT server is shown in figure 5.6.2.1-1. The inclusion of an access token in any particular SIP REGISTER message is optional.

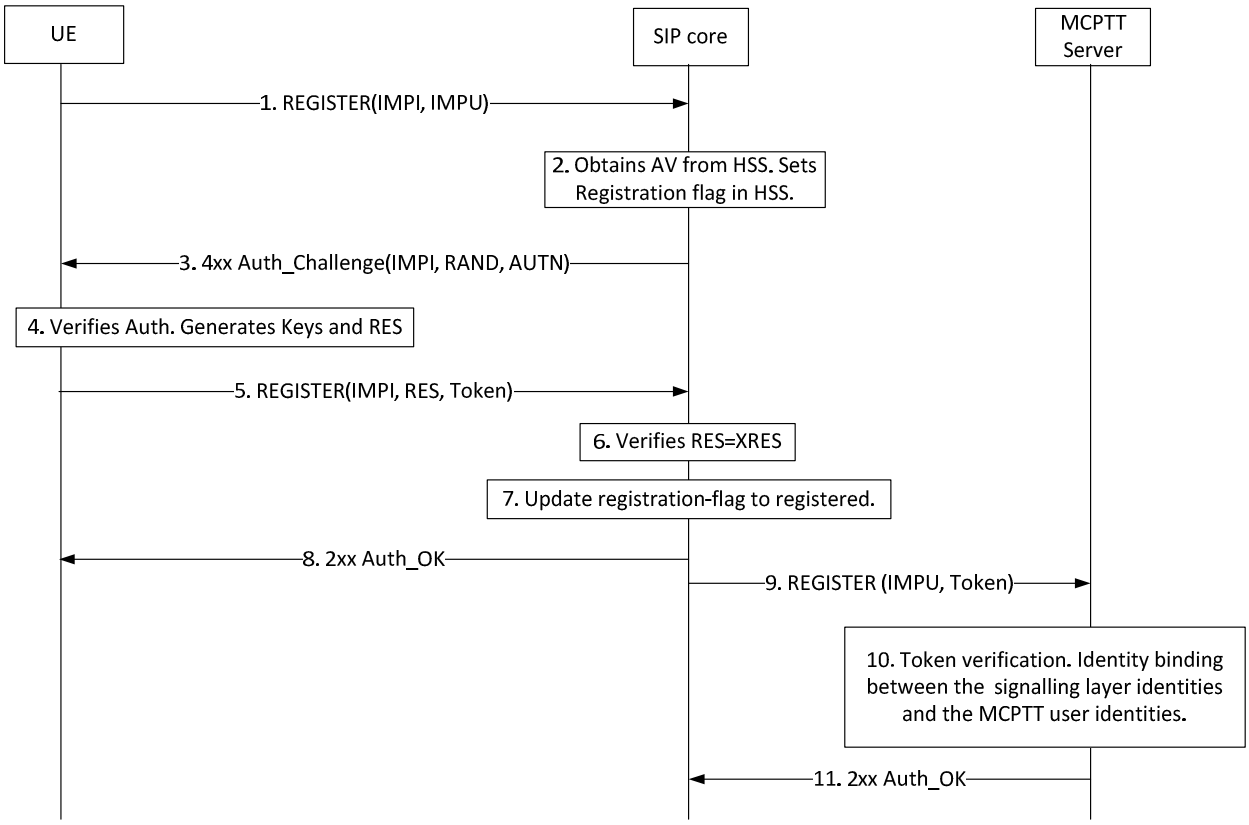


Figure 5.6.2.1-1: MCPTT User Service Authorization using SIP REGISTER message

Step 5 of figure 5.6.2.1-1 shows the access token message passed to the SIP core in a SIP REGISTER. Upon successful SIP authentication, the SIP core forwards the access token to the MCPTT server in the third part registration request message (Step 9).

In Steps 9 through 11, the MCPTT server receives the third part registration request message, validates the access token, binds the IMPU and MCPTT ID (if the access token is valid), and responds to the 3rd party registration message.

5.6.2.2 Using SIP PUBLISH

The use of a SIP PUBLISH message to provide the access token to the MCPTT server is shown in figure 5.6.2.2-1. The inclusion of an access token in any particular SIP PUBLISH message is optional.

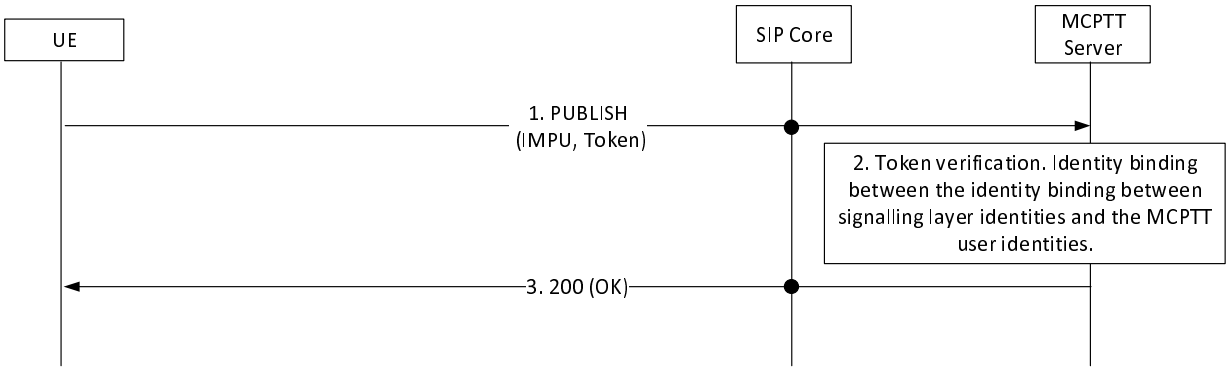


Figure 5.6.2.2-1: MCPTT User Service Authorization using SIP PUBLISH message

As shown in Step 1 of figure 5.6.2.2-1, the SIP PUBLISH message carries the access token through the SIP core to the MCPTT server.

In Steps 2 and 3, the MCPTT server receives the SIP PUBLISH message, validates the access token, binds the IMPU and MCPTT ID (if the access token is valid), and responds to the SIP PUBLISH message.

6 Signalling plane protection

6.1 SIP-1 interface security

The security mechanisms as specified in 3GPP TS 33.203 [6] for Gm interface shall be used to provide confidentiality and integrity of signalling on SIP-1 interface.

6.2 HTTP-1 interface security

The support of Transport Layer Security (TLS) on HTTP-1 is mandatory. The profile for TLS implementation and usage shall follow the provisions given in 3GPP TS 33.310 [5], annex E.

If the PSK TLS based authentication mechanism is supported, the HTTP client in the MCPTT UE and the HTTP Proxy shall support the TLS version, PSK ciphersuites and TLS Extensions as specified in the TLS profile given in 3GPP TS 33.310 [5], annex E. The usage of pre-shared key ciphersuites for TLS is specified in the TLS profile given in 3GPP TS 33.310 [5], annex E.

7 End-to-end communication security

7.1 Overview

This clause details the procedures for MCPTT users communicating using end-to-end security. This provides assurance to MCPTT users that no unauthorized access to communications is taking place within the MCPTT network. End-to-end communication security can be applied to media and when operating off-network, to floor control signalling.

An MCPTT Key Management Server (KMS) manages the security domain. For any end-point to use or access end-to-end secure communications, it shall be provisioned with key material associated to its identity by the KMS. Through the use of the KMS, MCPTT administrators are able to manage access to communications within the MCPTT network.

NOTE 1: For the purposes of this release, it is assumed that all MCPTT users are within a single security domain managed by a single KMS.

Key provisioning for group calls is performed by a group management server, authorized and provisioned by the KMS. The group management server is responsible for distributing the key material to MCPTT users within the group. With the group security context established, MCPTT users can communicate using end-to-end security.

Prior to protecting group calls during off-network operation, the UE shall acquire the necessary group key material either while operating on-network or through offline provisioning.

NOTE 2: It is a deployment option whether the MCPTT Server is included in the end-to-end security context. Where the MCPTT Server is not included in the security context, it will be unable to mix content on behalf of the users.

Key provisioning for private calls is performed by the initiating UE as the call is setup. This creates an end-to-end security context that is unique to the pair of users involved in the call. With a security context established, it may be used to encrypt media and, when off-network, floor control traffic between the end-points.

Prior to protecting private calls during off-network operation, the UE shall acquire the necessary individual key material either while operating on-network or through offline provisioning.

End-to-end security is independent of the transmission path and hence is applicable to both on and off-network communications.

7.2 Key provisioning and management

7.2.1 General

To be able to be involved in end-to-end communication security the MCPTT user requires key material to be provisioned from a MCPTT Key Management Server (KMS). In addition, management entities which setup or control the end-to-end communication, such as the MCPTT Server and group management server, will also require provisioning of key material.

NOTE: For clarity, an MCPTT KMS provides different functionality to a MIKEY-TICKET KMS defined in 3GPP TS 33.328 [8].

7.2.2 Functional model for key management

7.2.2.0 General

Within the MCPTT architecture, the MCPTT Key Management Server (KMS) provisions key material associated with a specific MCPTT identity. The MCPTT KMS has interfaces with the key management clients. A key management client is responsible for making requests for identity-specific key material. Key provisioning clients are located in the MCPTT UE, in the MCPTT Server and in the group management server.

The reference points for the MCPTT KMS are shown in figure 7.2.2.0-1.

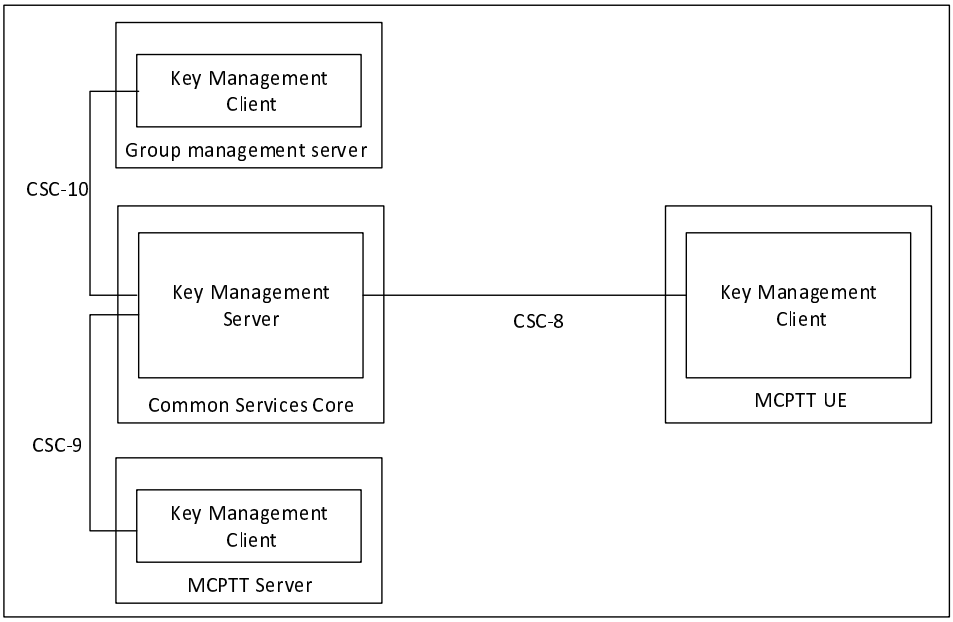


Figure 7.2.2.0-1: Reference Points for MCPTT Key Management Server

Figure 7.2.2.0-1 shows the CSC-8, CSC-9 and CSC-10 reference points for the MCPTT Key Management Server within the MCPTT system.

The KMS may or may not be located within the Common Services Core (CSC) of the MCPTT domain and may or may not make use of the MCPTT proxy.

If the KMS does not make use of the MCPTT proxy, then a secure HTTP connection (HTTPS) shall be established directly between the KMS server and the KMS client. The use of the TrK as defined in clause 9.3 may be used to protect the key material content in this configuration.

If the KMS does connect to and employ the use of the MCPTT proxy, then for public safety users the TrK shall be used as defined in clause 9.3 to protect the key material content.

7.2.2.1 Reference point CSC-8 (between key management server and the key management client within the MCPTT UE)

The CSC-8 reference point, between the key management client in the MCPTT UE and the MCPTT KMS, provides identity-specific key material to the MCPTT UE.

If the KMS does not employ the MCPTT proxy, then CSC-8 is a direct HTTP interface between the KMS server and the KMS client in the UE. CSC-8 therefore does not pass through the MCPTT proxy.

If the MCPTT proxy is used between the KMS and the KMS client within the MCPTT UE, then CSC-8 shall use the HTTP-1 and HTTP-2 reference points.

7.2.2.2 Reference point CSC-9 (between the key management server and the key management client within the MCPTT Server)

The CSC-9 reference point, which exists between the MCPTT key management server and the MCPTT Server, is used, where necessary, to provide the MCPTT Server with identity-specific key material to allow the MCPTT Server to be involved in end-to-end secure communications.

If the KMS does not employ the MCPTT proxy, then CSC-9 is a direct HTTP interface between the KMS server and the KMS client in the MCPTT server. CSC-9 therefore does not pass through the MCPTT proxy.

If the MCPTT proxy is used between the KMS and the KMS client within the MCPTT Server, then CSC-9 shall use the HTTP-1 and HTTP-2 reference points for the transport of key material.

7.2.2.3 Reference point CSC-10 (between the key management server and the key management client within a group management server)

The CSC-10 reference point, which exists between the MCPTT key management server and a group management server, is used to provide the group management server with identity-specific key material to allow the group management server to distribute key material to support group communications.

If the KMS does not employ the MCPTT proxy, then CSC-10 is a direct HTTP interface between the KMS server and the KMS client in the group management server. CSC-10 therefore does not pass through the MCPTT proxy.

If the MCPTT proxy is used between the KMS and the KMS client within a group management server, then CSC-10 shall use the HTTP-1, and HTTP-2 signalling reference points for the transport of key material.

7.2.3 Security procedures for key management

The procedure for the provision of identity-specific key material when the MCPTT proxy is supported between the KMS and the KMS client is described in figure 7.2.3-1. The procedure is the same whether the key management client in the MCPTT UE, MCPTT Server or group management server is making the request.

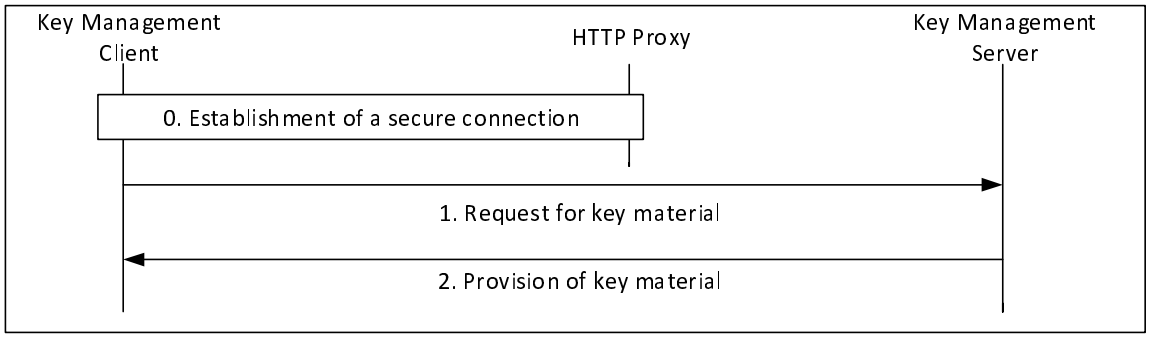


Figure 7.2.3-1: Provisioning of key material via the HTTP proxy

The procedure in figure 7.2.3-1 is now described step-by-step.

0) The key management client establishes a connection to the MCPTT KMS. As with other elements in the Common Services Core, the connection routed via, and secured by, the HTTP Proxy. The message flow below is within this secure connection.

NOTE: Additionally, the connection between the MCPTT KMS and the HTTP Proxy is secured according to clause 8.

1) The key management client makes a request for user key material from the MCPTT KMS. The request contains details of the identity (e.g. the MCPTT ID) requested for key management, and the time for which the key material is required.

2) The KMS provides a response containing key material. The response includes the type of key material, the period of use for the material and any domain-specific parameters required for its use. For public safety use, the key material itself shall be wrapped using a 256-bit transport key (TrK). The TrK is distributed via an out-of-band mechanism along with a 32-bit identifier, TrK-ID.

The procedure for the provisioning of identity-specific key material when the MCPTT proxy is not used between the KMS and the KMS client is as described in Figure 7.2.3-2.

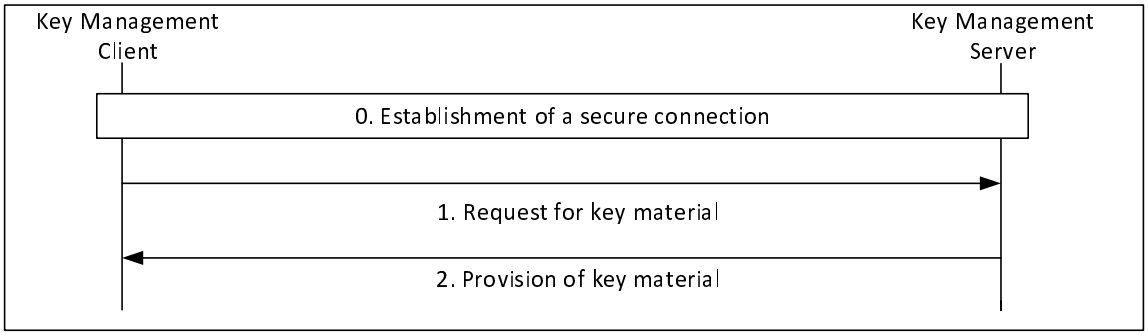


Figure 7.2.3-2: Provisioning of key material without a proxy

The procedure in Figure 7.2.3-2 is now described step-by-step:

0) The key management client establishes a direct HTTPS connection to the MCPTT KMS. The following message flow is within this secure connection.

1) The key management client makes a request for user key material from the MCPTT KMS. The request contains details of the identity requested for key management, and the time at which the key material is required.

2) The KMS provides a response containing key material. The response includes the type of key material, the period of use for the material and any domain-specific parameters required for its use. Optionally, the key material itself may also be wrapped using a 256-bit transport key (TrK), distributed via an out-of-band mechanism along with a 32-bit identifier (TrK-ID).

As a result of this procedure, the key management client has securely obtained key material for use within the MCPTT system.

7.2.4 Provisioned key material to support end-to-end communication security

End-to-end communication security for either group or private calls requires the provisioning of key material from the KMS. The key material required to be provisioned to each user is listed below:

- Domain specific key material, also known as a MCPTT KMS Certificate, which includes:
  - The MCPTT KMS Public Authentication Key (KPAK in IETF RFC 6507 [9]).
  - The MCPTT KMS Public Confidentiality Key (Z_T in IETF RFC 6508 [10]).
  - The UID conversion (as described below).
  - Choice of cryptographic domain parameters (such as those listed in IETF RFC 6509 [8]).
  - The time period for which this information is valid.
- A user signing key for each UID for the upcoming time period (SSK and PVT in IETF RFC 6507 [9]).
- A user decryption key for each UID for the upcoming time period (RSK in IETF RFC 6508 [10]).
- The time period, for which the user key material is valid (e.g. month).

The UID conversion mechanism defines how UIDs are generated. Using this information a MCPTT client can take a user identifier (e.g. an MCPTT ID), and the current time, (e.g. the year and month) and convert these to a UID.

EXAMPLE: UID = Hash (MCPTT ID, KMS URI, validity period info).

As a consequence, there is a one-to-one correspondence between MCPTT IDs and UIDs during each time period.

After provisioning, the key material may be stored in the user's profile.

7.3 Group call key distribution

7.3.1 General

To create the group's security association, a Group Master Key (GMK) and associated identifier (GMK-ID) is distributed to MCPTT UEs by a Group Management Server (GMS). The GMK is distributed encrypted specifically to a user and signed using an identity representing the Group Management Server. Prior to group key distribution, each MCPTT UE within the group shall be provisioned by the MCPTT Key Management Server (KMS) with time-limited key material associated with the MCPTT User as described in clause 7.2. The Group Management Server shall also be provisioned by the MCPTT KMS with key material for an identity which is authorized to create groups.

The GMK is distributed within a Group Key Transport payload. This payload is a MIKEY-SAKKE I_MESSAGE, as defined in IETF RFC 6509 [11], which ensures the confidentiality, integrity and authenticity of the payload. The GMK is distributed with a 32-bit Group User Key Identifier (GUK-ID). The GUK-ID is generated from the GMK-ID and a User Salt derived from the user's MCPTT ID.

The GMK is encrypted to the user identity (UID) associated to the MCPTT UE. The UID used to encrypt the data will be derived from the user's MCPTT ID and a time stamp as described in clause 7.2. The user's MCPTT ID is added to the recipient field (IDRr) of the message.

The Group Key Transport payload is signed using (the KMS-provisioned key associated to) the identity of the Group Management Server (GMS). This identity is derived from the GMS's URI (e.g. gp.manager@mcptt.example.org) and a time stamp. The GMS's URI is added to the initiator field (IDRi) of the message.

The GMK is provided together with an activation time, which is the time from which the key should be used for transmission of group communications, replacing previous keys assigned to that group; the group identity for confirmation of the association of the GMK to the group; and an optional text payload which may be used to provide user-readable text. These associated parameters are encrypted by the GMK and carried in the MIKEY-SAKKE I_Message in the group key transport payload.

The security processes are summarized in figure 7.3.1-1 and further detailed below.

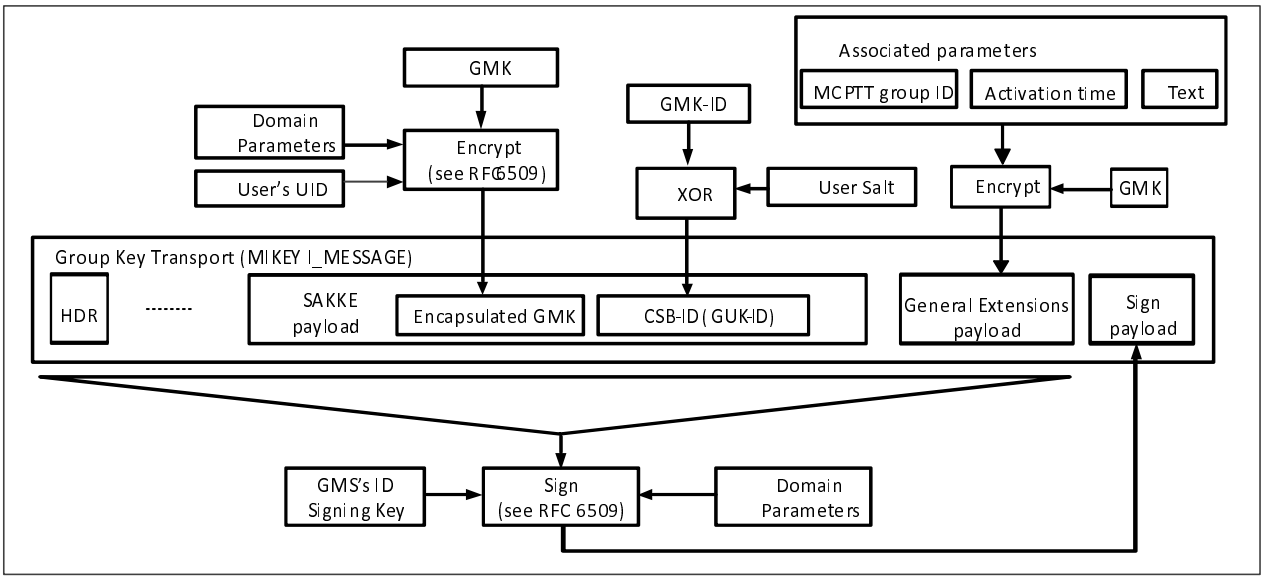


Figure 7.3.1-1: Generation of a group key transport message (encapsulation of GMK)

At the MCPTT UE, the GMS's URI is extracted from the initiator field (IDRi) of the message. Along with the time, this is used to check the signature on the Group Key Transport message. If valid, the UE extracts and decrypts the encapsulated GMK using the (KMS-provisioned) user's UID key. The MCPTT UE also extracts GUK-ID and xors the GUK-ID and User Salt together to extract the GMK-ID. If the Status field in the GMK parameters indicate the GMK has been revoked, the GMK and GMK-ID shall not be used.

The extraction procedure is described in figure 7.3.1-2.

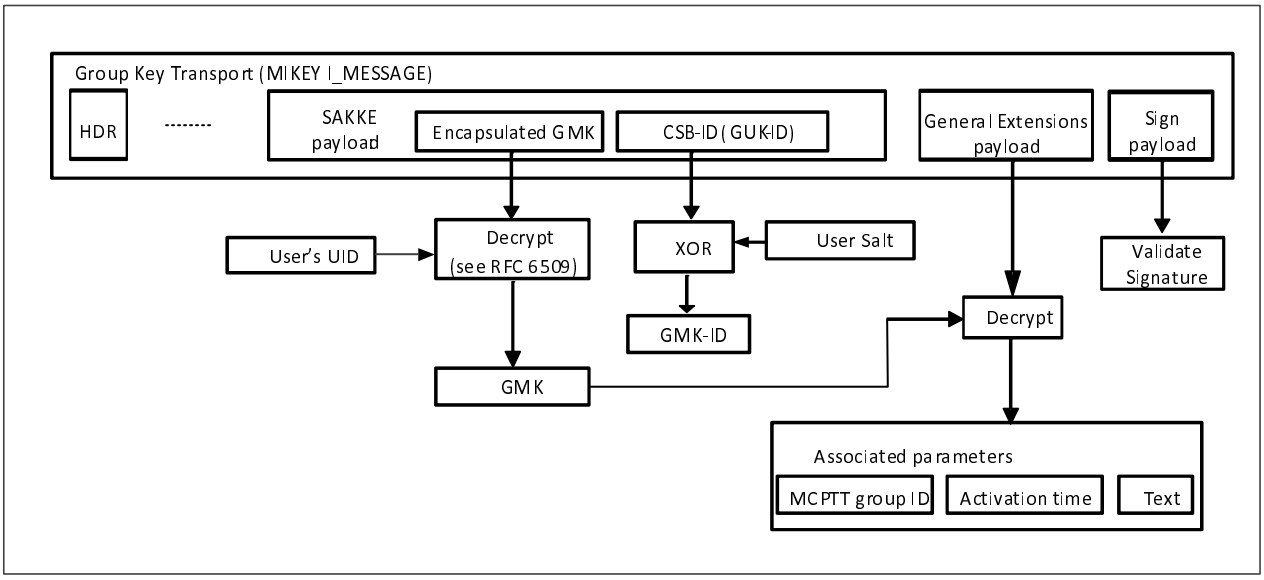


Figure 7.3.1-2: Processing of a group key transport message (extraction of GMK)

Following successful extraction of the GMK, the MCPTT UE decrypts and extracts the MCPTT group ID carried in the encapsulated associated parameters payload. The MCPTT UE stores the GMK together with the group identity, activation time and optional text field. If the decryption process for the encapsulated associated parameters fails, the GMK is rejected.

The Group Key Transport payload includes the Group User Key Identifier (GUK-ID) within the CSB-ID field. The GMK is unique within the MCPTT system and is identified by a GMK Identifier (GMK-ID) from which the GUK-ID is derived. On creating the GMK, the Group Manager generates a GMK-ID as follows. The 4 most significant bits of the

GMK-ID is the 'purpose tag' which defines the purpose of the GMK. The 28 least significant bits of the GMK-ID is a 28-bit randomly-generated value.

For each user, the GMS creates a 28-bit User Salt by hashing the user's MCPTT ID through a KDF using the GMK as the key as defined in clause F.1.3. The User Salt is xor'd with the 28 least-significant bits of the GMK-ID to create the 32-bit GUK-ID. The process for generating the GUK-ID is summarized in figure 7.3.1-3.

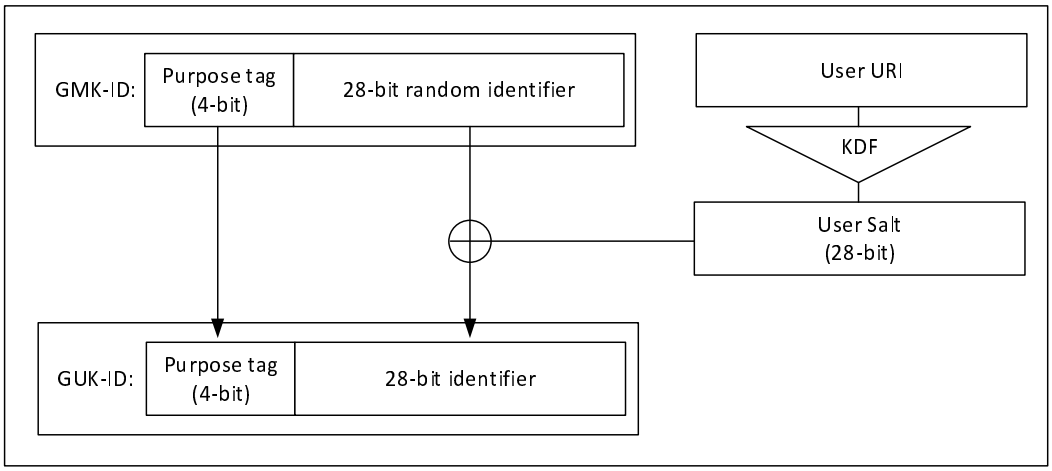


Figure 7.3.1-3: Generating the GUK-ID

The GUK-ID is placed in the CSB ID field within the header of the I_MESSAGE.

NOTE: Knowledge of the GUK-ID, GMK-ID and User Salt does not reveal the MCPTT ID to receivers without the GMK for the group.

7.3.2 Security procedures for GMK provisioning

This procedure distributes a MIKEY payload from the GMS to MCPTT UEs within the group. The payload is transported as part of the 'Notify group configuration request' message defined in clause 10.1.5.3 of 3GPP TS 23.179 [2].

Figure 7.3.2-1 shows the security procedures for creating a security association for a group.

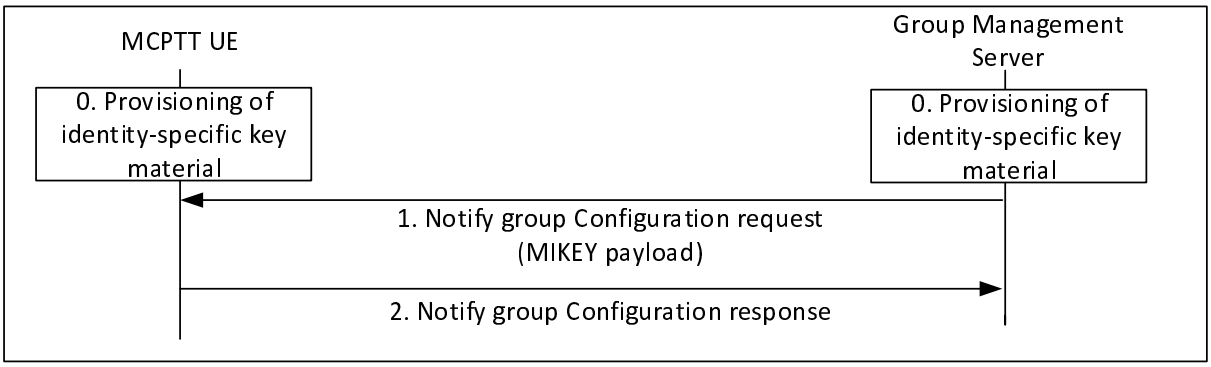


Figure 7.3.2-1: Security configuration for groups

A description of the procedures depicted in figure 7.3.2-1 follows. For clarity, step 1 corresponds to step 2 in figure 10.1.5.3-2 of clause 10.1.5.3 of 3GPP TS 23.179 [2].

0) Prior to beginning this procedure the MCPPT client shall be provisioned with identity-specific key material by a MCPTT KMS as described in clause 7.2. The GMS shall also be securely provisioned with identity-specific key material for an identity that is authorized to create groups.

1) The GMS shall send a MIKEY payload to MCPTT clients within the group within a 'Notify group configuration request' message. The message shall encapsulate a GMK for the group. The payload shall be encrypted to the

user identity (MCPTT ID) associated to the MCPTT client and shall be signed by the GMS. The message shall also provide the GUK-ID. Parameters associated with the GMK shall be encrypted using the GMK, and sent in the MIKEY payload together with the encapsulated GMK. This process is shown in Figure 7.1.1-1.

2) On receipt of a MIKEY message, the MCPTT client shall check the signature on the payload, verify that the GMS is authorized to create groups, extract the GMK, GUK-ID and GMK-ID and check that the GMK-ID is not a duplicate for an existing GMK. The MCPTT client shall also extract the group identity, activation time and text from the encapsulated associated parameters in the payload using the GMK, and check that decryption is successful. This process is shown in Figure 7.1.1-2.Should any of these checks fail, an error shall be returned to the GMS. Upon successful receipt and processing, the MCPTT UE shall store the GMK, GMK-ID and GUK-ID and respond to the GMS with a 'Notify group configuration response' message.

Where multicast floor control is required on the downlink from the MCPTT Server to the MCPTT client, the procedure may be performed twice, once for obtaining a GMK for the protection of media, and once for obtaining a different key for the protection of multicast floor control, the Multicast Floor Control Key (MKFC). Alternatively, the GMS may distribute the two different keys (GMK, MKFC) in one procedure by embedding the two MIKEY payloads in the message. The MKFC shall be treated as a GMK for transport, using the security procedures defined in clause 7.3.1 and being encapsulated as defined in Annex E.2.

To revoke a security context, the group management server repeats the above steps with the Status field of the GMK parameters indicating that the GMK has been revoked.

7.3.3 Key Identification and purpose tags

The 'purpose tag' within the key identifier (e.g. GMK-ID) shall be the most significant four bits of the key and shall be used to indicate the use of the key.

- 0: the GMK shall be used for group communications.
- 1: the PCK shall be used to protect Private Call communications.
- 2: the CSK shall be used to protect application signalling (XML and SRTCP) between the MCPTT client and MCPTT domain.
- 3: the SPK shall be used to protect application signalling (XML and SRTCP) between servers in MCPTT domain(s).
- 4: The MKFC shall be used to protect multicast floor control signalling from the MCPTT Server to MCPTT clients.
- 5-15: not defined.

In this way, the MCPTT UE is able to identify the purpose of the key.

7.3.4 Group creation procedure

Group creation procedure is described in clause 10.4.3 of 3GPP TS 23.179 [2]. To create the security context for the group, the GMS follows the procedures in clause 7.3.1, creating a new GMK and GMK-ID for the temporary group.

An encapsulated GMK and GUK-ID is sent to group members by the GMS within a notification message (step 4 within clause 10.4.3 of 3GPP TS 23.179 [2]). The procedure is equivalent to that described in clause 7.3.2.

7.3.5 Dynamic group keying

7.3.5.1 General

In the GMK distribution procedures described in this clause, the GMS is provisioned with the same information as any MCPTT UE by the KMS as described in clause 7.2; the only distinguishing feature is that the GMS's identity is authorized to create groups.

NOTE: This authorization could be conveyed within the identity itself. For example, via a specific string within the URI such as 'group'. For example, the identity user.001.group@mcptt.example.org may be authorized to create a group, whereas user.001@mcptt.example.org may not.

Additionally, the only information the GMS requires to create the group are the MCPTT IDs of the group members. These two features combined allow groups to be created and keyed at any time, by any authorized entity.

Such flexibility is required to support a number of group procedures within 3GPP TS 23.179 [2].

NOTE: The dynamic group keying mechanisms may not support off-network scenarios.

7.3.5.2 Group regrouping procedures (within a single MCPTT system)

Group Regroup procedures are described in clause 10.6.2.1 of 3GPP TS 23.179 [2]. To create the security context for the temporary group, the GMS follows the procedures in clause 7.3.1, creating a new GMK and GMK-ID for the temporary group.

An encapsulated GMK and GUK-ID is sent to group members by the GMS within a notification message (step 5 within clause 10.6.2.1 of 3GPP TS 23.179 [2]). The procedure is equivalent to that described in clause 7.3.2.

7.3.5.3 Group regrouping procedures (involving multiple MCPTT systems)

Group Regroup procedures involving multiple MCPTT systems are described in clause 10.6.2.2 of 3GPP TS 23.179 [2]. figure 7.3.5.3-1.

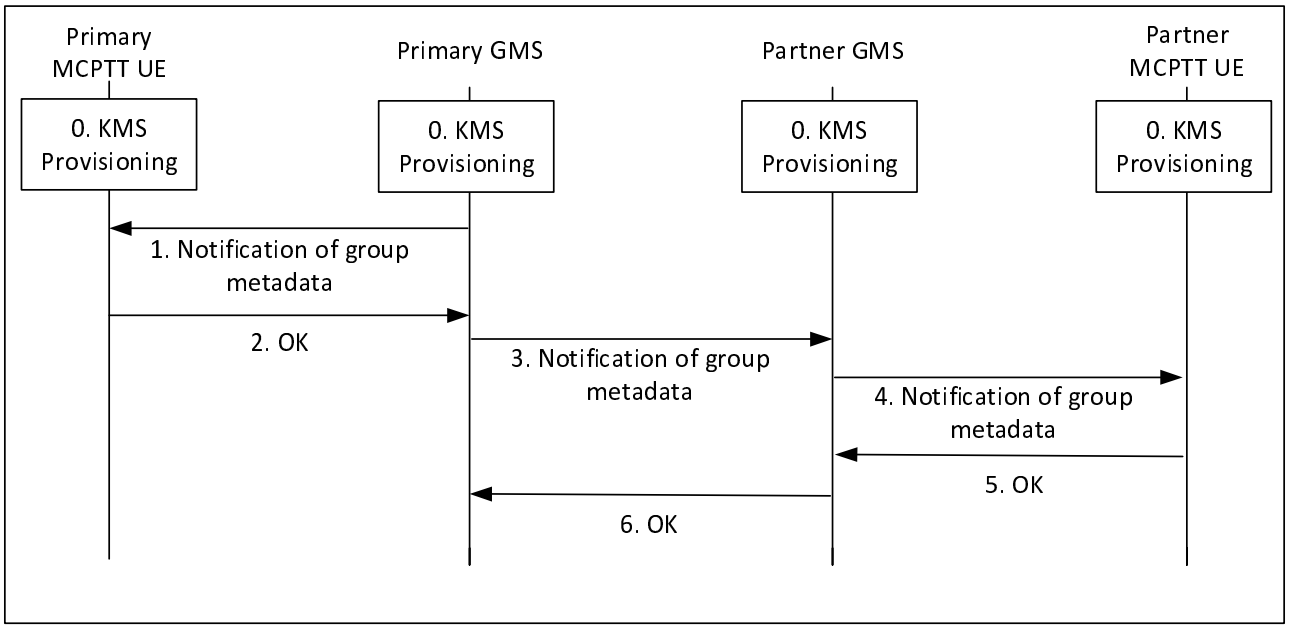


Figure 7.3.5.3-1: Group Regroup security procedures (multiple MCPTT systems)

0) Prior to beginning the procedure, the MCPTT UEs, primary GMS and partner GMSare provisioned by a KMS as described in clause 7.2.

1) To create the security context for the temporary group, the primary GMS creates a new GMK and GMK-ID for the temporary group. The primary GMS notifies the affiliated users within its own MCPTT system (Step 8 of clause 10.6.2.2 in 3GPP TS 23.179 [2]). Within this message, the primary GMS includes a Group Key Transport payload including a GMK and GUK-ID following the procedures in clause 7.3.1. The GMK is encrypted to the identity of the MCPTT user and is signed using the identity of the primary GMS.

2) The MCPTT UEs acknowledge the notification.

3) The primary GMS then notifies the partner GMS of the group regroup operation (Step 9 of clause 10.6.2.2 in 3GPP TS 23.179 [2]). Within this message, the primary GMS includes a Group Key Transport payload following the procedures in clause 7.3.1, treating the partner GMS as another user within the group. Accordingly, the

payload encrypts the new GMK to the identity of the partner GMS and is signed using the identity of the primary GMS. The GUK-ID is derived using the User Salt generated from the partner GMS's URI.

4) The partner GMS extracts the GMK and GMK-ID from the notification. The partner GMS then notifies the affiliated users within the partner MCPTT system (Step 11 of clause 10.6.2.2 in 3GPP TS 23.179 [2]). The partner GMS re-encrypts the GMK to the identity of the affiliated users in the partner system, generates new GUK-IDs for each user and signs using its identity (the identity of the partner GMS) following the procedure in clause 7.3.1.

5) The partner MCPTT UEs acknowledge the notification.

6) The partner GMS acknowledges the notification to the primary GMS.

Where multicast floor control is required on the downlink from the MCPTT Server to the MCPTT client, the procedure may be performed twice, once for obtaining a GMK for the protection of media, and once for obtaining a different key for the protection of multicast floor control, the Multicast Floor Control Key (MKFC). Alternatively, the GMS may distribute the two different keys (GMK, MKFC) in one procedure by embedding the two MIKEY payloads in the message. The MKFC shall be treated as a GMK for transport, using the security procedures defined in clause 7.3.1 and being encapsulated as defined in Annex E.2.

7.3.6 Derivation of SRTP/SRTCP master keys

As a result of this mechanism, the group members share a GMK and GUK-ID. The GMK shall be used as the MIKEY Traffic Generating Key (TGK), the GUK-ID shall be used as the MIKEY CSB ID. These shall be used to generate the SRTP Master Key and SRTP Master Salt as specified in IETF RFC 3830 [22]. The key derivation function defined in section 4.1.4 of IETF RFC 3830 [22] using the PRF-HMAC-SHA-256 Pseudo-Random Function as described in IETF RFC 6043 [25], section 6.1 shall be supported for generating the SRTP Master Key and Salt.

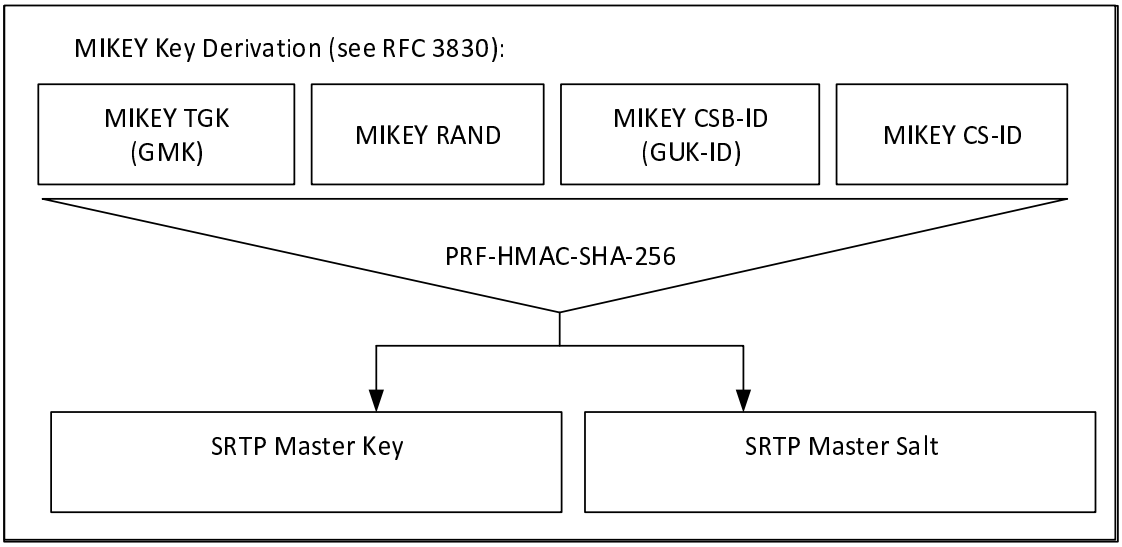


Figure 7.3.6-1: Key Derivation for media stream protection

To identify the security context from the media stream a SRTP Master Key Identifier (MKI) is required. The MKI should be a 64-bit value formed by concatenating the GMK-ID with the GUK-ID (GMK-ID || GUK-ID). The GMK-ID shall have a purpose-tag of '0'.

Where the transmitting user is known through other means, the MKI may be solely the 32-bit GMK-ID. In this case the terminating user extracts the GUK-ID by calculating the User Salt and xor'ing this value with the GMK-ID.

When the MCPTT client is operating off network, the GMK is used to derive keys for floor control (SRTCP). Thus, the Master Key and Master Salt used for SRTCP is the same with the Master Key and Master Salt used for SRTP, so is the MKI.

See clause 9.4 for key derivation procedures for floor control (SRTCP) when the MCPTT client is operating on-network.

7.4 Private call key distribution

7.4.1 General

The purpose of this procedure is to allow two MCPTT UEs to create an end-to-end security context to protect an MCPTT private call. To create the security context, the initiating MCPTT UE generates a Private Call Key (PCK) and securely transfers this key, along with a key identifier (PCK-ID), to the terminating MCPTT UE.

The PCK is distributed encrypted specifically to the terminating user and signed by the initiating user. Prior to call commencement, both MCPTT UEs shall be provisioned by the KMS with time-limited key material associated with the MCPTT User. The PCK is distributed with a 32-bit Key Identifier (PCK-ID) within a MIKEY payload within the SDP content of the Private Call Request. This payload is a MIKEY-SAKKE I_MESSAGE, as defined in IETF RFC 6509 [11], which ensures the confidentiality, integrity and authenticity of the payload.

The PCK is encrypted to the user identity (UID) associated to the terminating MCPTT UE. The UID used to encrypt the data will be derived from the terminating user's URI (e.g. user.002@mcptt.example.org) and a time-related parameter (e.g. the current year and month). The terminating user's URI is added to the recipient field (IDRr) of the message.

The payload includes the encrypted PCK and the key identifier (PCK-ID). The PCK is unique within the MCPTT system. On creating the PCK, the initiator generates a 32-bit PCK Identifier (PCK-ID). The 4 most significant bits of the PCK-ID shall indicate the purpose of the PCK is to protect Private call communications, the other 28-bits shall be randomly generated.

The payload is signed using (the KMS-provisioned key associated to) the identity of the initiating user. This identity is derived from the initiating user's URI (e.g. user.001@mcptt.example.org) and a time-related parameter (e.g. the current year and month). The initiating user's URI is added to the initiator field (IDRi) of the message.

The security processes are summarized in figure 7.4.1-1.

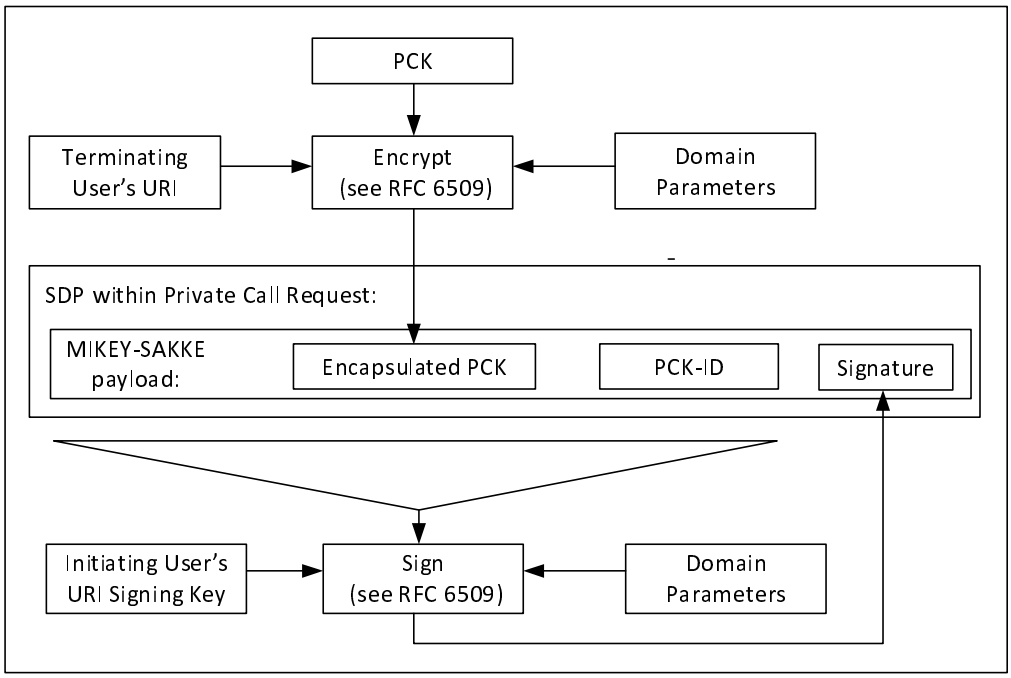


Figure 7.4.1-1: Security information within a Private Call Request

Via this mechanism, PCK distribution is confidentiality protected, authenticated and integrity protected.

At the terminating MCPTT UE, the initiating user's URI is extracted from the initiator field (IDRi) of the message. This is converted to a UID and used to check the signature on the MIKEY-SAKKE I_MESSAGE. If valid, the UE extracts

and decrypts the encapsulated PCK using the (KMS-provisioned) user's UID key. The MCPTT UE also extracts the PCK-ID. This process is shown in figure 7.4.1-2.

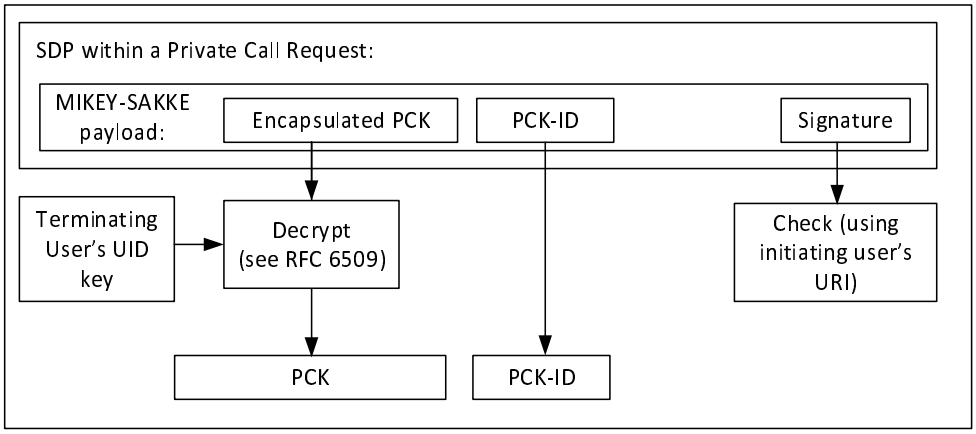


Figure 7.4.1-2: Processing the security content of a private call request

With the PCK successfully shared between the two MCPTT UEs, the UEs are able to use SRTP/SRTCP to create an end-to-end secure session.

NOTE: This solution is for the end-to-end protection of PCKs and does not protect the identities transmitted. These may be protected by other means.

7.4.2 Security procedures (on-network)

The security procedures provide a mechanism for establishing a security context as part of the Private Call Request sent from the initiating UE to the terminating UE. 3GPP TS 23.179 [2] contains four procedures for the commencing a MCPTT private call; manual or automatic commencement for a single or multiple MCPTT systems. As the security procedures are the same in each of these cases, the procedures below apply to each of the four procedures in 3GPP TS 23.179 [2].

The security processes for securing an on-network private call are summarized in figure 7.4.2-1. The intent of these procedures is to transfer a PCK and PCK-ID to the terminating UE.

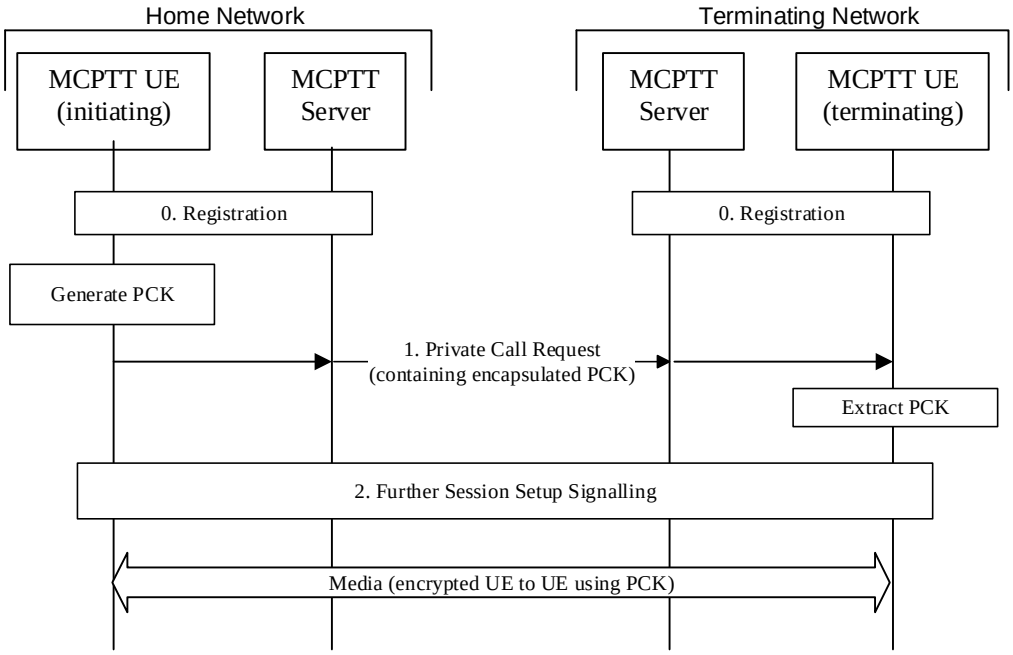


Figure 7.4.2-1: Private Call security procedures for on-network PCK distribution

The procedure in figure 7.4.2-1 is now described step-by-step.

0. Prior to beginning this procedure it is assumed that the MCPTT UEs have an authenticated MCPTT user and that the MCPTT UEs have been provisioned with key material associated with a user's MCPTT IDs by a MCPTT KMS as described in clause 7.2.

1. The initiating MCPTT UE generates the PCK and sends a MCPTT private call request to the terminating MCPTT UE. This message is created in Step 3 of clause 10.10.2.1.1, clause 10.10.2.1.2, clause 10.10.2.2.1 and clause 10.10.2.2.2 within 3GPP TS 23.179 [2]. The message is transferred via one or more MCPTT Servers, for example steps 3, 5, 8 of Clause 10.10.2.2.2 within 3GPP TS 23.179 [2]. Within this message is a SDP offer within which the initiating MCPTT UE includes a MIKEY-SAKKE I_MESSAGEs as defined in IETF RFC 6509 [11]. The I_MESSAGE encapsulates the PCK for the terminating MCPTT user, encrypting the key to the UID of the terminating user (derived from the user's URI). The I_MESSAGE also contains an identifier for the PCK (PCK-ID). The I_MESSAGE is signed using (the key associated with) the initiating user's UID.

NOTE 1: This message may be pre-generated to increase the efficiency of the communication.

NOTE 2: Optionally, the MCPTT UE may attach an additional SAKKE component which encrypts the PCK to the initiating user (in addition to the terminating user).

2. Further session signalling occurs as defined in 3GPP TS 23.179 [2].

With the PCK and PCK-ID shared between the initiating and terminating users, the media communicated between the UEs may be protected using the PCK.

7.4.3 Security procedures (off-network)

The security processes for securing an off-network private call are summarized in figure 7.4.3-1. As part of this process, the PCK and PCK-ID are securely transferred to the terminating UE.

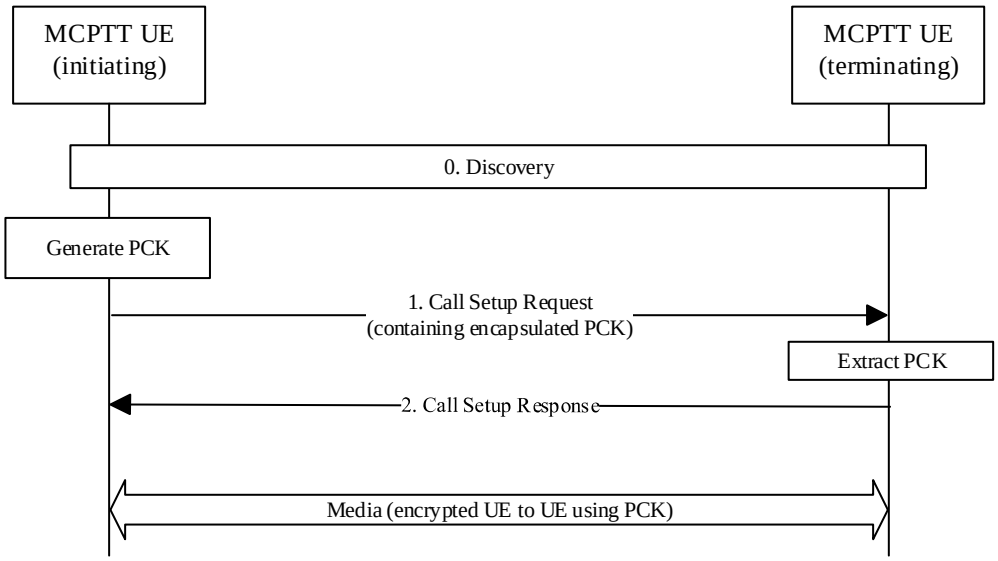


Figure 7.4.3-1: Private Call security procedures for off-network PCK distribution

The procedure in figure 7.4.3-1 is now described step-by-step.

0. Prior to beginning this procedure the MCPTT UEs may have performed a discovery procedure. Additionally, the MCPTT UEs have been provisioned with key material associated with a user's MCPTT IDs by a MCPTT KMS as described in clause 7.2.

1. The initiating MCPTT UE generates the PCK and sends a Call Setup Request to the terminating MCPTT UE (Step 4 of Clause 10.10.3.1 within 3GPP TS 23.179 [2]). Within this message, the initiating MCPTT UE includes a MIKEY-SAKKE I_MESSAGEs as defined in IETF RFC 6509 [11]. The I_MESSAGE encapsulates the PCK for the terminating MCPTT user, encrypting the key to the UID of the terminating user (derived from the user's URI). The I_MESSAGE also contains an identifier for the PCK (PCK-ID). The I_MESSAGE is signed using (the key associated with) the initiating user's UID.

NOTE 1: This message may be pre-generated to increase the efficiency of the communication.

NOTE 2: Optionally, the MCPTT UE may attached an additional SAKKE component which encrypts the PCK to the initiating user (in addition to the terminating user).

2. A Call Setup Response is returned to the initiating UE as defined in 3GPP TS 23.179 [2].

With the PCK and PCK-ID shared between the initiating and terminating users, the media communicated between the UEs may be protected using the PCK.

7.4.4 Derivation of SRTP/SRTCP master keys

As a result of this mechanism, the group members share a PCK and PCK-ID. The PCK shall be used as the MIKEY Traffic Generating Key (TGK), the PCK-ID shall be used as the MIKEY CSB ID. These shall be used to generate the SRTP Master Key and SRTP Master Salt as specified in IETF RFC 3830 [22]. The key derivation function defined in section 4.1.4 of RFC 3830 [22] using the PRF-HMAC-SHA-256 Pseudo-Random Function as described in IETF RFC 6043 [25], section 6.1 shall be supported for generating the SRTP Master Key and Salt.

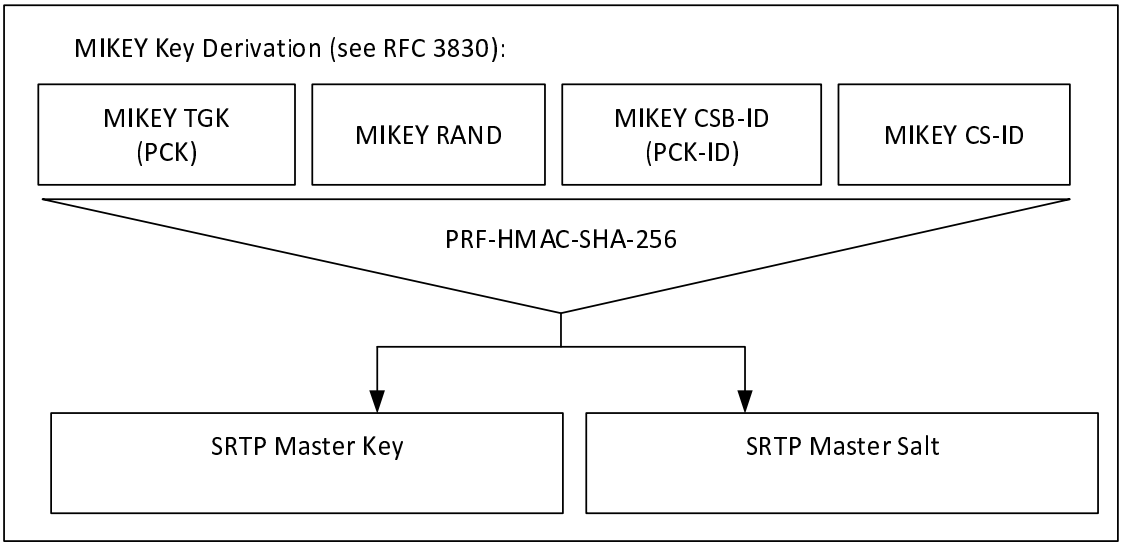


Figure 7.4.4-1: Key Derivation for media stream protection

To identify the security context from the media stream a SRTP Master Key Identifier (MKI) is required. The MKI shall be the32-bit PCK-ID which has a purpose tag of '1'.

When the MCPTT client is operating off network, the PCK is used to derive keys for floor control (SRTCP). Thus, the Master Key and Master Salt used for SRTCP is the same with the Master Key and Master Salt used for SRTP, so is the MKI.

See clause 9.4 for key derivation procedures for floor control (SRTCP) when the MCPTT client is operating on-network.

7.4.5 Void

7.5 Protection of media stream (SRTP)

7.5.1 General

The following mechanism shall be used to protect MCPTT communications which use the Real-Time Transport Protocol (RTP), cf. IETF RFC 3550 [12]. The integrity and confidentiality protection for MCPTT communications using RTP shall be achieved by using the Secure Real-Time Transport Protocol (SRTP), IETF RFC 3711 [13].

The key management mechanism for SRTP is described elsewhere. As a result of this mechanism, those communicating will have shared the following:

1) A SRTP Master Key.

2) A SRTP Master Salt.

3) A SRTP Master Key Identifier (MKI) referencing the above two values.

The mechanism described in IETF RFC 3711 [13] is used to encrypt the RTP payload. A diagram of the key derivation mechanism (as described in IETF RFC 3711) is shown in figure 7.5.1-1.

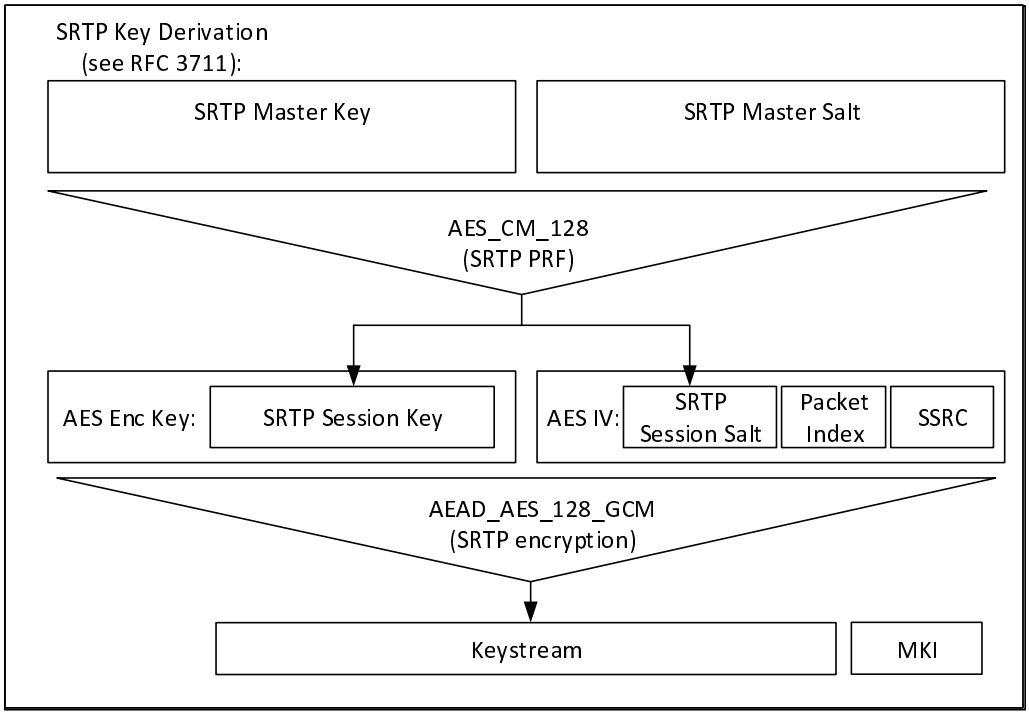


Figure 7.5.1-1: Security mechanism for media stream protection

The AES-CM-128 algorithm as defined in IETF RFC 3711 [13] shall be supported as the SRTP PRF (which is used to derive the SRTP session key and salt). A SRTP key derivation rate of 0 shall be used to indicate that session keys and salts shall not be refreshed. The AEAD_AES_128_GCM algorithm as defined in IETF RFC 7714 [26] shall be supported for providing confidentiality and data authentication of SRTP packets. The AEAD_AES_128_GCM algorithm requires that the SRTP session key is 16 octets in length, and the SRTP session salt is 12 octets in length.

The SRTP authentication tag may be appended to every 'rth' packet as defined in IETF RFC 4771 [24] to provide the SRTP ROC counter to MCPTT UEs performing a late-entry to the communication. A 'mode 3' integrity transform (RCCm3) shall be supported for transmitting the ROC within a 4 octet SRTP authentication tag.

NOTE: The ROC and MKI fields of the SRTP packet are not authenticated as part of the AEAD_AES_128_GCM algorithm. However, modification of these fields would cause a failure to validate the AEAD authentication tag which would cause the packet to be correctly rejected by the receiver. Should an unauthenticated SRTP encryption mode be used, the impact of a malicious modification of the ROC or MKI packets should be considered.

7.5.2 Security procedures for media stream protection

Media stream protection does not require any signalling mechanism to convey information. The information is provided within each SRTP Packet.

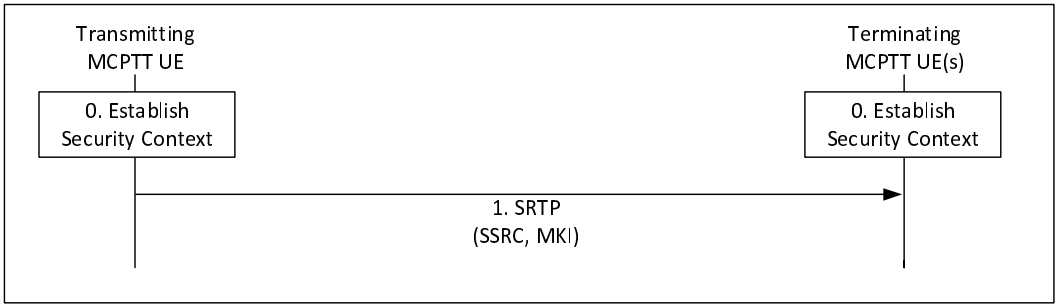


Figure 7.5.2-1: Security procedure for media stream protection

Figure 7.5.2-1 shows the security mechanism.

0) Prior to beginning this procedure the MCPPT UEs involved in the communication shall have established a security context (SRTP Master Key, SRTP Master Salt, MKI).

1) Transmitting UEs shall send SRTP packets using the format described in IETF RFC 3711 [13]. The packet shall include a Master Key Identifier (MKI) field which contains the information required to locate the SRTP Master Key and Master Salt, and may include the SRTP ROC as defined in IETF RFC 4771 [24]. On receipt of a SRTP packet, a terminating UE shall use the contents of the MKI to look up the appropriate SRTP Master Key and salt and generate the appropriate SRTP session key and salt if it satisfies the key derivation rate criteria as specified in IETF RFC3711. If it appears in the SRTP packet, the terminating UE shall use the contents of the SRTP authentication tag to establish the SRTP ROC as defined in IETF RFC 4771 [24].

NOTE 1: Assuming members of the group have been keyed/pre-provisioned at some point in the past, this security mechanism is entirely stateless.

NOTE 2: The receiver does not need to generate an appropriate SRTP session key and salt every time when it receives a SRTP packet. The key derivation rate defined in IETF RFC3711 [13] determines the session key generation frequency. Refer to RFC3711 for more information.

NOTE 3: As the SRTP synchronization source identifier (SSRC) is used for encryption and decryption, the SSRC value in the SRTP packet needs to be maintained from the transmitting UE to the receiving UE. This includes the uplink and the downlink, over unicast or multicast.

A diagram of the SRTP packet format is within figure 7.5.2-2.

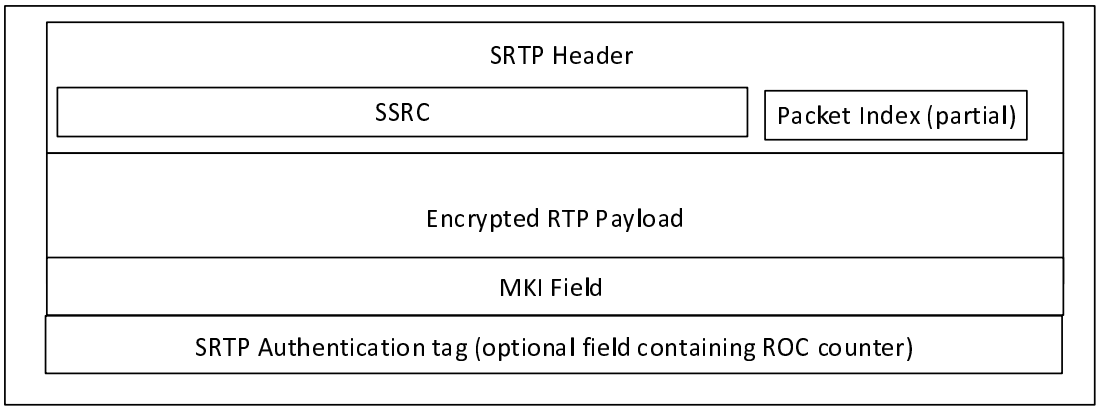


Figure 7.5.2-2: SRTP packet format showing security parameters

The length of the MKI field is defined by the key distribution procedure used to create the original security context.

7.6 Protection of offline floor control signalling (SRTCP)

7.6.1 General

Floor control signalling is sent from the MCPTT UE to a Floor Arbitrator. The Floor Arbitrator will be part of the MCPTT server when MCPTT group communications are online. When MCPTT group communications are offline, the Floor Arbitrator will be an MCPTT client. Floor control signalling is transmitted using MBCP or TBCP, both of which use RTCP, cf. IETF RFC 3550 [12]. The integrity and confidentiality protection for one-to-many communications using RTCP may be achieved by using SRTCP, IETF RFC 3711 [13].

For online communications, the connection between the MCPTT UE and the Floor Arbitrator (MCPTT Server) is protected by using SRTCP which key agreement is specified in clause 9.4. This clause provides a mechanism for protecting offline floor control signalling during the group call or the private call.

When the MCPTT client is operating off network, the key management as well as the Master Key and Master Salt for SRTCP is the same with that for SRTP. As a result of this mechanism, the group members in the group call or MCPTT UEs in the private call share a Master Key, a Master Salt and an MKI for the protection of SRTCP.

The mechanism described in IETF RFC 3711 [13] is used to encrypt the RTCP payload. A diagram of the key derivation mechanism (as described in IETF RFC 3711 [13]) is shown in figure 7.6.1-1.

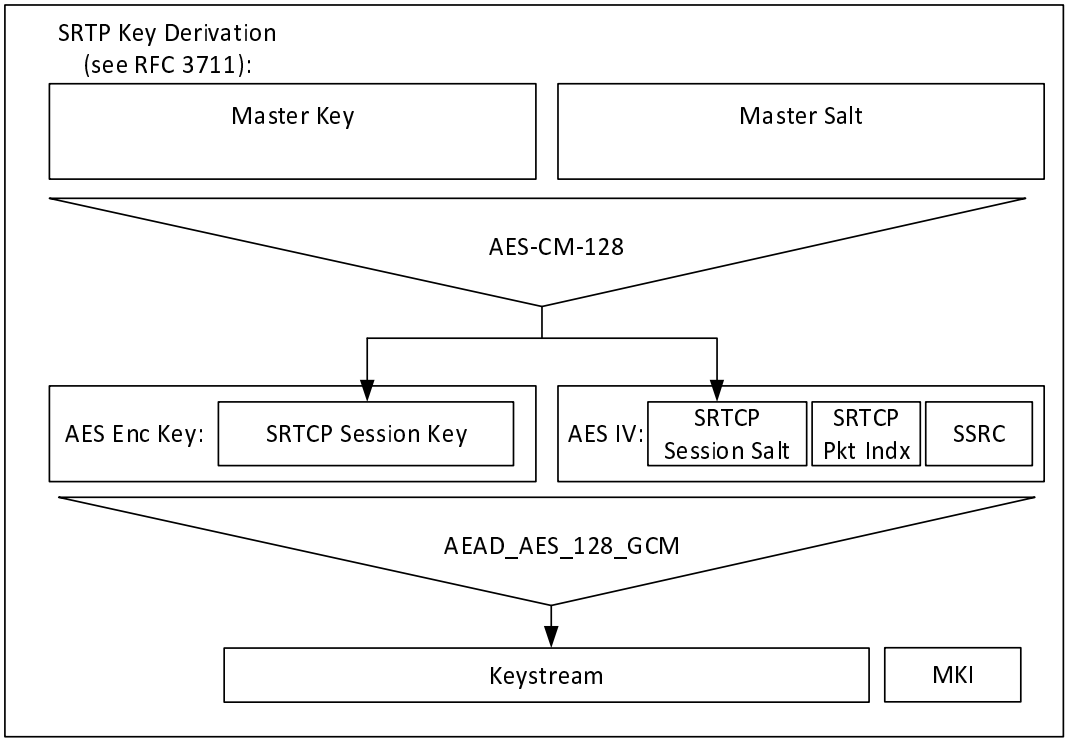


Figure 7.6.1-1: Security mechanism for floor control protection

The AES-CM-128 algorithm as defined in IETF RFC 3711 [13] shall be supported as the SRTCP PRF (which is used to derive the SRTCP session key and salt). A SRTP key derivation rate of 0 shall be used to indicate that session keys and salts shall not be refreshed. The AEAD_AES_128_GCM algorithm as defined in IETF RFC 7714 [26] shall be supported for providing confidentiality and data authentication of SRTCP packets. The AEAD_AES_128_GCM algorithm requires that the SRTCP session key is 16 octets in length and the session salt is 12 octets in length.

7.6.2 Security procedures for offline floor control protection

Floor control protection does not require any signalling mechanism to convey information. The information is provided within each SRTCP Packet.

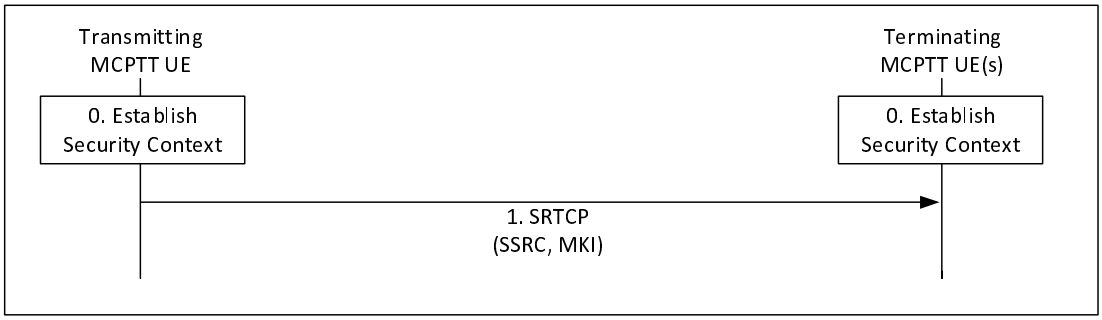


Figure 7.6.2-1: Security procedure for media stream protection

Figure 7.6.2-1 shows the security mechanism.

0) Prior to beginning this procedure the MCPTT UEs involved in the communication shall have established a security context for SRTCP (Master Key, Master Salt, MKI).

1) Transmitting UEs shall send SRTCP packets using the format described in IETF RFC 3711 [13]. The packet shall include a Master Key Identifier (MKI) field which contains the information required to locate the Master Key and Master Salt. On receipt of a SRTCP packet, a terminating UE shall use the contents of the MKI to look

up the appropriate Master Key and Salt and generate the appropriate SRTCP session key and salt if it satisfies the key derivation rate criteria as specified in IETF RFC 3711 [13].

NOTE 1: Assuming members of the group or private call MCPTT UEs have been keyed/pre-provisioned at some point in the past, this security mechanism is entirely stateless.

NOTE 2: The receiver does not need to generate an appropriate SRTCP session key and salt every time when it receives a SRTCP packet. The key derivation rate defined in IETF RFC 3711 [13] determines the session key generation frequency. Refer to RFC3711 for more information.

A diagram of the SRTCP packet format is within figure 7.6.2-2.

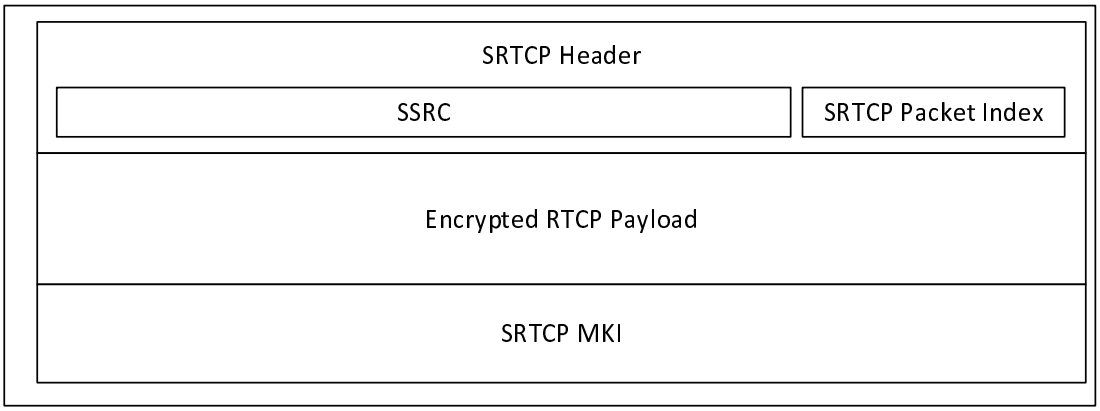


Figure 7.6.2-2: SRTCP packet format showing security parameters

The length of the MKI is determined by the key distribution mechanism.

8 Inter/Intra domain interface security

8.1 General

For all interfaces between network elements within trusted domain or between the trusted domains, namely HTTP-2, HTTP-3, SIP-2 and SIP-3:

- 3GPP TS 33.210 [4] shall be applied to secure signalling messages on the reference points unless specified otherwise; and

- 3GPP TS 33.310 [5] may be applied regarding the use of certificates with the security mechanisms of 3GPP TS 33.210 [4] unless specified otherwise in the present document.

NOTE: For the case of an interface between two network elements in the same trusted domain, 3GPP TS 33.210 [4] does not mandate the protection of the interface by means of IPsec. However, it is up to the domain administrator's policy to also protect interfaces within the same trusted domain.

SEG as specified in 3GPP TS 33.210 [4] may be used in the trusted domain to terminate the IPsec tunnel.

9 Protection of floor control and sensitive application signalling

9.1 Key agreement for protection of floor control and sensitive application data (Client to Server)

9.1.1 Identity-based key management for Client Server Key (CSK)

A Client-Server Key is required to protect floor control signalling between the MCPTT client and the MCPTT Server.

Additionally, the MCPTT Service provider may require that MCPTT related identities and other sensitive information transferred between clients and MCPTT domain on the SIP-1 and SIP-2 interfaces be protected at the application layer from any viewing, including protection from viewing at the SIP signalling layer. Symmetric key based protection of SIP payload using CSK may be used to satisfy this requirement.

Identity based Public Key Cryptography (IDPKC) based on MIKEY-SAKKE IETF RFC 6509 [11] may be used to establish the CSK between two SIP endpoints. Before IDPKC can be used by client to securely share the encryption key, the MCPTT user shall first be authorized by KMS for MCPTT key management services. Once the MCPTT user is authorized, the KMS distributes the user's key material to the client as specified in clause 7.2.3.

MIKEY-SAKKE IETF RFC 6509 [11] shall be used by the client to securely transport the CSK over SIP to all the servers within an MCPTT domain.

The server receives the SIP message with the protected CSK and retrieves it from the message. It associates the MCPTT User's SIP Core identity (IMPU), MCPTT ID and the received CSK. Identity binding is used to uniquely identify the CSK used in protection of the SIP payload in subsequent SIP messages sent by both the client and the servers within an MCPTT domain.

The purpose of the CSK is the following:

- Protection of floor control signalling between the MCPTT client and MCPTT Server.
- Protection of sensitive MCPTT application data in the signalling plane.
- Protection of the Access Token in the signalling plane.

The uses of the CSK are shown in Figure 9.1-1.

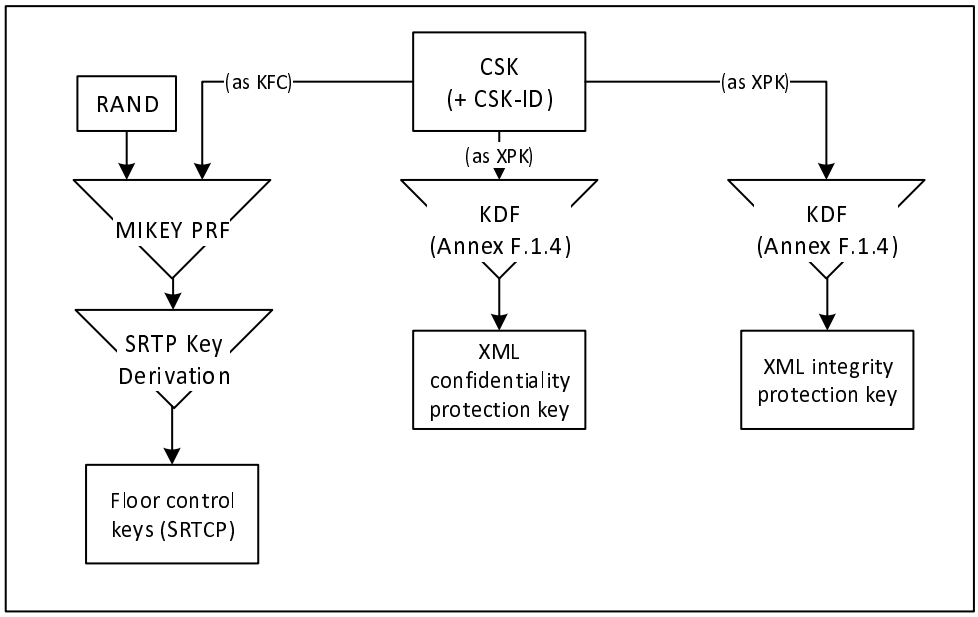


Figure 9.1-1: Uses of the Client-Server Key

9.1.2 Creation of the CSK

The 128-bit CSK is generated by the client and provided encrypted to the server through the SIP interface along with the CSK-ID identifying the CSK.

The key remains in use until: a new CSK is required, the SIP session is torn down, the MCPTT user logs off, or some other indication. If during the active SIP session an update of the CSK is required, the client may establish a new CSK and provide it to the server.

9.1.3 Secure distribution of the CSK

9.1.3.0 General

The CSK is created by the client and distributed to the servers within an MCPTT domain using MIKEY-SAKKE IETF RFC 6509 [11].The shared CSK is distributed in the MIKEY-SAKKE I_MESSAGE, as defined in IETF RFC 6509 [11], which ensure the confidentiality, integrity and authenticity of the payload.

9.1.3.1 MIKEY-SAKKE I_MESSAGE

The key is encrypted to the identity associated to the MCPTT domain (UID), and inserted in the SAKKE payload of the I_MESSAGE. The UID used to encrypt the data will be derived from the MCPTT Domain Security Identifier (MDSI) and a time-related parameter (e.g. the current year and month) as described in clause 7.2. The MDSI is added to the recipient field (IDRr) of the message.

The I_MESSAGE message also contains a signature in the SIGN payload, which is based on the user identity (UID) of the MCPTT User. This identity is derived from the MCPTT ID of the user and a time-related parameter (e.g. the current year and month). The MCPTT ID is added to the initiator field (IDRi) of the message. Where the MCPTT ID is sensitive, the identity mechanism defined in clause E.7 is used to define the initiator field of the message.

Clause E.4 provides MIKEY message structure for CSK distribution.

The resulting MIKEY-SAKKE message is sent over SIP (for example, during MCPTT User authorization).

9.1.3.2 Distribution of CSK during MCPTT Service Authorization and group subscription

The client shall include the MIKEY message, containing the CSK, in the SIP message that is used to perform the MCPTT user authorization procedure.

An illustration is provided below as an example of how this message in included in the body of the SIP REGISTER message.

EXAMPLE:

```
REGISTER sip:MCPTT_Server_PSI SIP/2.0
Via: SIP/2.0/UDP den3.level3.com
Max-Forwards:70
From: MCPTT client IMPU
To:
Call-ID: <>
CSeq:  1 REGISTER
Contact: <URI>
Content-Type: multipart/mixed;boundary="boundary1"
Content-Length: 619

--boundary1
Content-Type: application/mikey
MIKEY I_MESSAGE
--boundary1
Content-Type: application/...
Encrypted Access token, MCPTT ID
--boundary1—
```

The MIME media type "application/mikey" IETF RFC 3830 [22], is used in this example to insert MIKEY I_MESSAGE in the SIP payload.

9.1.3.3 Obtaining CSK from the I_MESSAGE

The server performs the following steps when it receives the SIP message with the MIKEY I_MESSAGE containing the encrypted CSK:

1. Where the MCPTT ID is sensitive and is encrypted with CSK key,use the client's UID obtained from the IDRi field in the message to compute the signature and verify it with the value in the SIGN payload of the MIKEY message.

2. The SAKKE based encryption scheme defined in IETF RFC 6509 [11], is used to extract the CSK from the SAKKE payload of the MIKEY message.

3. Compute client's UID based on the MCPTT ID decrypted using the CSK key, and compare the UID with the UID obtained from the IDRi field of the MIKEY message.

9.1.3.4 Procedure

The following steps describe how the client obtains the user specific key material and securely transfers the CSK to a server within the MCPTT domain.

Prior to beginning of this procedure, the client would have obtained user-specific key material from the KMS.

1) The client randomly generates the CSK. The client computes MCPTT Domain UID from the MCPTT Domain Security Identifier (MDSI) and uses it to encrypt the CSK based on SAKKE based encryption IETF RFC 6509 [11].

2) The client generates MIKEY-SAKKE I_MESSAGE. The message shall encapsulate the encrypted CSK and shall be signed using the key associated with the MCPTT User's UID generated from the MCPTT ID.

3) The client includes the I_MESSAGE in the SIP payload and sends the SIP message addressed to the PSI of the server.

The following steps describe how the server retrieves the encryption key from the SIP message:

1) The server receives the SIP message with the I_MESSAGE embedded in the SIP payload.

2) The server checks the signature on the I_MESSAGE message using client's UID.

3) If the signature is valid, the server extracts and decrypts the encapsulated CSK using the key associated with the MCPTT Domain"s UID generated from the MCPTT Domain Security Identifier (MDSI).

4) Once the CSK has been extracted, MCPTT specific information including access token protected in the SIP message as defined in clause 9.3.4, may be decrypted.

Figure 9.1.3-.4-1 shows the functional diagram for the client and a server within the MCPTT domain. The server is shown in this example.

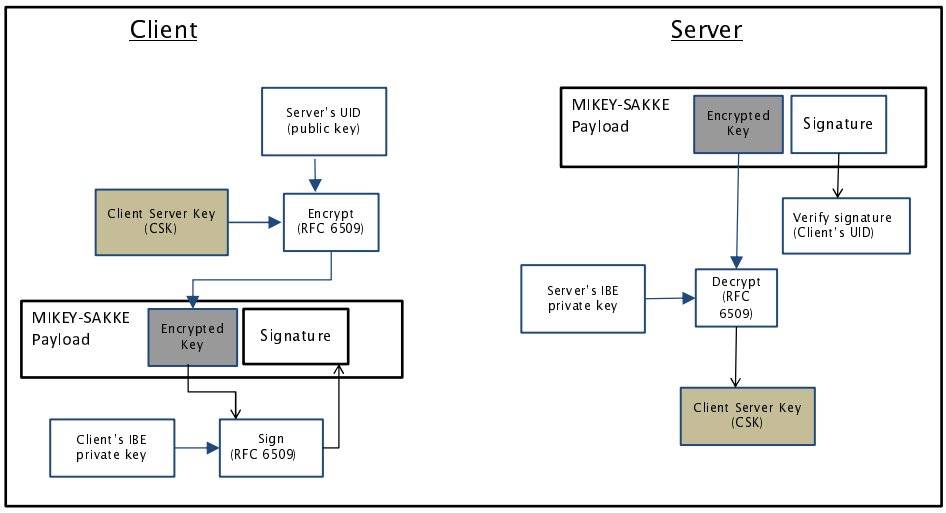


Figure 9.1.3.4-1: Functional diagram for Identity based distribution of CSK

9.2 Key agreement for protection of floor control and sensitive application data between servers

Floor control between MCPTT servers may need to be protected. Additionally, certain values and identifiers transferred in the signalling plane between servers within an MCPTT domain, or between MCPTT domains may be treated as sensitive by public safety users.

To protect information from all other entities outside of the MCPTT domain(s), a shared 128-bit Signalling Protection Key (SPK) needs to be established between the servers. The SPK is provided along with a 32-bit identifier, the SPK-ID and 128-bit random value SPK-RAND. The most significant four bits of the identifier (the Purpose Tag) of the SPK-ID shall be '3' to denote the purpose of the SPK is for signalling protection, as described in clause 7.3.3.

The SPK and associated values shall be directly provisioned into the communicating servers, along with the SPK-ID. With the SPK provisioned, floor control and XML content within the SIP may be protected as defined in clause 9.3.

NOTE: The XML protection mechanism specified in this clause is for public-safety use only.

The uses of the SPK for inter-server protection are shown in Figure 9.2-1.

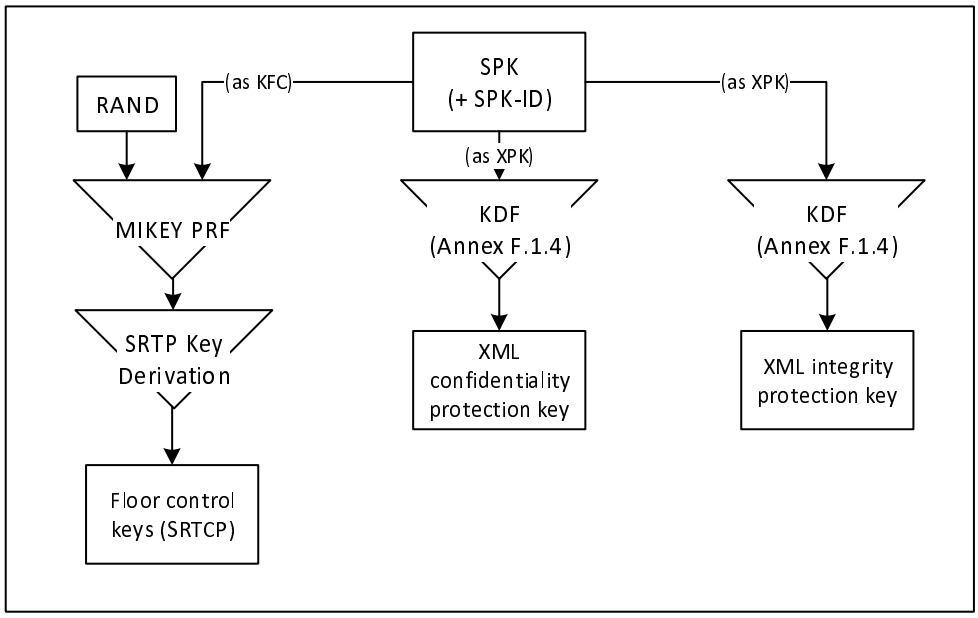


Figure 9.2-1: Uses of the Signalling Protection Key

9.3 Protection of XML content

9.3.1 General

Certain values, keys and identifiers transferred in XML between a server in the MCPTT domain and client may be treated as sensitive by public safety users. To protect these values from all other entities outside of the MCPTT Domain, this clause defines an optional mechanism to provide confidentiality protection on these values using XML encryption. Additionally, as some public safety users may require integrity protection on transmitted content, this clause defines an optional mechanism to provide integrity protection using XML signatures.

NOTE 1: The protection mechanism specified in this clause is for public-safety use only.

NOTE 2: The introduction of XML security mechanisms increases the size of the XML document. Consideration should be given to the impact of this size increase.

Editor's Note: It needs to be confirmed that the virtual proxy techniques being studied in SA3-LI (LIV8 S8HR study) can be extended to control use of MCPTT encryption in VPLMN roaming scenarios.

9.3.2 Protected content

Confidentiality protection may only be applied to the following identifiers and values:

- MCPTT ID.
- MCPTT Group ID.
- User location information.
- Alerts.
- Access token.
- KMS provisioned key material.

Integrity protection may be applied to the entire XML document, and to individual KMS certificates.

9.3.3 Key agreement

The protection mechanisms defined rely on a shared XML Protection Key (XPK) to be able to encrypt and sign XML.

For connections between the client and the MCPTT Domain the XPK shall be the 128-bit shared Client-Server Key (CSK) established as defined in clause 9.1. The XPK-ID shall be the CSK-ID.

For connections between servers inside and across MCPTT Domains the XPK shall be the128-bit manually provisioned SIP Protection Key (SPK) established as defined in clause 9.2. The XPK-ID shall be the SPK-ID

For connections between the KMS and the MCPTT client, the XPK shall be the 256-bit manually provisioned TrK, described in clause 7.2.3. The XPK-ID shall be the TrK-ID.

The integrity key and confidentiality key for application data protection shall be derived from the XPK as defined in clause F.1.4. The XPK-ID may be listed in the XML to aid decryption.

9.3.4 Confidentiality protection using XML encryption (xmlenc)

This clause defines an optional mechanism to allow specific XML content to be encrypted between the client and the server. XML content is encrypted as defined by XML Encryption Syntax, Version 1.1 [27].

NOTE: Only encryption of XML simple content is supported. Encryption of XML tags, attributes and elements is not supported.

To encrypt content within a specific XML element, the content shall be replaced with the <EncryptedData> element. The <EncryptedData> element shall contain a <CipherData> element, containing a <CipherValue> element containing the encrypted content. Encryption shall be performed as defined in [27] using the CSK as the cipher key.

Where protecting content, the <EncryptedData> element may:

- Use the 'Type' attribute specifying that content is encrypted ('http://www.w3.org/2001/04/xmlenc#Content').
- Contain <KeyData><KeyInfo> element containing the base64 encoded XPK-ID.
- Contain <EncryptionMethod> element listing the encryption algorithm used for encrypting the XML content. The AES-128-GCM algorithm shall be supported, as identified by the algorithm identifier: 'http://www.w3.org/2009/xmlenc11#aes128-gcm'.

Where protecting key material, the <EncryptedData> element may:

- Use the 'Type' attribute specifying that content is encrypted ('http://www.w3.org/2001/04/xmlenc# EncryptedKey').
- Contain <KeyData><KeyInfo> element containing the base64 encoded XPK-ID.
- Contain <EncryptionMethod> element listing the encryption algorithm used for encrypting the XML key material. The AES-256 key wrap algorithm as defined in RFC 3394 [34] shall be supported, as identified by the algorithm identifier 'http://www.w3.org/2001/04/xmlenc#kw-aes256'.

Where these elements do not occur, the information they contain shall be known to both the client and server in the MCPTT domain through other means.

The following is an example of unprotected XML content:

EXAMPLE:

```
<ExampleTag xsd:type="Normal">
  sensitive.data@example.org
</ExampleTag>
```

When XML encryption is applied, the following is an example of the encrypted content:

EXAMPLE:

```
<ExampleTag xsd:type="Encrypted">
  <EncryptedData xmlns='http://www.w3.org/2001/04/xmlenc#'
   Type='http://www.w3.org/2001/04/xmlenc#Content'>
    <EncryptionMethod Algorithm="http://www.w3.org/2009/xmlenc11#aes128-gcm"/>
    <ds:KeyInfo>
      <ds:KeyName>base64XpkId</KeyName>
    </ds:KeyInfo>
    <CipherData>
      <CipherValue>A23B45C56</CipherValue>
    </CipherData>
  </EncryptedData>
</ExampleTag>
```

9.3.5 Integrity protection using XML signature (xmlsig)

Where integrity protection is required, an XML HMAC signature may be applied using the XPK to key the HMAC. The XML Signature syntax is defined by W3C in [28]. This specifies how signatures are created and applied to XML. The XML shall contain a <Signature> element which signs the elements requiring integrity protection. Within this element a <SignatureValue> element shall contain the HMAC signature of the content.

The <Signature> element may contain:

- <CanonicalizationMethod> element listing an appropriate algorithm.
- <SignatureMethod> element listing an appropriate algorithm. HMAC-SHA256 shall be supported for signatures.
- <Reference> element containing a URI identifying the content to be signed and the method for hashing the content. SHA-256 shall be supported for hashing content.
- <KeyInfo><KeyName> element containing the base64 encoded XPK-ID.

Where these elements do not occur, the information they contain shall be known to both the client and server in the MCPTT domain through other means.

The following provides an example of an XML signature added to a generic XML document.

EXAMPLE:

```
<SignedXMLDoc Id="xmldoc">
<XMLDoc>
…
</XMLDoc>
<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
    <SignedInfo>
      <CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
      <SignatureMethod Algorithm="http://www.w3.org/2001/04/xmldsig-more#hmac-sha256">
        <HMACOutputLength>128</HMACOutputLength>
      </SignatureMethod>
      <Reference URI="#xmldoc">
        <DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
        <DigestValue>nnnn</DigestValue>
      </Reference>
    </SignedInfo>
    <SignatureValue>DEADBEEF</SignatureValue>
    <KeyInfo>
      <KeyName>base64XpkId</KeyName>
    </KeyInfo>
  </Signature>
</SignedXMLDoc>
```

9.4 Key agreement for online floor control (SRTCP)

9.4.1 General

Floor control encryption is required between the MCPTT UE and MCPTT Server and between a pair of MCPTT Servers. Floor control security is protected hop-by-hop, meaning that floor control traffic is always decrypted by the floor control server within the MCPTT server and then re-encrypted to its destination.

The purpose of key agreement is to establish a Key for Floor Control (KFC) from which the master key for the SRTCP protocol can be derived. A 32-bit identifier for the key (KFC-ID) and a 128-bit random value (KFC-RAND) is also established.

9.4.2 Key agreement between MCPTT client and MCPTT Server

In Clause 9.1, a Client-Server Key is generated and shared between the MCPTT client and MCPTT Server, along with the identifier (CSK-ID). For floor control, the KFC shall be the CSK and the KFC-ID shall be the CSK-ID. KFC-RAND shall be the MIKEY RAND value transmitted in the MIKEY message used to distribute the CSK.

9.4.3 Key agreement between MCPTT Servers

In Clause 9.2, a Signalling Protection Key (SPK) is shared between MCPTT Servers along with a SPK-ID. For floor control signalling transferred between MCPTT Servers, the KFC shall be the SPK, the KFC-ID shall be the SPK-ID and the KFC-RAND shall be the SPK-RAND.

9.4.4 Key agreement for multicast from MCPTT Server

In clause 7.3.2, a Multicast Key for Floor Control (MKFC) is generated and shared from the GMS to the MCPTT Server and MCPTT client, along with an identifier (MKFC-ID). For the protection of multicast floor control, the KFC shall be the MKFC and the KFC-ID shall be the MKFC-ID. KFC-RAND shall be the MIKEY RAND value transmitted in the MIKEY message used to distribute the MKFC.

9.4.5 Derivation of SRTCP key material

As a result of the key agreement process, the entities (MCPTT client and server, or MCPTT servers) shall share a KFC, a KFC-ID and a KFC-RAND. The KFC shall be used as the MIKEY Traffic Generating Key (TGK), the KFC-ID shall be used as the MIKEY CSB ID and the KFC-RAND shall be used as the MIKEY RAND value. These shall be used to generate the SRTCP Master Key and SRTCP Master Salt as specified in IETF RFC 3830 [22]. The key derivation function defined in section 4.1.4 of IETF RFC 3830 [22] using the PRF-HMAC-SHA-256 Pseudo-Random Function as described in IETF RFC 6043 [25], section 6.1 shall be supported for generating the SRTCP Master Key and Salt. SRTCP session keys are generated from the SRTCP Master Key and Salt as defined in Clause 7.6.

NOTE: Within RFC 3830 [22], the SRTCP Master Key and SRTCP Master Salt are referred to as the SRTP Master Key and the SRTP Master salt respectively.

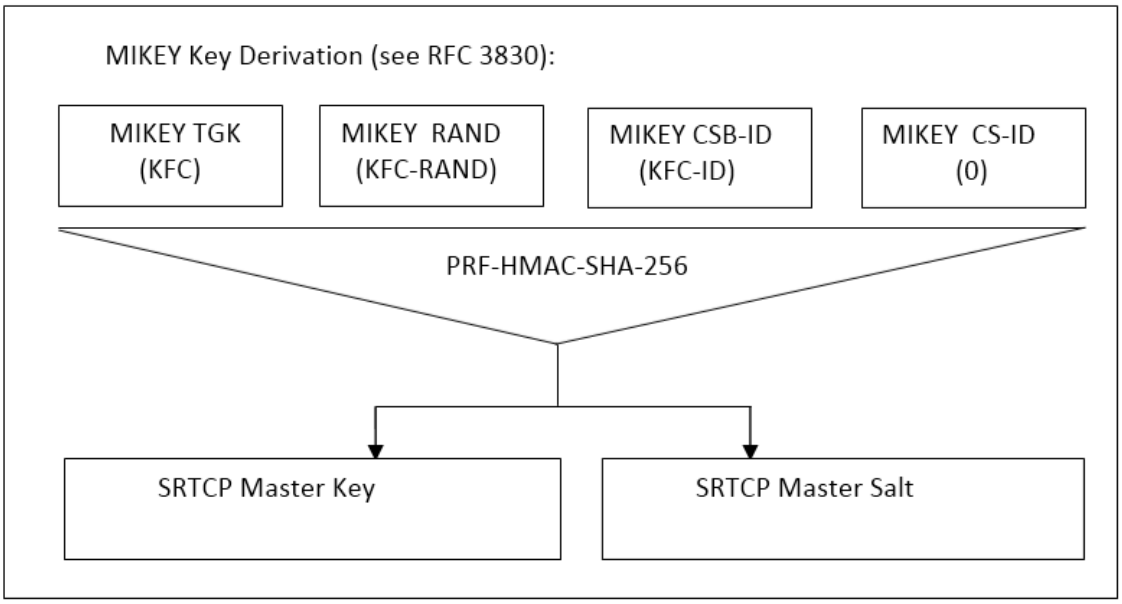


Figure 9.4.5-1: Key derivation for on-network floor control protection

To identify the security context from the media stream a SRTCP Master Key Identifier (MKI) is required. The MKI shall be the 32-bit KFC-ID.

Annex A (normative): Security requirements

A.0 Introduction

Stage 1 requirements pertaining to MCPTT security are found in 3GPP TS 22.179 [3], Stage 2 Architectural requirements pertaining to MCPTT security are found in 3GPP TS 23.179 [2]. The following are MCPTT derived security requirements:

A.1 Configuration & service access

[33.179 MCPTT-A.1-001] The MCPTT UE and the network entity providing the MCPTT configuration data, shall mutually authenticate each other prior to MCPTT UE configuration to use the MCPTT service.

[33.179 MCPTT-A.1-002] The MCPTT User and the MCPTT Service shall mutually authenticate each other prior to providing the MCPTT UE with the MCPTT User profile and access to user-specific services.

[33.179 MCPTT-A.1-003] The transmission of configuration data and user profile data between an authorized MCPTT server in the network and the MCPTT UE shall be confidentiality protected, integrity protected and protected from replays.

A.2 Group key management

[33.179 MCPTT-A.2-001] Group key material shall be integrity and confidentiality protected for a specific MCPTT User during distribution from the MCPTT service to MCPTT UEs.

[33.179 MCPTT-A.2-002] Group key material shall be authenticated as coming from a valid, authorized source. The authorized source may be an MCPTT Administrator or may be another authorized entity (e.g. an authorized MCPTT User or Dispatcher).

[33.179 MCPTT-A.2-003] It shall be possible for authorized entities to dynamically create and distribute a new group security context at any time. This may be as part of a group creation process, be due to a periodic update to maintain key freshness, or due to compromise of group key material.

[33.179 MCPTT-A.2-004] The creation of a new group security context (e.g. via User-Regroup operation) shall not change or compromise an existing group security context.

[33.179 MCPTT-A.2-005] It shall be possible for an authorized, authenticated entity to revoke and update a group security context from use.

A.3 On-network operation

[33.179 MCPTT-A.3-001] All users of the MCPTT service shall be authenticated to prevent an adversary impersonating a user for the purpose of denial of service.

[33.179 MCPTT-A.3-002] The MCPTT service should take measures to detect and mitigate DoS attacks to minimize the impact on the network and on MCPTT users.

[33.179 MCPTT-A.3-003] The MCPTT user shall be authenticated by the MCPTT application.authorized

[33.179 MCPTT-A.3-004] A mechanism shall exist that allows the MCPTT application to be authenticated by the MCPTT user.

[33.179 MCPTT-A.3-005] The MCPTT UE and MCPTT service should enforce the result of the authentication for the duration of communications (e.g. by integrity protection or implicit authentication by encryption with a key that is derived from the authentication and is unknown to the adversary).

[33.179 MCPTT-A.3-006] The security solution should minimize the impact of a compromised MCPTT UE on other MCPTT UEs.

[33.179 MCPTT-A.3-007] The MCPTT Service shall provide a means to ensure integrity of all MCPTT user signalling at the application layer.

[33.179 MCPTT-A.3-008] The MCPTT Service shall protect the administrative and security management parameters from manipulation by individuals who are not explicitly authorized by the Mission Critical Organization.

[33.179 MCPTT-A.3-009] The MCPTT service shall provide a means to support confidentiality of MCPTT user identities from all entities outside the MCPTT service.

[33.179 MCPTT-A.3-010] The MCPTT service shall provide a means to support confidentiality of MCPTT signalling from all entities outside the MCPTT service.

[33.179 MCPTT-A.3-011] The MCPTT Service shall provide a means to support end-to-end confidentiality and integrity protection for all media traffic transmitted between MCPTT UEs.

[33.179 MCPTT-A.3-012] The MCPTT Service shall provide a means to support the confidentiality and integrity protection of location information transmitted from the MCPTT UE to the MCPTT application server.

A.4 Ambient listening

[33.179 MCPTT-A.4-001] Specific roles in the organization and shall be identified to authorize and activate Ambient Listening and privileges shall be assigned to these roles to activate and register the use of ambient listening.

[33.179 MCPTT-A.4-002] The activation of the Ambient Listening functionality shall be automatically registered by the system and will be stored as an 'event' by the system.

[33.179 MCPTT-A.4-003] Any decision to activate Ambient Listening, or review of such a decision, may also be recorded in a suitable incident log unless to do so would interfere with the purpose for which the functionality is being used i.e. an investigation tool for evidence gathering in cases of suspected gross misconduct of staff or evidence gathering in criminal cases. If this is the case the authorization needs to be recorded elsewhere as appropriate.

[33.179 MCPTT-A.4-004] A radio user should be told as soon as possible that they are, or have been, subject to Ambient Listening and the reason why the functionality was activated. The fact they have been informed, by whom and when, should be recorded in a suitable log.

A.5 Data communication between MCPTT network entities

[33.179 MCPTT-A.5-001] A security mechanism shall exist that allows transmission of data between 3GPP MCPTT network entities to be authenticated, confidentiality protected, integrity protected and protected from replays.

NOTE: UE-to-UE and UE-to-network relays are not considered to be 'network entities'.

A.6 Key stream re-use

[33.179 MCPTT-A.6-001] The MCPTT system shall ensure that key streams are not reused.

A.7 Late entry to group communication

[33.179 MCPTT-A.7-001] An authorized MCPTT User shall be able to obtain the information necessary to derive the group security context for the MCPTT Group while an MCPTT Group communication is on-going. As a result, the MCPTT User shall be able to listen to the group communication within 350ms. This requirement applies for both on-network and off-network MCPTT operation.

A.8 Private call confidentiality

[33.179 MCPTT-A.8-001] It shall be possible to establish a unique Private Call security context between any pair of authorized MCPTT users within the MCPTT system. The security context shall not be available to other MCPTT users, except, where necessary, authorized MCPTT monitoring functions (e.g. LI, Discreet Listening). If the security context is made available to monitoring functions, appropriate controls and logging shall exist. This requirement applies when MCPTT UEs are operating both on-network and off-network.

[33.179 MCPTT-A.8-002] The Private Call security context shall provide a means to provide confidentiality and integrity protection of user traffic, and authenticate the MCPTT users involved in the Private Call.

A.9 Off-network operation

[33.179 MCPTT-A.9-001] The MCPTT service should take measures to detect and mitigate DoS attacks to minimize the impact to relays and to off-network MCPTT users.

[33.179 MCPTT-A.9-002] The MCPTT Service shall provide a means to support end-to-end security for all media traffic transmitted between MCPTT UEs, including where relays are used.

[33.179 MCPTT-A.9-003] The MCPTT Service shall provide a means to support the confidentiality and integrity protection of location information transmitted from the MCPTT UE to the MCPTT application server, including where relays are used.

[33.179 MCPTT-A.9-004] MCPTT off-network UEs shall be explicitly or implicitly authenticated to each other.

[33.179 MCPTT-A.9-005] MCPTT off-network UEs and MCPTT relays shall be explicitly or implicitly authenticated to each other.

[33.179 MCPTT-A.9-006] The security solution should minimize the impact of a compromised MCPTT UE on other MCPTT UEs.

[33.179 MCPTT-A.9-007] The MCPTT Service shall provide a means to ensure integrity of all MCPTT user signalling at the application layer.

[33.179 MCPTT-A.9-008] The MCPTT service shall provide a means to support confidentiality of MCPTT user identities from all entities outside the MCPTT service.

[33.179 MCPTT-A.9-009] The MCPTT service shall provide a means to support confidentiality of MCPTT signalling from all entities outside the MCPTT service.

A.10 Privacy of MCPTT identities

[33.179 MCPTT-A.10-001] The MCPTT identities of each plane shall be used within the corresponding plane and concealed to other planes.

[33.179 MCPTT-A.10-002] When required by the MCPTT Service provider, MCPTT application services layer identities (such as the Mission Critical user identity, MCPTT-ID and MCPTT Group ID) and other application services sensitive information (as further described in 3GPP TS 23.179 [2], clause 8.1), shall be contained within the application plane and shall provide a means to support confidentiality and integrity of the application plane from the SIP signaling plane.

[33.179 MCPTT-A.10-003] When protection of identities and other sensitive MCPTT application information is NOT required by the MCPTT Service provider, the MCPTT application services layer identities (such as the Mission Critical user identity, MCPTT-ID and MCPTT Group ID) and other application services sensitive information (as further described in 3GPP TS 23.179 [2], clause 8.1), shall remain contained within the application plane but do not require confidentiality protection.

Editor's note: It is FFS whether integrity protection is needed when confidentiality protection is not in use.

A.11 User authentication and authorization requirements

The solution for user authentication and authorization described in the present document shall satisfy the following requirements:

Interoperability between different networks and different manufacturers' clients and servers:

- Satisfy requirements for MCPTT roaming and migration.

Flexibility in deployment models (see 3GPP TS 23.179 [2]):

- Support all deployment models listed in 3GPP TS 23.179 [2].

Support for interchangeable MCPTT user authentication solutions:

- Allow implementations to use different means to authenticate the user, e.g. Web SSO, SIP digest, GBA, biometric identifiers, username+password.

Scalability (number of users):

- Provide efficient support for small MCPTT systems with few users, to large MCPTT systems with hundreds of thousands of users.

Extensibility:

- Be extensible to provide authorization for further mission critical services including group aware services, additional interfaces, etc.

Annex B (normative): OpenID connect profile for MCPTT

B.0 General

The information in this annex provides a normative description of the MCPTT Connect Authentication and Authorization framework based on the OpenID Connect 1.0 standard. Characterization of the id token, access token, how to obtain tokens, how to validate tokens, and how to use the refresh token is explained.

The OpenID Connect 1.0 standard provides the source of the information contained in this annex. MCPTT Connect profiles the OpenID Connect standard and includes the MCPTT ID in the id token and the access token, as well as the definition of MCPTT specific scopes for key management, MCPTT service, configuration management, and group management. This profile is completely standard compliant with OpenID Connect.

B.1 MCPTT tokens

B.1.1 ID token

B.1.1.0 General

The ID Token shall be a JSON Web Token (JWT) and contain the following standard and MCPTT token claims. Token claims provide information pertaining to the authentication of the MCPTT user by the IdM server as well as additional claims. This clause profiles the required standard and MCPTT claims for the MCPTT Connect profile.

B.1.1.1 Standard claims

These standard claims are defined by the OpenID Connect 1.0 specification and are REQUIRED for MCPTT. Other claims defined by OpenID Connect are optional. The standards-based claims for an MCPTT id token are shown in table B.1.1.1-1.

Table B.1.1.1-1: ID token standard claims

Parameter	Description
iss	REQUIRED. The URL of the IdM server.
sub	REQUIRED. A case-sensitive, never reassigned string (not to exceed 255 bytes), which uniquely identifies the MCPTT user within the MCPTT server provider's domain.
aud	REQUIRED. The OAuth 2.0 client_id of the MCPTT client
exp	REQUIRED. Implementers MAY provide for some small leeway, usually no more than a few minutes, to account for clock skew (not to exceed 30 seconds)
iat	REQUIRED. Time at which the ID Token was issued. Its value is a JSON number representing the number of seconds from 1970-01-01T0:0:0Z as measured in UTC until the date/time.

B.1.1.2 MCPTT claims

The MCPTT Connect profile extends the OpenID Connect standard claims with the additional claims shown in table B.1.1.2-1.

Table B.1.1.2-1: ID token MCPTT claims

Parameter	Description
mcptt_id	REQUIRED. The MCPTT ID of the current MCPTT user of the MCPTT client.

B.1.2 Access token

B.1.2.0 Introduction

The access token is opaque to MCPTT clients and is consumed by the MCPTT resource servers (i.e. KMS, MCPTT server, etc). The access token shall be encoded as a JSON Web Token as defined in IETF RFC 7519 [32]. The access token shall include the JSON web digital signature profile as defined in IETF RFC 7515 [35].

B.1.2.1 Standard claims

MCPTT access tokens shall convey the following standards-based claims as defined in IETF RFC 7662 [33].

Table B.1.2.1-1: Access token standard claims

Parameter	Description
exp	REQUIRED. Implementers MAY provide for some small leeway, usually no more than a few minutes, to account for clock skew (not to exceed 30 seconds).
scope	REQUIRED. A JSON string containing a space-separated list of the MCPTT authorization scopes associated with this token.
client_id	REQUIRED. The identifier of the MCPTT client making the API request as previously registered with the IdM server.

B.1.2.2 MCPTT claims

The MCPTT Connect profile extends the standard claims defined in IETF RFC 7662 [33] with the additional claims shown in table B.1.2.2-1.

Table B.1.2.2-1: Access token MCPTT claims

Parameter	Description
mcptt_id	REQUIRED. The MCPTT ID of the current MCPTT user of the MCPTT client.

B.2 MCPTT client registration

Before an MCPTT client can obtain ID tokens and access tokens (required to access MCPTT resource servers) it shall first be registered with the IdM server of the service provider as required by OpenID Connect 1.0. The method by which this is done is not specified by this profile. For native MCPTT clients, the following information shall be registered:

- The client is issued a client identifier. The client identifier represents the client's registration with the authorization server, and enables the IdM server to reference parameters associated with that client's registration when being requested for an access token by the MCPTT client.

- Registration of the client's redirect URIs.

Other information about the MCPTT client such as (for example): application name, website, description, logo image, legal terms to be consented to, may optionally be registered.

B.3 Obtaining tokens

B.3.0 General

Once an MCPTT client has been successfully registered with the IdM server of the MCPTT service provider, the MCPTT client may request ID tokens and access tokens (as required to access MCPTT resource servers such as PTT and KMS). MCPTT Connect will support a number of different MCPTT client types, including: native, web-based, and browser-based. Only native MCPTT clients are defined in this first version of the MCPTT Connect profile. The exact method in which an MCPTT client requests the access token depends upon the client profile. The MCPTT client profiles, along with steps required from them to obtain OAuth access tokens, are explained in technical detail below.

B.3.1 Native MCPTT client

B.3.1.0 General

This conforms to the Native Application profile of OAuth 2.0 as per IETF RFC 6749 [19].

MCPTT clients fitting the Native application profile utilize the authorization code grant type with the PKCE extension for enhanced security as shown in figure B.3.1.0-1.

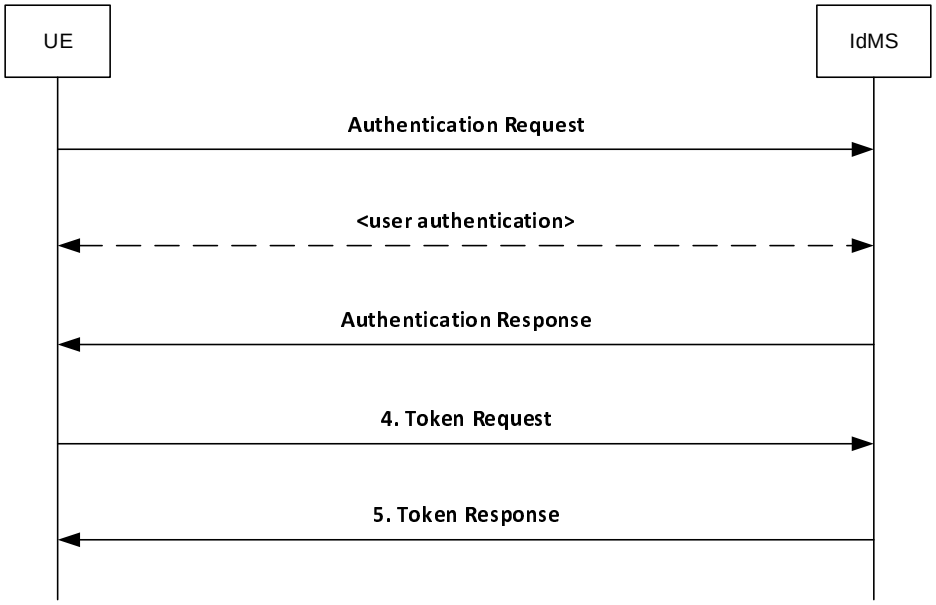


Figure B.3.1.0-1: Authorization Code flow

B.3.1.1 Authentication Request

As described in OpenID Connect 1.0, the MCPTT client constructs a request URI by adding the following parameters to the query component of the authorization endpoint's URI using the "application/x-www-form-urlencoded" format, redirecting the user's web browser to the authorization endpoint of the IdM server. The standard parameters shown in table B.3.1.1-1 are required by the MCPTT Connect profile. Other parameters defined by the OpenID Connect specification are optional.

Table B.3.1.1-1: Authentication Request standard required parameters

Parameter	Values
response_type	REQUIRED. For native MCPTT clients the value shall be set to "code".
client_id	REQUIRED. The identifier of the MCPTT client making the API request. It shall match the value that was previously registered with the IdM server of the MCPTT service provider.
scope	REQUIRED. Scope values are expressed as a list of space-delimited, case-sensitive strings which indicate which MCPTT resource servers the client is requesting access to (e.g. MCPTT, KMS, etc.). If authorized, the requested scope values will be bound to the access token returned to the client. The scope value "openid" is defined by the OpenID Connect standard and is mandatory, to indicate that the request is an OpenID Connect request, and that an ID token should be returned to the MCPTT client. This profile further defines the following additional authorization scopes: - "3gpp:mcptt:ptt_server" (service authorization) - "3gpp:mcptt:key_management_server" (key management authorization) - "3gpp:mcptt:config_management_server" (config mgmt authorization) - "3gpp:mcptt:group_management_server" (group mgmt authorization) Others may be added in the future as new MCPTT resource servers are introduced by 3GPP (see note).
redirect_uri	REQUIRED. The URI of the MCPTT client to which the IdM server will redirect the MCPTT client's user agent in order to return the authorization code to the MCPTT client. The URI shall match the redirect URI registered with the IdM server during the client registration phase.
state	REQUIRED. An opaque value used by the MCPTT client to maintain state between the authorization request and authorization response. The IdM server includes this value in its authorization response back to the MCPTT client.
acr_values	REQUIRED. Space-separated string that specifies the acr values that the IdM server is being requested to use for processing this authorization request, with the values appearing in order of preference. For minimum interoperability requirements, a password-based ACR value is mandatory to support. "3gpp:acr:password".
code_challenge	REQUIRED. The base64url-encoded SHA-256 challenge derived from the code verifier that is sent in the authorization request, to be verified against later.
code_challenge_method	REQUIRED. The hash method used to transform the code verifier to produce the code challenge. This profile current requires the usage of "S256"
NOTE: The order in which they are expressed does not matter.	

An example of an authentication request for MCPTT Connect might look like:

EXAMPLE:

```
GET/as/authorization.oauth2?response_type=code&client_id=mcptt_client&scope=openid 3gpp:mcptt:ptt_se
rver&redirect_uri=
http://3gpp.mcptt/cb&state=abc123&acr_values=3gpp:acr:password&code_challange=0x123456789abcdef&code
_challenge_method=S256
HTTP/1.1
Host: IdMS.server.com:9031
Cache-Control: no-cache
Content-Type: application/x-www-form-urlencoded
```

Upon receiving the authentication request from the MCPTT client, the IdM server performs user authentication. Note that user authentication is completely opaque to the MCPTT client (which never sees any of it, as it is run directly between the IdM server and the user-agent on the UE).

B.3.1.2 Authentication response

The authorization endpoint running on the IdM server issues an authorization code and delivers it to the MCPTT client. The authorization code is used by the MCPTT client to obtain an ID token, access token and refresh token from the IdM server. The authorization code is added to the query component of the redirection URI using the "application/x-www-form-urlencoded" format. The authorization code standard parameters are shown in table B.3.1.2-1.

Table B.3.1.2-1: Authentication Response standard required parameters

Parameter	Values
code	REQUIRED. The authorization code generated by the authorization endpoint and returned to the MCPTT client via the authorization response.
state	REQUIRED. The value shall match the exact value used in the authorization request. If the state does not match exactly, then the NGMI API client is under a Cross-site request forgery attack and shallreject the authorization code by ignoring it and shall not attempt to exchange it for an access token. No error is returned.

An example of an authentication response for MCPTT Connect might look like.

EXAMPLE:

```
HTTP/1.1 302 Found
Location: http://mcptt_client/cb?
code=SplxlOBeZQQYbYS6WxSbIA
  &state=abc123
```

B.3.1.3 Token request

In order to exchange the authorization code for an ID token, access token and refresh token, the MCPTT client makes a request to the authorization server's token endpoint by sending the following parameters using the "application/x-www-form-urlencoded" format, with a character encoding of UTF-8 in the HTTP request entity-body. Note that client authentication is REQUIRED for native applications (using PKCE) in order to exchange the authorization code for an access token. Assuming that client secrets are used, the client secret is sent in the HTTP Authorization Header. The token request standard parameters are shown in table B.3.1.3-1.

Table B.3.1.3-1: Token Request standard required parameters

Parameter	Values
grant_type	REQUIRED. The value shall be set to "authorization_code".
code	REQUIRED. The authorization code previously received from the IdM server as a result of the authorization request and subsequent successful authentication of the MCPTT user.
client_id	REQUIRED. The identifier of the client making the API request. It shall match the value that was previously registered with the OAuth Provider during the client registration phase of deployment, or as obtained by the Motorola Solutions development portal.
redirect_uri	REQUIRED. The value shall be identical to the "redirect_uri" parameter included in the authorization request.
code_verifier	REQUIRED. A cryptographically random string that is used to correlate the authorization request to the token request.

An example of a token request for MCPTT Connect might look like.

EXAMPLE:

```
POST /as/token.oauth2 HTTP/1.1
Host: IdM.server.com:9031
Cache-Control: no-cache
Content-Type: application/x-www-form-urlencoded

grant_type=authorization_code&code=SplxlOBeZQQYbYS6WxSbIA&client_id=myNativeApp&code_verifier=0x1234
56789abcdef&redirect_uri=http://3gpp.mcptt/cb
```

B.3.1.4 Token Response

If the access token request is valid and authorized, the IdM server returns an ID token, access token and refresh token to the MCPTT client; otherwise it will return an error.

An example of a successful response might look like:

EXAMPLE:

```
HTTP/1.1 200 OK
    Content-Type: application/json;charset=UTF-8
    Cache-Control: no-store
    Pragma: no-cache
    {

"access_token":"eyJhbGciOiJSUzI1NiJ9.eyJtY3B0dF9pZCI6ImFsaWNlQG9yZy5jb20iLCJleHAiOjE0NTM1MDYxMjEsInN
jb3BlIjpbIm9wZW5pZCIsIjNncHA6bWNwdHQ6cHR0X3NlcnZlciJdLCJjbGllbnRfaWQiOiJtY3B0dF9jbGllbnQifQ.XYIqai4Y
KSZCKRNMLipGC_5nV4BE79IJpvjexWjIqqcqiEx6AmHHIRo0mhcxeCESrXei9krom9e8Goxr_hgF3szvgbwl8JRbFuv97XgepDLj
Eq4jL3Cbu41Q9b0WdXAdFmeEbiB8wo_xggiGwv6IDR1b3TgAAsdjkRxSK4ctIKPaOJSRmM7MKMcKhIug3BEkSC9-
aXBTSIv5fAGN-ShDbPvHycBpjzKWXBvMIR5PaCg-9fwjELXZXdRwz8C6JbRM8aqzhdt4CVhQ3-Arip-S9CKd0tu-
qhHfF2rvJDRlg8ZBiihdPH8mJs-qpTFep_1-kON3mL0_g54xVmlMwN0XQA",
  "refresh_token":"Y7NSzUJuS0Jp7G4SKpBKSOJVHIZxFbxqsqCIZhOEk9",
"id_token":"eyJhbGciOiJSUzI1NiJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwiYXVkIjoibWNwdHRfY2xpZW50IiwiaXNzIjoiSWR
NUy5zZXJ2ZXIuY29tOjkwMzEiLCJpYXQiOjE0NTM0OTgxNTgsImV4cCI6MTQ1MzQ5ODQ1OCwibWNwdHRfaWQiOiJhbGljZUBvcmc
uY29tIn0.Dpn7AhIMaqMEgg12NYUUfJGSFJMPG8M2li9FLtPotDlHvwU2emBws8z5JLw81SXQnoLqZ8ZF8tIhZ1W7uuMbufF4Wsr
7PAadZixz3CnV2wxFV9qR_VA1-0ccDTPukUsRHsic0SgZ3aIbcYKd6VsehFe_GDwfqysYzD7yPwCfPZo",
"token_type": "Bearer",
"expires_in": 7199
}
```

The MCPTT client may now validate the user with the ID token and configure itself for the user (e.g. by extracting the MCPTT ID from the ID Token). The MCPTT client then uses the access token to make authorized requests to the MCPTT resource servers (MCPTT server, KMS, etc.) on behalf of the end user.

B.4 Refreshing an access token

B.4.0 General

To protect against leakage or other compromise, access token lifetimes are typically short lived (though it is ultimately a matter of security policy & configuration by the service provider). Some client types can be issued longer-lived refresh tokens, which enable them to refresh the access token and avoid having to prompt the user for authentication again when the access token expires. Refresh tokens are available only to clients utilizing the authorization code grant type (native MCPTT clients and web-based MCPTT clients). Refresh tokens are not given to clients utilizing the implicit grant type (browser-based MCPTT clients). Figure B.4.0-1 shows how Native MCPTT clients can use the refresh token as a grant type to obtain new access tokens.

Figure B.4.0-1: Requesting a new access token

B.4.1 Access token request

To obtain an access token from the MCPTT IdM server using a refresh token, the MCPTT client makes an access token request to the token endpoint of the IdM server. The MCPTT client does this by adding the following parameters using the "application/x-www-form-urlencoded" format, with a character encoding of UTF-8 in the HTTP request entity-body. The access token request standard parameters are shown in table B.4.1-1.

Table B.4.1-1: Access token request standard required parameters

Parameter	Values
grant_type	REQUIRED. The value shall be set to "refresh_token".
scope	Space-delimited set of permissions that the MCPTT client requests. Note that the scopes requested using this grant type shall be of equal to or lesser than scope of the original scopes requested by the MCPTT client as part of the original authorization request.

An example of a token request for MCPTT Connect might look like:

EXAMPLE:

```
POST /as/token.oauth2 HTTP/1.1
Host: IdM.server.com:9031
Cache-Control: no-cache
Content-Type: application/x-www-form-urlencoded

grant_type=refresh_token&refresh_token=Y7NSzUJuS0Jp7G4SKpBKSOJVHIZxFbxqsqCIZhOEk9&scope=3gpp:mcptt:p
tt_server
```

If the MCPTT client was provided with client credentials by the MCPTT IdM server, then the client shall authenticate with the token endpoint of the IdM server utilizing the client credential (shared secret or public-private key pair) established during the client registration phase.

B.4.2 Access token response

In response to the access token request (above) the token endpoint on the IdM server will return an access token to the MCPTT client, and optionally another refresh token.

An example successful response for MCPTT Connect might look like:

EXAMPLE:

```
HTTP/1.1 200 OK
    Content-Type: application/json;charset=UTF-8
    Cache-Control: no-store
    Pragma: no-cache
{
 "access_token":"eyJhbGciOiJSUzI1NiJ9.eyJtY3B0dF9pZCI6ImFsaWNlQG9yZy5jb20iLCJleHAiOjE0NTM1MDYxMjEsIn
Njb3BlIjpbIm9wZW5pZCIsIjNncHA6bWNwdHQ6cHR0X3NlcnZlciJdLCJjbGllbnRfaWQiOiJtY3B0dF9jbGllbnQifQ.XYIqai4
YKSZCKRNMLipGC_5nV4BE79IJpvjexWjIqqcqiEx6AmHHIRo0mhcxeCESrXei9krom9e8Goxr_hgF3szvgbwl8JRbFuv97XgepDL
jEq4jL3Cbu41Q9b0WdXAdFmeEbiB8wo_xggiGwv6IDR1b3TgAAsdjkRxSK4ctIKPaOJSRmM7MKMcKhIug3BEkSC9-
aXBTSIv5fAGN-ShDbPvHycBpjzKWXBvMIR5PaCg-9fwjELXZXdRwz8C6JbRM8aqzhdt4CVhQ3-Arip-S9CKd0tu-
qhHfF2rvJDRlg8ZBiihdPH8mJs-qpTFep_1-kON3mL0_g54xVmlMwN0XQA",
   "refresh_token": "iTxQYALqlc7uLyFGpnl8tR8Y9gkw91mFy2qC9Yywkz",
   "token_type": "Bearer",
   "expires_in": 7199
}
```

It is possible to configure the IdM server to confirm that the user account is still valid each time the refresh token is presented, and to revoke the refresh token if not. This security practice is RECOMMENDED.

B.5 Using the token to access MCPTT resource servers

MCPTT Connect shall initially support the bearer access token type. Access tokens of type "bearer" are communicated from the MCPTT client to MCPTT resource servers by including the access token in the HTTP Authorization Header, per IETF RFC 6750 [20].

The access token is opaque to the MCPTT client, meaning that the client does not have any knowledge of the access token itself. The client will be given some metadata corresponding to the access token, such as its expiration time, so that it does not send an expired access token to MCPTT resource servers. If the access token is presented to an MCPTT resource server and the scope is invalid or the token is expired or revoked, the MCPTT resource server should return an error message indicating such to the MCPTT client.

B.6 Token validation

B.6.1 ID token validation

The MCPTT client shall validate the id token as per section 3.1.3.7 of the OpenID Connect 1.0 specification.

B.6.2 Access token validation

MCPTT resource servers shall validate access tokens received from the MCPTT client according to IETF RFC 7519 [32].

Annex C (informative): OpenID connect detailed flow

C.1 Detailed flow for MCPTT user authentication and registration using OpenID Connect

Figure D.1-1 shows the detailed flow for MCPTT User Authentication and Registration using the OpenID Connect messages as described in annex B.

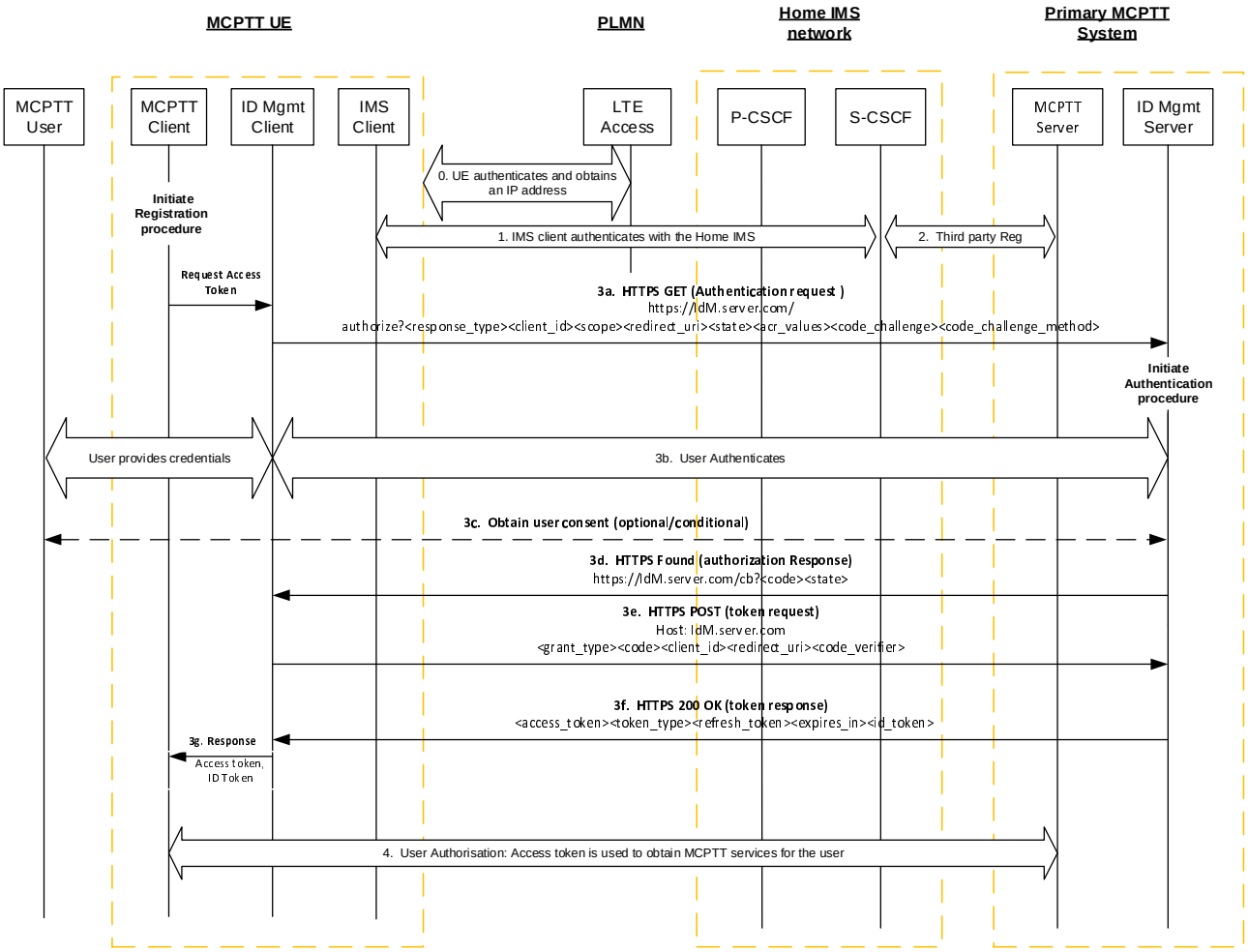


Figure C.1-1: OpenID Connect MCPTT User Authentication and Registration

Step 0: The UE attaches to the network, establishes normal connectivity, and sets up network security as defined in 3GPP TS 33.401 [14]. Local P-CSCF in the Home IMS network is discovered at this point.

Step 1: The UE IMS Client authenticates with the Home IMS network. For IMS authentication, 3GPP TS 33.203 [9] applies.

Step 2: The SIP core sends a SIP 3rd Party Registration to the MCPTT application Server, notifying it of the MCPTT UE SIP registration. The 3rd party REGISTER message includes the registered IMPU and S-CSCF's SIP-URI or IP Address.

Step 3a: The IdM client in the UE issues a HTTPS Authentication request to the OIDC based IdM Server in the MCPTT network. The client includes the code_challenge value in this request.

Step 3b: The user provides the MCPTT User Identity and associated credentials to the IdM server. The user is successfully authenticated (and optionally authorized) by the IdM Server.

Step 3c: The IdM Server may optionally request user consent for granting the MCPTT client access to MCPTT services in the MCPTT Server.

Step 3d: The IdM Server generates an authorization code that is associated with the code_challenge provided by the client. It sends a browser redirect HTTP message with the Authorization Response containing the authorization code.

Step 3e: The UE IdM Client performs a HTTP POST request to exchange the authorization code for an access token. In the request, the client includes the code-verifier string. This string is cryptographically associated with the code_challenge value provided in the Authorization Request in Step 3a.

Step 3f: The IdM Server verifies the IdM Client based on the received code-verifier string and issues a 200 OK with an access token and ID token (specific to the MCPTT user and MCPTT services) included in it.

NOTE: The server verifies by calculating the code challenge from the received code_verifier and comparing it with the code_challenge value provided by the client in Step 3a.

Step 3g: The access token and ID token are provided to the MCPTT client.

Step 4: The MCPTT UE performs user authorization.

Annex D (Normative): KMS provisioning messages to support MCPTT

D.1 General aspects

This annex specifies the key management procedures between the KMS and the key management client that allows keys to be provisioned to the key management client based on a identity. It describes the requests and responses for the authorization following provisioning messages:

- KMS Initialize.
- KMS KeyProvision.
- KMS CertCache.

All KMS communications are made via HTTPS. The MCPTT key management client is provisioned via XML content in the KMS's response. The XML content is designed to be extendable to allow KMS/client providers to add further information in the XML. Where the interface is extended, a different XML namespace should be used (so that may be ignored by non-compatible clients).

It is assumed that transmissions between the KMS and the key management client are secure and that the KMS has authenticated the identity of the key management client.

Additionally, to allow the transmission of key material securely between a secure element within the KMS and a secure element within the key management client, a security extension is defined which allows messages to be signed and key material to be encrypted using a shared Transport Key (TrK).

D.2 KMS requests

Requests to the KMS are made to specific resource URIs. Resource URIs are rooted under the tree "/keymanagement/identity/v1" for a particular domain. For example the resource path to initialize a user within the domain "example.org" is:

EXAMPLE 1:

```
http://example.org/keymanagement/identity/v1/init
```

To make a "KMS Initialize" request the key management client shall make a HTTP POST request to the subdirectory "init" i.e. Request-URI takes the form of:

EXAMPLE 2:

```
…/keymanagement/identity/v1/init
```

To make a "KMS KeyProvision" request the key management client shall make a HTTP POST request to the subdirectory "keyprov" i.e. Request-URI takes the form of

EXAMPLE 3:

```
…/keymanagement/identity/v1/keyprov
```

Optionally, the Request-URI of the POST request may contain a specific user or group URI which the key management client would like the KMS to provision. The URI shall be within a subdirectory of "keyprov". For example, the user URI "user@example.org" is provisioned via a request to: "/keymanagement/identity/v1/keyprov/user%40example.org". Additionally, if the Request-URI contains a specific URI, the client may also request a specific time which the client would like the KMS to provision. The time URI shall be the same time as used in the MIKEY payload, a NTP-UTC 64-bit timestamp as defined in IETF RFC 5905 [29]. For example, if the user required keys specifically for 23rd Feb 2014 at 08:39:14.000 UTC, the request would be:

EXAMPLE 4:

```
…/keymanagement/identity/v1/keyprov/user%40example.org/D6B4323200000000
```

To make a "KMS CertCache" request the key management client shall make a HTTP POST request to the subdirectory "certcache". For example, the request-URI takes the form of "/keymanagement/identity/v1/certcache". If a cache has been previously received, the request URI may optionally be directed to the subdirectory indicating the number of the client's latest version of the cache. For example, the request-URI takes the form of

EXAMPLE 5:

```
…/keymanagement/identity/v1/certcache/12345
```

If the optional security extension is used, requests may be authenticated using the shared Transport Key (TrK). To achieve this, the request should be accompanied with an XML payload containing details of the request, signed by the shared TrK.

D.3 KMS responses

D.3.0 General

This clause defines the HTTP responses made by the KMS to the three KMS requests. The KMS attaches XML content to the HTTP responses. The XML serves to provision the client based upon its request.

The header format of the XML content is the same for each request, though each response carries differing content within a "KMSMessage" tag. There are two types of XML content provided by the KMS within the "KMSMessage" tag; KMS Certificates and (private) user Key Set provisioning.

In response to a "KMS Initialize" request, the KMS shall respond with the KMS's own certificate (the Root KMS certificate) within a "KMSInit" tag.

In response to a "KMS KeyProvision" request, the KMS shall provision appropriate user Key Sets within a "KMSKeyProv" tag.

In response to a "KMS CertCache" request, the KMS shall provision a cache of KMS certificates allowing inter-domain communications within a "KMSCertCache" tag.

D.3.1 KMS certificates

D.3.1.1 Description

A KMS Certificate is a certificate that applies to an entire domain of users. A Certificate consists of XML containing the information required to encrypt messages to a domain of users and verify signatures from the domain of users.

A KMS has exactly one root certificate, which contains the public keys used by the KMS. The root certificate is the only certificate for which the KMS has the private keys and is able to issue user-specific key material. Should the root certificate need to be updated, a new KMS with a new KMS URI should be established with a new root certificate.

It is assumed that the MCPTT user is managed by a single KMS. The root certificate for this KMS is required to encrypt messages to the MCPTT user, and verify signatures from the MCPTT user.

The KMS may also provision a number of 'external' KMS certificates to allow inter-domain communications.

D.3.1.2 Fields

The KMS Certificate shall be within a XML tag named "KmsCertificate". This type shall have the following subfields.

Table D.3.1.2-1: Contents of a KMS Certificate

Name	Description
Version	(Attribute) The version number of the certificate type (1.1.0).
Role	(Attribute) This shall indicate whether the certificate is a "Root" or "External" certificate.
CertUri	(Optional) The URI of the Certificate (this object).
KmsUri	The URI of the KMS which issued the Certificate.
Issuer	(Optional) String describing the issuing entity.
ValidFrom	(Optional) Date from which the Certificate may be used.
ValidTo	(Optional) Date at which the Certificate expires.
Revoked	(Optional) A Boolean value defining whether a Certificate has been revoked.
UserIDFormat	Shall contain the value '2', indicating that the generation mechanism defined in clause F.2.1 shall be used.
UserKeyPeriod	The number of seconds that each user key issued by this KMS should be used (e.g. '2419200').
UserKeyOffset	The offset in seconds from 0h on 1st Jan 1900 that the segmentation of key periods starts (e.g. '0').
PubEncKey	The SAKKE Public Key, "Z_T", as defined in [10]. This is an OCTET STRING encoding of an elliptic curve point.
PubAuthKey	The ECCSI Public Key, "KPAK" as defined in [9]. This is an OCTET STRING encoding of an elliptic curve point.
ParameterSet	(Optional) The choice of parameter set used for SAKKE and ECCSI (e.g. '1').
KmsDomainList	(Optional) List of domains associated with the certificate.

D.3.1.3 User IDs

To secure communications with a specific user, the initiator shall compose the User Identifier (UID) to which the message will be encrypted. IETF RFC 6509 [11] defines a UID generation scheme for Tel URIs, however this cannot be used with MCPTT as MCPTT IDs may not be Tel URIs.

Clause F.2.1 defines the UID generation scheme for MCPTT. This shall be identified within the KMS certificate by using the value '2' within the UserIDFormat field.

D.3.2 User Key Provision

D.3.2.1 Description

User keys are private information associated to a user's identity (UserID) which allow a user to decrypt information encrypted to that identity and sign information as that identity. User keys are provisioned as XML containing the key information required and associated metadata.

D.3.2.2 Fields

The KMS shall provision keys within an XML tag named "KmsKeySet". This shall have the following subfields.

Table D.3.2.2-1: Contents of a KMS Key Set

Name	Description
Version	(Attribute) The version number of the key provision XML (1.1.0).
KmsUri	The URI of the KMS which issued the key set.
CertUri	(Optional) The URI of the Certificate which may be used to validate the key set.
Issuer	(Optional) String describing the issuing entity.
UserUri	URI of the user for which the key set is issued.
UserID	UID corresponding to the key set.
ValidFrom	(Optional) Date and time from which the key set may be used.
ValidTo	(Optional) Date and time at which the key set expires.
KeyPeriodNo	Current Key Period No. since 1 January 1900 (e.g. 1514)
Revoked	(Optional) A Boolean value defining whether the key set has been revoked.
UserDecryptKey	The SAKKE "Receiver Secret Key" as defined in [10]. This is an OCTET STRING encoding of an elliptic curve point as defined in section 2.2 of [31].
UserSigningKeySSK	The ECCSI private Key, "SSK" as defined in [9]. This is an OCTET STRING encoding of an integer as described in section 6 of [30].
UserPubTokenPVT	The ECCSI public validation token, "PVT" as defined in [9]. This is an OCTET STRING encoding of an elliptic curve point as defined in Section 2.2 of [31].

NOTE: The key may be valid outside of its defined key period of use to enable decryption of old messages encrypted to the user.

D.3.3 Example KMS response XML

D.3.3.1 Example KMSInit XML

If the security extension is used, it is assumed that before this response is received, the secure element within the KMS and the secure element within the MCPTT key management client have shared a bootstrap TrK, e.g. 'tk.11.user@example.org'.

In this example, the KMS provides the MCPTT user with the KMS root ceriticate and a new TrK to protect future KMS communications. Keys are encrypted and the message is signed using the bootstrap TrK.

EXAMPLE:

```
<?xml version="1.0" encoding="UTF-8"?>
<SignedKmsResponse xmlns= "TOBEDEFINED" xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
    xmlns:ds = "http://www.w3.org/2000/09/xmldsig#" xmlns:se = "TOBEDEFINED"
    xsi:schemaLocation = "TOBEDEFINED SE_KmsInterface_XMLSchema.xsd" Id = "xmldoc">
<KmsResponse xmlns= "TOBEDEFINED"  Version = "1.0.0">
<KmsUri>kms.example.org</KmsUri>
  <UserUri>user@example.org</UserUri>
  <Time>2014-01-26T10:05:52</Time>
  <KmsId>KMSProvider12345</KmsId>
  <ClientReqUrl>http://kms.example.org/keymanagement/identity/v1/init</ClientReqUrl>
  <KmsMessage>
    <KmsInit Version = "1.0.0" xsi:type = "se:KmsInitTkType">
      <KmsCertificate Version = "1.1.0" Role = "Root">
        <CertUri>cert1.kms.example.org</CertUri>
        <KmsUri>kms.example.org</KmsUri>
        <Issuer>www.example.org</Issuer>
        <ValidFrom>2000-01-26T00:00:00</ValidFrom>
        <ValidTo>2025-01-26T23:59:59</ValidTo>
        <Revoked>false</Revoked>
        <UserIdFormat>2</UserIdFormat>
        <UserKeyPeriod>2592000</UserKeyPeriod>
        <UserKeyOffset>0</UserKeyOffset>
        <PubEncKey>029A2F</PubEncKey>
        <PubAuthKey>029A2F</PubAuthKey>
        <ParameterSet>1</ParameterSet>
        <KmsDomainList>
```

```
          <KmsDomain>sec1.example.org</KmsDomain>
          <KmsDomain>sec2.example.org</KmsDomain>
        </KmsDomainList>
      </KmsCertificate>
      <NewTransportKey xmlns= "TOBEDEFINED">
            <EncryptedKey xmlns="http://www.w3.org/2001/04/xmlenc#"
Type="http://www.w3.org/2001/04/xmlenc#EncryptedKey">
              <EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#kw-aes256"/>
              <ds:KeyInfo>
                <ds:KeyName>tk.11.user@example.org</KeyName>
              </ds:KeyInfo>
              <CipherData>
                <CipherValue>DEADBEEF</CipherValue>
              </CipherData>
              <CarriedKeyName>tk.12.user@example.org</CarriedKeyName>
            </EncryptedKey>
          </NewTransportKey>
    </KmsInit>
  </KmsMessage>
</KmsResponse>
<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
    <SignedInfo>
      <CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
      <SignatureMethod Algorithm="http://www.w3.org/2001/04/xmldsig-more#hmac-sha256">
        <HMACOutputLength>128</HMACOutputLength>
      </SignatureMethod>
      <Reference URI="#xmldoc">
        <DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
        <DigestValue>nnnn</DigestValue>
      </Reference>
    </SignedInfo>
    <SignatureValue>DEADBEEF</SignatureValue>
    <KeyInfo>
      <KeyName>tk.11.user@example.org</KeyName>
    </KeyInfo>
  </Signature>
</SignedKmsResponse>
```

D.3.3.2 Example KMSKeyProv XML

In this example, the MCPTT user's key material is provided for two user identifiers. The key material includes the UserDecryptKey (see IETF RFC 6508 [10]) and the UserSigningKey and PVT (see IETF RFC 6507 [9]) for each identifier.

As the security extension has been used, the key material is encrypted and the message signed using the shared TrK. Additionally, a new TrK is provided as part of the key provision.

EXAMPLE:

```
<?xml version="1.0" encoding="UTF-8"?>
<SignedKmsResponse xmlns= "TOBEDEFINED" xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
    xmlns:ds = "http://www.w3.org/2000/09/xmldsig#" xmlns:se = "TOBEDEFINED"
    xsi:schemaLocation = "TOBEDEFINED SE_KmsInterface_XMLSchema.xsd" Id = "xmldoc">
<KmsResponse xmlns= "TOBEDEFINED"  Version = "1.0.0">
  <KmsUri>kms.example.org</KmsUri>
  <UserUri>user@example.org</UserUri>
  <Time>2014-01-26T10:07:14</Time>
  <KmsId>KMSProvider12345</KmsId>
  <ClientReqUrl>http://kms.example.org/keymanagement/identity/v1/keyprov</ClientReqUrl>
  <KmsMessage>
    <KmsKeyProv Version = "1.0.0" xsi:type = "se:KmsKeyProvTkType">
      <KmsKeySet Version = "1.1.0">
        <KmsUri>kms.example.org</KmsUri>
        <CertUri>cert1.kms.example.org</CertUri>
        <Issuer>www.example.org</Issuer>
        <UserUri>user@example.org</UserUri>
        <UserID>0123456789ABCDEF0123456789ABCDEF</UserID>
        <ValidFrom>2015-12-30T00:00:00</ValidFrom>
        <ValidTo>2016-03-29T23:59:59</ValidTo>
        <KeyPeriodNo>1514</KeyPeriodNo>
        <Revoked>false</Revoked>
        <UserDecryptKey xsi:type = "se:EncKeyContentType">
          <EncryptedKey xmlns = "http://www.w3.org/2001/04/xmlenc#">
            <EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#kw-aes256"/>
            <ds:KeyInfo>
```

```
            <ds:KeyName>tk.12.user@example.org</KeyName>
          </ds:KeyInfo>
          <CipherData>
            <CipherValue>DEADBEEF</CipherValue>
          </CipherData>
        </EncryptedKey>
      </UserDecryptKey>
      <UserSigningKeySSK xsi:type = "se:EncKeyContentType">
        <EncryptedKey xmlns = "http://www.w3.org/2001/04/xmlenc#">
          <EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#kw-aes256"/>
          <ds:KeyInfo>
            <ds:KeyName>tk.12.user@example.org</KeyName>
          </ds:KeyInfo>
          <CipherData>
            <CipherValue>DEADBEEF</CipherValue>
          </CipherData>
      </EncryptedKey>
      </UserSigningKeySSK>
      <UserPubTokenPVT xsi:type = "se:EncKeyContentType">
        <EncryptedKey xmlns = "http://www.w3.org/2001/04/xmlenc#">
          <EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#kw-aes256"/>
          <ds:KeyInfo>
            <ds:KeyName>tk.12.user@example.org</KeyName>
          </ds:KeyInfo>
          <CipherData>
            <CipherValue>DEADBEEF</CipherValue>
          </CipherData>
        </EncryptedKey>
      </UserPubTokenPVT>
    </KmsKeySet>
    <KmsKeySet Version = "1.1.0">
      <KmsUri>kms.example.org</KmsUri>
      <CertUri>cert1.kms.example.org</CertUri>
      <Issuer>www.example.org</Issuer>
      <UserUri>user.psuedonym@example.org</UserUri>
      <UserID>00112233445566778899aabbccddeeff</UserID>
      <ValidFrom>2015-12-30T00:00:00</ValidFrom>
      <ValidTo>2016-03-29T23:59:59</ValidTo>
      <ValidTo>2016-03-29T23:59:59</ValidTo>
  <KeyPeriodNo>1514</KeyPeriodNo>
      <Revoked>false</Revoked>
      <UserDecryptKey xsi:type = "se:EncKeyContentType">
        <EncryptedKey xmlns = "http://www.w3.org/2001/04/xmlenc#">
          <EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#kw-aes256"/>
          <ds:KeyInfo>
            <ds:KeyName>tk.12.user@example.org</KeyName>
          </ds:KeyInfo>
          <CipherData>
            <CipherValue>DEADBEEF</CipherValue>
          </CipherData>
        </EncryptedKey>
      </UserDecryptKey>
      <UserSigningKeySSK xsi:type = "se:EncKeyContentType">
        <EncryptedKey xmlns = "http://www.w3.org/2001/04/xmlenc#">
          <EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#kw-aes256"/>
          <ds:KeyInfo>
            <ds:KeyName>tk.12.user@example.org</KeyName>
          </ds:KeyInfo>
          <CipherData>
            <CipherValue>DEADBEEF</CipherValue>
          </CipherData>
        </EncryptedKey>
      </UserSigningKeySSK>
      <UserPubTokenPVT xsi:type = "se:EncKeyContentType">
        <EncryptedKey xmlns = "http://www.w3.org/2001/04/xmlenc#">
          <EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#kw-aes256"/>
          <ds:KeyInfo>
            <ds:KeyName>tk.12.user@example.org</KeyName>
          </ds:KeyInfo>
          <CipherData>
            <CipherValue>DEADBEEF</CipherValue>
          </CipherData>
        </EncryptedKey>
      </UserPubTokenPVT>
    </KmsKeySet>
    <NewTransportKey xmlns= "TOBEDEFINED">
          <EncryptedKey xmlns="http://www.w3.org/2001/04/xmlenc#"
Type="http://www.w3.org/2001/04/xmlenc#EncryptedKey">
```

```
              <EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#kw-aes256"/>
              <ds:KeyInfo>
                <ds:KeyName>tk.12.user@example.org</KeyName>
              </ds:KeyInfo>
              <CipherData>
                <CipherValue>DEADBEEF</CipherValue>
              </CipherData>
              <CarriedKeyName>tk.13.user@example.org</CarriedKeyName>
            </EncryptedKey>
          </NewTransportKey>
    </KmsKeyProv>
  </KmsMessage>
</KmsResponse>
<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
    <SignedInfo>
      <CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
      <SignatureMethod Algorithm="http://www.w3.org/2001/04/xmldsig-more#hmac-sha256">
        <HMACOutputLength>128</HMACOutputLength>
      </SignatureMethod>
      <Reference URI="#xmldoc">
        <DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
        <DigestValue>nnnn</DigestValue>
      </Reference>
    </SignedInfo>
    <SignatureValue>DEADBEEF</SignatureValue>
    <KeyInfo>
      <KeyName>tk.12.user@example.org</KeyName>
    </KeyInfo>
  </Signature>
</SignedKmsResponse>
```

D.3.3.3 Example KMSCertCache XML

In this example, a number of 'external' KMS certificates are provided to the MCPTT user. These allow the user to encrypt to users managed by a different KMS.

As the security extension is in use, the message is signed using the TrK.

EXAMPLE:

```
<?xml version="1.0" encoding="UTF-8"?>
<SignedKmsResponse xmlns= "TOBEDEFINED" xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
    xmlns:ds = "http://www.w3.org/2000/09/xmldsig#" xmlns:se = "TOBEDEFINED"
    xsi:schemaLocation = "TOBEDEFINED SE_KmsInterface_XMLSchema.xsd" Id = "xmldoc">
<KmsResponse xmlns= "TOBEDEFINED"  Version = "1.0.0">
  <KmsUri>kms.example.org</KmsUri>
  <UserUri>user@example.org</UserUri>
  <Time>2014-01-26T10:14:12</Time>
  <KmsId>KMSProvider12345</KmsId>
  <ClientReqUrl>http://kms.example.org/keymanagement/identity/v1/certcache</ClientReqUrl>
  <KmsMessage>
    <KmsCertCache Version = "1.0.0">
      <SignedKmsCertificate Id = "cert1">
        <KmsCertificate Version = "1.1.0" Role = "External">
          <CertUri>cert2.kms.example.org</CertUri>
          <KmsUri>kms.example.org</KmsUri>
          <Issuer>www.example.org</Issuer>
          <ValidFrom>2000-01-26T00:00:00</ValidFrom>
          <ValidTo>2100-01-26T23:59:59</ValidTo>
          <Revoked>false</Revoked>
          <UserIdFormat>2</UserIdFormat>
          <UserKeyPeriod>2592000</UserKeyPeriod>
          <UserKeyOffset>0</UserKeyOffset>
          <PubEncKey>029A2F</PubEncKey>
          <PubAuthKey>029A2F</PubAuthKey>
          <ParameterSet>1</ParameterSet>
          <KmsDomainList>
            <KmsDomain>sec3.example.org</KmsDomain>
          </KmsDomainList>
        </KmsCertificate>
        <Signature xmlns = "http://www.w3.org/2000/09/xmldsig#">
          <SignedInfo>
            <CanonicalizationMethod Algorithm = "http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
            <SignatureMethod Algorithm = "http://www.w3.org/2001/04/xmldsig-more#ecdsa-sha256"/>
            <Reference URI = "#cert1">
              <DigestMethod Algorithm = "http://www.w3.org/2001/04/xmlenc#sha256"/>
```

```
            <DigestValue>nnnn</DigestValue>
          </Reference>
        </SignedInfo>
        <SignatureValue>DEADBEEF</SignatureValue>
        <KeyInfo>
          <KeyName>cert1.kms.example.org</KeyName>
        </KeyInfo>
      </Signature>
    </SignedKmsCertificate>
    <SignedKmsCertificate Id = "cert2">
      <KmsCertificate Version = "1.1.0" Role = "External">
        <CertUri>cert1.kms.another.example.org</CertUri>
        <KmsUri>kms.another.example.org</KmsUri>
        <Issuer>www.another.example.org</Issuer>
        <ValidFrom>2000-01-26T00:00:00</ValidFrom>
        <ValidTo>2100-01-26T23:59:59</ValidTo>
        <Revoked>false</Revoked>
        <UserIdFormat>2</UserIdFormat>
        <UserKeyPeriod>604800</UserKeyPeriod>
        <UserKeyOffset>432000</UserKeyOffset>
        <PubEncKey>029A2F</PubEncKey>
        <PubAuthKey>029A2F</PubAuthKey>
        <ParameterSet>1</ParameterSet>
        <KmsDomainList>
          <KmsDomain>another.example.org</KmsDomain>
        </KmsDomainList>
      </KmsCertificate>
      <Signature xmlns = "http://www.w3.org/2000/09/xmldsig#">
        <SignedInfo>
          <CanonicalizationMethod Algorithm = "http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
          <SignatureMethod Algorithm = "http://www.w3.org/2001/04/xmldsig-more#ecdsa-sha256"/>
          <Reference URI = "#cert2">
            <DigestMethod Algorithm = "http://www.w3.org/2001/04/xmlenc#sha256"/>
            <DigestValue>nnnn</DigestValue>
          </Reference>
        </SignedInfo>
        <SignatureValue>DEADBEEF</SignatureValue>
        <KeyInfo>
          <KeyName>cert1.kms.example.org</KeyName>
        </KeyInfo>
      </Signature>
    </SignedKmsCertificate>
  </KmsCertCache>
 </KmsMessage>
</KmsResponse>
<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
    <SignedInfo>
      <CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
      <SignatureMethod Algorithm="http://www.w3.org/2001/04/xmldsig-more#hmac-sha256">
        <HMACOutputLength>128</HMACOutputLength>
      </SignatureMethod>
      <Reference URI="#xmldoc">
        <DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
        <DigestValue>nnnn</DigestValue>
      </Reference>
    </SignedInfo>
    <SignatureValue>DEADBEEF</SignatureValue>
    <KeyInfo>
      <KeyName>tk.13.user@example.org</KeyName>
    </KeyInfo>
  </Signature>
</SignedKmsResponse>
```

D.3.4 KMS Response XML schema

D.3.4.1 Base XML schema

This clause contains the base XML schema (without the security extension) for KMS responses:

```
<?xml version="1.0" encoding="UTF-8"?>

<xsd:schema xmlns:xsd = "http://www.w3.org/2001/XMLSchema" xmlns:ds =
"http://www.w3.org/2000/09/xmldsig#" xmlns = "TOBEDEFINED" targetNamespace = "TOBEDEFINED"
elementFormDefault = "qualified" version = "1.0">
  <xsd:import namespace = "http://www.w3.org/2000/09/xmldsig#" schemaLocation = "xmldsig-core-
schema.xsd"/>

  <xsd:element type = "KmsResponseType" name = "KmsResponse"/>

  <xsd:complexType name = "KmsResponseType">
    <xsd:sequence>
      <xsd:element type = "xsd:anyURI" name = "KmsUri" maxOccurs = "1"/>
      <xsd:element type = "xsd:anyURI" name = "UserUri" maxOccurs = "1"/>
      <xsd:element type = "xsd:dateTime" name = "Time" maxOccurs = "1"/>
      <xsd:element type = "xsd:string" name = "KmsId" minOccurs = "0" maxOccurs = "1"/>
      <xsd:any namespace = "##other" processContents = "lax" minOccurs = "0" maxOccurs =
"unbounded"/>
      <xsd:element type = "xsd:anyURI" name = "ClientReqUrl" maxOccurs = "1"/>
      <xsd:element name = "KmsMessage" maxOccurs = "1" minOccurs = "0">
        <xsd:complexType>
          <xsd:choice maxOccurs = "1" minOccurs = "0">
            <xsd:element type = "KmsInitType" name = "KmsInit"/>
            <xsd:element type = "KmsKeyProvType" name = "KmsKeyProv"/>
            <xsd:element type = "KmsCertCacheType" name = "KmsCertCache"/>
            <xsd:any namespace = "##other" processContents = "lax" minOccurs = "0" maxOccurs =
"unbounded"/>
          </xsd:choice>
          <xsd:anyAttribute namespace = "##other" processContents = "lax"/>
        </xsd:complexType>
      </xsd:element>
      <xsd:element type = "ErrorType" name = "KmsError" minOccurs = "0" maxOccurs = "1"/>
      <xsd:any namespace = "##other" processContents = "lax" minOccurs = "0" maxOccurs =
"unbounded"/>
    </xsd:sequence>
    <xsd:attribute name = "Id" type = "xsd:string"/>
    <xsd:attribute name = "Version" type = "xsd:string" fixed="1.0.0"/>
    <xsd:anyAttribute namespace = "##other" processContents = "lax"/>
  </xsd:complexType>

  <xsd:complexType name = "ErrorType">
    <xsd:sequence>
      <xsd:element type = "xsd:integer" name = "ErrorCode" maxOccurs = "1"/>
      <xsd:element type = "xsd:string" name = "ErrorMsg" maxOccurs = "1"/>
      <xsd:any namespace = "##other" processContents = "lax" minOccurs = "0" maxOccurs =
"unbounded"/>
    </xsd:sequence>
    <xsd:attribute name = "Id" type = "xsd:string"/>
    <xsd:attribute name = "Version" type = "xsd:string"/>
    <xsd:anyAttribute namespace = "##other" processContents = "lax"/>
  </xsd:complexType>

  <xsd:complexType name = "KmsInitType">
    <xsd:sequence>
      <xsd:choice maxOccurs = "1">
        <xsd:element type = "SignedKmsCertificateType" name = "SignedKmsCertificate"/>
        <xsd:element type = "KmsCertificateType" name = "KmsCertificate"/>
      </xsd:choice>
      <xsd:any namespace = "##other" processContents = "lax" minOccurs = "0" maxOccurs =
"unbounded"/>
    </xsd:sequence>
    <xsd:attribute name = "Id" type = "xsd:string"/>
    <xsd:attribute name = "Version" type = "xsd:string"/>
    <xsd:anyAttribute namespace = "##other" processContents = "lax"/>
  </xsd:complexType>

  <xsd:complexType name = "KmsKeyProvType">
    <xsd:sequence>
      <xsd:element type = "KmsKeySetType" name = "KmsKeySet" minOccurs = "0" maxOccurs =
"unbounded"/>
```

```
      <xsd:any namespace = "##other" processContents = "lax" minOccurs = "0" maxOccurs =
"unbounded"/>
    </xsd:sequence>
    <xsd:attribute name = "Id" type = "xsd:string"/>
    <xsd:attribute name = "Version" type = "xsd:string" fixed="1.0.0"/>
    <xsd:anyAttribute namespace = "##other" processContents = "lax"/>
  </xsd:complexType>

  <xsd:complexType name = "KmsCertCacheType">
    <xsd:sequence>
      <xsd:choice maxOccurs = "unbounded" minOccurs = "0">
        <xsd:element type = "SignedKmsCertificateType" name = "SignedKmsCertificate"/>
        <xsd:element type = "KmsCertificateType" name = "KmsCertificate"/>
      </xsd:choice>
      <xsd:any namespace = "##other" processContents = "lax" minOccurs = "0" maxOccurs =
"unbounded"/>
    </xsd:sequence>
    <xsd:attribute name = "Id" type = "xsd:string"/>
    <xsd:attribute name = "Version" type = "xsd:string" fixed="1.0.0"/>
    <xsd:attribute name = "CacheNum" type = "xsd:integer"/>
    <xsd:anyAttribute namespace = "##other" processContents = "lax"/>
  </xsd:complexType>


  <xsd:element name = "SignedKmsCertificate" type = "SignedKmsCertificateType"/>
  <xsd:complexType name = "SignedKmsCertificateType">
    <xsd:sequence>
      <xsd:element name = "KmsCertificate" type = "KmsCertificateType"/>
      <xsd:element ref = "ds:Signature" minOccurs = "0"/>
    </xsd:sequence>
    <xsd:attribute name = "Id" type = "xsd:string"/>
    <xsd:anyAttribute namespace = "##other" processContents = "lax"/>
  </xsd:complexType>

  <xsd:element name = "KmsCertificate" type = "KmsCertificateType"/>
  <xsd:complexType name = "KmsCertificateType">
    <xsd:sequence>
      <xsd:element type = "xsd:anyURI" name = "KmsUri" maxOccurs = "1"/>
      <xsd:element type = "xsd:anyURI" name = "CertUri" maxOccurs = "1" minOccurs = "0"/>
      <xsd:element type = "xsd:string" name = "Issuer" maxOccurs = "1" minOccurs = "0"/>
      <xsd:element type = "xsd:dateTime" name = "ValidFrom" maxOccurs = "1" minOccurs = "0"/>
      <xsd:element type = "xsd:dateTime" name = "ValidTo" maxOccurs = "1" minOccurs = "0"/>
      <xsd:element type = "xsd:boolean" name = "Revoked" maxOccurs = "1" minOccurs = "0"/>
      <xsd:element type = "xsd:string" name = "UserIdFormat" maxOccurs = "1"/>
      <xsd:element type = "xsd:integer" name = "UserKeyPeriod" maxOccurs = "1"/>
      <xsd:element type = "xsd:integer" name = "UserKeyOffset" maxOccurs = "1"/>
      <xsd:element type = "xsd:hexBinary" name = "PubEncKey" maxOccurs = "1"/>
      <xsd:element type = "xsd:hexBinary" name = "PubAuthKey" maxOccurs = "1"/>
      <xsd:element type = "xsd:integer" name = "ParameterSet" maxOccurs = "1" minOccurs = "0"/>
      <xsd:element name = "KmsDomainList" maxOccurs = "1" minOccurs = "0">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element type = "xsd:anyURI" name = "KmsDomain" maxOccurs = "unbounded"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
      <xsd:any namespace = "##other" processContents = "lax" minOccurs = "0" maxOccurs =
"unbounded"/>
    </xsd:sequence>
    <xsd:attribute name = "Id" type = "xsd:string"/>
    <xsd:attribute name = "Version" type = "xsd:string" fixed="1.1.0"/>
    <xsd:attribute name = "Role" type = "RoleType"/>
    <xsd:anyAttribute namespace = "##other" processContents = "lax"/>
  </xsd:complexType>

  <xsd:simpleType name = "RoleType">
    <xsd:restriction base = "xsd:string">
      <xsd:enumeration value = "Root"/>
      <xsd:enumeration value = "External"/>
    </xsd:restriction>
  </xsd:simpleType>


  <xsd:element name = "KmsKeySet" type = "KmsKeySetType"/>

  <xsd:complexType name = "KmsKeySetType">
    <xsd:sequence>
      <xsd:element type = "xsd:anyURI" name = "KmsUri" maxOccurs = "1"/>
```

```
      <xsd:element type = "xsd:anyURI" name = "CertUri" maxOccurs = "1" minOccurs = "0"/>
      <xsd:element type = "xsd:string" name = "Issuer" maxOccurs = "1" minOccurs = "0"/>
      <xsd:element type = "xsd:anyURI" name = "UserUri" maxOccurs = "1"/>
      <xsd:element type = "xsd:string" name = "UserID" maxOccurs = "1"/>
      <xsd:element type = "xsd:dateTime" name = "ValidFrom" maxOccurs = "1" minOccurs = "0"/>
      <xsd:element type = "xsd:dateTime" name = "ValidTo" maxOccurs = "1" minOccurs = "0"/>
      <xsd:element type = "xsd:integer" name = "KeyPeriodNo" maxOccurs = "1"/>
      <xsd:element type = "xsd:boolean" name = "Revoked" maxOccurs = "1" minOccurs = "0"/>
      <xsd:element type = "KeyContentType" name = "UserDecryptKey" maxOccurs = "1"/>
      <xsd:element type = "KeyContentType" name = "UserSigningKeySSK" maxOccurs = "1"/>
      <xsd:element type = "KeyContentType" name = "UserPubTokenPVT" maxOccurs = "1"/>
    </xsd:sequence>
    <xsd:attribute name = "Id" type = "xsd:string"/>
    <xsd:attribute name = "Version" type = "xsd:string" fixed = "1.1.0"/>
    <xsd:anyAttribute namespace = "##other" processContents = "lax"/>
  </xsd:complexType>

  <xsd:complexType name = "KeyContentType">
    <xsd:simpleContent>
      <xsd:extension base="xsd:hexBinary">
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>

</xsd:schema>
```

D.3.4.2 Security Extension to KMS response XML schema

This clause contains the security extension to the base XML schema.

```
<?xml version="1.0" encoding="UTF-8"?>

<xsd:schema xmlns:xsd = "http://www.w3.org/2001/XMLSchema" xmlns:ds =
"http://www.w3.org/2000/09/xmldsig#"  xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
xmlns:ikms = "TOBEDEFINED" xmlns = "TOBEDEFINED" targetNamespace = "TOBEDEFINED" elementFormDefault
= "qualified" version = "1.0">

  <xsd:import namespace = "http://www.w3.org/2000/09/xmldsig#" schemaLocation = "xmldsig-core-
schema.xsd"/>
    <xsd:import namespace="http://www.w3.org/2001/04/xmlenc#" schemaLocation="xenc-schema.xsd"/>
    <xsd:import namespace="TOBEDEFINED" schemaLocation="KmsInterface_XMLSchema.xsd"/>


<xsd:element type="EncKeyContentType" name="NewTransportKey"/>
<xsd:element type = "SignedKmsResponseType" name = "SignedKmsResponse"/>
<xsd:element name="SignedKmsRequest" type="SignedKmsRequestType"/>

<xsd:complexType name="EncKeyContentType">
  <xsd:complexContent>
    <xsd:restriction base="ikms:KeyContentType">
      <xsd:sequence>
      <xsd:element ref="xenc:EncryptedKey" maxOccurs="1" minOccurs="1"/>
    </xsd:sequence>
    </xsd:restriction>
  </xsd:complexContent>
</xsd:complexType>


<xsd:complexType name = "SignedKmsResponseType">
    <xsd:sequence>
      <xsd:element ref = "ikms:KmsResponse"/>
      <xsd:element ref = "ds:Signature" minOccurs = "0"/>
    </xsd:sequence>
    <xsd:attribute name="Id" type="xsd:string"/>
    <xsd:anyAttribute namespace = "##other" processContents = "lax"/>
  </xsd:complexType>


<xsd:complexType name="KmsInitTkType">
  <xsd:complexContent>
    <xsd:restriction base="ikms:KmsInitType">
      <xsd:sequence>
      <xsd:choice maxOccurs = "1">
        <xsd:element ref = "ikms:SignedKmsCertificate"/>
        <xsd:element ref = "ikms:KmsCertificate"/>
      </xsd:choice>
      <xsd:element type="EncKeyContentType" name="NewTransportKey" maxOccurs="1"/>
```

```
      <!-- Can extend in another namespace - for more types of communication-->
      <xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
    </xsd:restriction>
  </xsd:complexContent>
</xsd:complexType>


  <xsd:complexType name = "KmsKeyProvTkType">
    <xsd:complexContent>
    <xsd:restriction base="ikms:KmsKeyProvType">
    <xsd:sequence>
      <xsd:element ref = "ikms:KmsKeySet" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element type="EncKeyContentType" name="NewTransportKey" minOccurs="0"
maxOccurs="unbounded"/>
      <!-- Can extend in another namespace - for more types of communication-->
      <xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
    </xsd:restriction>
  </xsd:complexContent>
  </xsd:complexType>

  <xsd:complexType name="SignedKmsRequestType">
    <xsd:sequence>
      <xsd:element name="KmsRequest" type="KmsRequestType"/>
      <xsd:element ref="ds:Signature"/>
    </xsd:sequence>
    <xsd:attribute name="Id" type="xsd:string"/>
    <xsd:anyAttribute namespace = "##other" processContents = "lax"/>
  </xsd:complexType>

<xsd:complexType name = "KmsRequestType">
          <xsd:sequence>
            <xsd:element type="xsd:anyURI" name="UserUri" maxOccurs="1"/>
            <xsd:element type="xsd:anyURI" name="KmsUri" maxOccurs="1"/>
            <xsd:element type="xsd:dateTime" name="Time" maxOccurs="1"/>
            <xsd:element type="xsd:string" name="ClientId" minOccurs="0" maxOccurs="1"/>
            <xsd:element type="xsd:string" name="DeviceId" minOccurs="0" maxOccurs="1"/>
            <xsd:element type="xsd:anyURI" name="ClientReqUrl" maxOccurs="1"/>
            <xsd:element type="ikms:ErrorType" name="ClientError" minOccurs="0" maxOccurs="1"/>
             <!-- Can extend in another namespace - for more types of communication-->
            <xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
          </xsd:sequence>
          <xsd:attribute name="Id" type="xsd:string"/>
          <xsd:attribute name="Version" type="xsd:string" fixed="1.0.0"/>
          <xsd:anyAttribute namespace="##other" processContents="lax"/>
        </xsd:complexType>

</xsd:schema>
```

Annex E (normative): MIKEY message formats for media security

E.1 General aspects

E.1.0 Introduction

MIKEY-SAKKE as defined in IETF RFC 6509 [11] is used to transport Group Master Keys (GMKs) from a Group Management Server to a Group Management Client on a MCPTT UE and Private Call Keys (PCKs) between MCPTT UEs.

The GMK is encrypted to the UID generated from the receiving user's MCPTT ID and current time period. It is signed using the UID generated from the URI associated to the Group Management Server and current time period. Similarly, the PCK is encrypted to the UID generated from the receiving user's MCPTT ID and current time period. It is signed using the UID generated from the initiating user's MCPTT ID and current time period. Details of this process are defined in IETF RFC 6508 [10] and IETF RFC 6507 [9]. The generation of the MIKEY-SAKKE UID is defined in clause F.2.1.

The GMK and PCK shall be 16 octets in length.

E.1.1 MIKEY common fields

If the transmitter requires an ACK for a transmission this is indicated by setting the V-bit in the MIKEY common header. For distribution of GSKs for one-to-many communications, the V-bit shall not be set.

Each MIKEY message contains the timestamp field (TS). The timestamp field shall be TS type NTP-UTC (TS type 0), and hence is a 64-bit UTC time.

E.2 MIKEY message structure for GMK distribution

The MIKEY-SAKKE message shall include the Common Header payload, Timestamp payload, RAND payload, IDRi payload, IDRr payload, IDRkmsi payload, IDRkmsr payload, SAKKE payload and a SIGN (ECCSI) payload. It is recommended that the message also includes a Security Properties payload. Optionally, the message may include a General Extension payload containing a second SAKKE message as described in clause E.5.

In the Common Header payload, the CSB ID field of MIKEY common header shall be the GUK-ID.

The Security Policy (SP) payload is used to specify the security properties of group communications using the GMK. Where no security profile is provided, the following default security profile shall be used.

Table E.2-1: MIKEY Group call SRTP Default Profile

SRTP Type	Meaning	Value	Meaning
0	Encryption Algorithm	6	AES-GCM
1	Session encryption key length	16	16 octets
2	Authentication algorithm	4	RCCm3 (Use of unauthenticated ROC)
4	Session salt key length	12	12 octets
5	SRTP PRF	0	AES-CM
6	Key derivation rate	0	No session key refresh.
13	ROC transmission rate	1	ROC transmitted in every packet.
18	SRTP Authentication tag length	4	4 octets for transmission of ROC
19	SRTCP Authentication tag length	0	ROC need not be transmitted in SRTCP.
20	AEAD authentication tag length	16	16 octets

Identity payloads shall be IDR payloads as defined in section 6.6 of IETF RFC 6043 [25]. The IDRi payload shall contain the MCPTT identifier associated with the group management server. The IDRr payload shall contain the MCPTT ID associated to the group management client. The message shall also include IDRkmsi and IDRkmsr that contains the URI of the MCPTT KMS used by the group management server and MCPTT user respectively.

NOTE: In some deployments MCPTT IDs within these payloads may be treated as private. In this case, the group management server and group management client should substitute these private identities for public identities via a privately-defined mapping.

The SAKKE payload shall encapsulate the GMK to the UID generated from the MCPTT ID of the group management client. Only one GMK key shall be transported in the SAKKE payload. The same GMK shall be encapsulated to each member of the group. The ID Scheme in the SAKKE payload shall be 'URI Scheme' to reflect the generation scheme defined in clause F.2.1.

Editor's note: A new 'ID Scheme' Type value should be requested from IANA in place of the 'URI Scheme' Type value.

The entire MIKEY message shall be signed by including an SIGN payload providing authentication of the group management server. The signature shall be of type 2 (ECCSI). The signature shall use the UID generated from the identifier associated with the group management server.

E.3 MIKEY message structure for PCK distribution

The MIKEY-SAKKE message shall include the Common Header payload, Timestamp payload, RAND payload, IDRi payload, IDRr payload, IDRkmsi payload, IDRkmsr payload, SAKKE payload and a SIGN (ECCSI) payload. It is recommended that the message also includes a Security Properties payload. Optionally, the message may include a General Extension payload containing a second SAKKE message as described in clause E.5.

In the Common Header payload, the CSB ID field of MIKEY common header shall be the PCK-ID. The CS-ID map type shall be GENERIC-ID as defined in IETF RFC 6043 [25].

The Security Properties payload is used to specify the security properties of private calls using the PCK. Where no security profile is provided, the following default security profile shall be used.

Table E.3-1: MIKEY Group call SRTP Default Profile

SRTP Type	Meaning	Value	Meaning
0	Encryption Algorithm	6	AES-GCM
1	Session encryption key length	16	16 octets
4	Session salt key length	12	12 octets
5	SRTP PRF	0	AES-CM
6	Key derivation rate	0	No session key refresh.
20	AEAD authentication tag length	16	16 octets

Identity payloads shall be IDR payloads as defined in section 6.6 of IETF RFC 6043 [25]. The IDRi payload shall contain the MCPTT ID associated with the initiating user. The IDRr payload shall contain the MCPTT ID associated to the receiving user. The message shall also include IDRkmsi and IDRkmsr that contains the URI of the MCPTT KMS used by the initiating user and terminating user respectively.

NOTE: In some deployments MCPTT IDs within these payloads may treated as private. In this case, the initiating and terminating MCPTT UEs should substitute these private identities for public identities via a privately-defined mapping.

The SAKKE payload shall encapsulate the PCK to the UID generated from the MCPTT ID of the terminating user. The ID Scheme in the SAKKE payload shall be 'URI Scheme' to reflect the generation scheme defined in clause F.2.1.

The entire MIKEY message shall be signed by including an SIGN payload providing authentication of initiating user. The signature shall be of type 2 (ECCSI). The signature shall use the UID generated from the MCPTT ID of the initiating user.

E.4 MIKEY message structure for CSK distribution

The MIKEY-SAKKE message shall include the Common Header payload, Timestamp payload, RAND payload, IDRi payload, IDRr payload, IDRkmsi payload, IDRkmsr payload, SAKKE payload and a SIGN (ECCSI) payload. The message may also include a Security Properties payload.

In the Common Header payload, the CSB ID field of MIKEY common header shall be the CSK-ID. The CS-ID map type shall be GENERIC-ID as defined in IETF RFC 6043 [25].

Identity payloads shall be IDR payloads as defined in section 6.6 of IETF RFC 6043 [25]. The IDRi payload shall contain the MCPTT ID associated with the initiating user. The IDRr payload shall contain the MDSI of the MCPTT Domain. The message shall also include IDRkmsi and IDRkmsr that contains the URI of the MCPTT KMS used by the initiating user and MCPTT Server respectively.

NOTE: Where confidentiality of user identifiers is required, the MCPTT ID may be replaced with the UID generated from the MCPTT ID as defined in clause F.2.1.

The SAKKE payload shall encapsulate the CSK to the UID generated from the MDSI of the MCPTT Domain. The ID Scheme in the SAKKE payload shall be 'URI Scheme' to reflect the generation scheme defined in clause F.2.1.

The entire MIKEY message shall be signed by including an SIGN payload providing authentication of initiating user. The signature shall be of type 2 (ECCSI). The signature shall use the UID generated from the MCPTT ID of the initiating user.

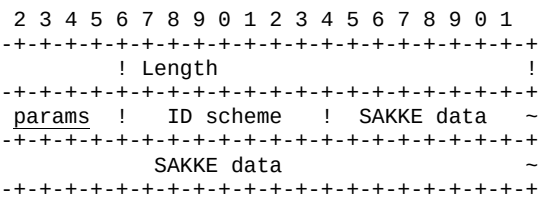
E.5 MIKEY general extension payload to support 'SAKKE-to-self'

In some circumstances it is useful for the initiator to be able to decrypt a MIKEY-SAKKE payload and recover the key (as well as the receiver). For example, where the initiating user is attached to the MCPTT service via more than one MCPTT UE, the other MCPTT UEs associated with the initiating user will also need the key material to be able to join the communication.

To support this scenario, an optional MIKEY General Extension Payload may be added to the MIKEY-SAKKE message. This general extension payload has value 'SAKKE-to-self'. The contents of the payload will be a full SAKKE payload as defined in IETF RFC 6509 [11]. Within the second SAKKE payload the key (GMK or PCK) shall be encapsulated to the UID generated from the MCPTT identifier associated with the initiating user (either group management server or private call initiator). The ID Scheme in the SAKKE payload shall be 'URI Scheme' to reflect the generation scheme defined in clause F.2.1.

Editor's note: A new 'MIKEY General Extension Payload' Type value should be requested from IANA in place of the 'SAKKE-to-self' Type value.

EXAMPLE SAKKE-to-self payload:



The SAKKE-to-self payload encapsulates a SAKKE payload. Consequently, the SAKKE-to-self payload will contain two 'next payload' fields. The second 'next payload' field, which corresponds to the encapsulated SAKKE payload, shall be set to zero and ignored.

E.6 MIKEY general extension payload to encapsulate parameters associated with a GMK

E.6.1 General

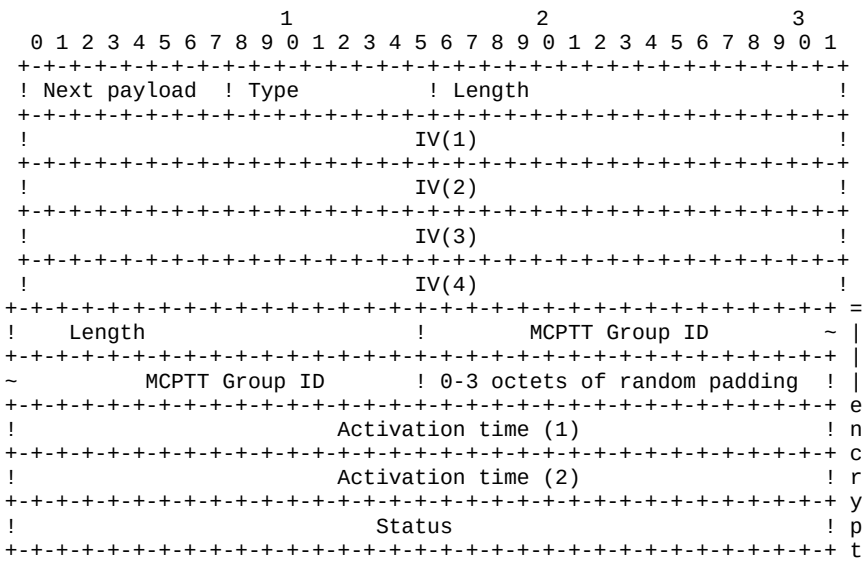
The parameters associated with the GMK shall be contained in the 'General extension payload' specified in IETF RFC 3830 [22] using the 'Vendor ID' Type value and contained within the signed envelope of the MIKEY-SAKKE I_MESSAGE specified in clause E.2. The format and cryptography of the payload are specified in this subclause.

Editor's note: A new '3GPP' Type value should be requested from IANA in place of the 'Vendor ID' Type value.

The four octets consisting of the header of the 'General extension payload' shall be formatted according to IETF RFC 3830 [22]. There shall be seven elements within the 'General extension payload' as follow:

- IV;
- MCPTT group ID, incorporating a length sub-element and 0-3 octets of random padding;
- Activation time;
- Status;
- Text, incorporating a length sub-element and 0-3 octets of random padding;
- Reserved, incorporating a length sub-element;
- 0-15 octets of random padding.

Thus the structure of the 'General extension payload' according to the present document is shown in figure E.6.1-1.



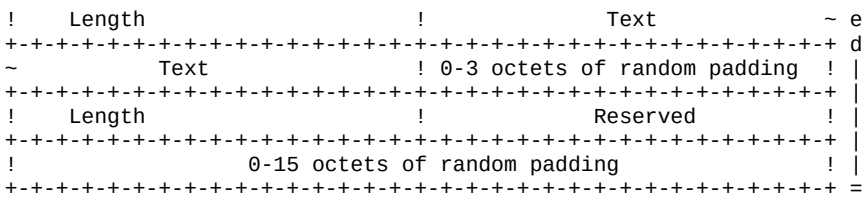


Figure E.6.1-1: Layout of general extension payload for associated parameters of GMK

The elements following the 'General extension payload' header are described in the following subclauses.

E.6.2 IV

The IV shall be randomly chosen by the GMS, and shall be 16 octets in length.

E.6.3 MCPTT group ID

The 'MCPTT group ID' element shall consist of two sub-elements followed by 0-3 bytes of random padding. A two octet 'Length' sub-element shall followed by an 'MCPTT group ID' sub-element, where this sub-element shall be encoded as ASCII 8 bit text. The 'Length' element shall indicate the length in octets of the 'MCPTT group ID' sub-element only, and the count of the length shall not include the 'Length' sub-element itself, and shall not include the length of any following random padding. Following the 'MCPTT group ID' sub-element, 0 -3 octets of random padding shall be added so that the total length of the ('Length' sub-element + 'MCPTT group ID' sub-element + random padding) shall be a multiple of 4 octets.

E.6.4 Activation time

The 'Activation time' element shall define the time in UTC at which the associated GMK is to be made active for transmission. It shall be 8 octets in length.

E.6.5 Text

The 'Text' element shall consist of two sub-elements followed by 0-3 bytes of random padding. A two octet 'Length' sub-element shall followed by a 'Text' sub-element, where this sub-element shall be encoded as ASCII 8 bit text. The 'Length' element shall indicate the length in octets of the 'Text' sub-element only, and the count of the length shall not include the 'Length' sub-element itself, and shall not include the length of any following random padding. Following the 'Text' sub-element, 0 -3 octets of random padding shall be added so that the total length of the ('Length' sub-element + 'Text' sub-element + random padding) shall be a multiple of 4 octets.

E.6.6 Reserved

The 'Reserved' element shall consist of two sub-elements. A two octet 'Length' sub-element shall be followed by a 'Reserved' sub-element, where the definition and encoding of this sub-element is outside the scope of the present document, and shall be ignored by the receiving client. The 'Length' element shall indicate the length in octets of the 'Text' sub-element only, and the count of the length shall not include the 'Length' sub-element itself. The length of the sum of the ('Length' sub-element + 'Reserved' sub-element) shall be a multiple of 4 octets.

E.6.7 Random padding

The five elements specified in the preceding five subclauses shall be padded by 0-15 octets of random data such that the total length of the 'General extension payload' shall be a multiple of 32 octets, to satisfy the block size requirements of the payload protection algorithm.

E.6.8 Cryptography

The concatenated 'MCPTT group ID', 'Activation time', 'Text', 'Reserved' and 'Random padding' elements shall be encrypted using AES-128 in Cipher Block Chaining mode using the IV (16 octets) as Initial Vector, as described in IETF RFC 3602 [23]. The encryption key shall be the GMK.

E.6.9 Status

The 'Status' element shall determine the current status of the GMK. It shall be 4 octets in length. The following values are defined:

0: Revoked

1: Not-revoked

E.7 Hiding identities within MIKEY messages

In some public-safety use cases there is a requirement to protect MCPTT IDs in transit. To protect these identifiers in MIKEY-SAKKE messages the following approach may be taken.

The sensitive MCPTT ID in the IDRr or IDRi field is replaced with the UID generated from the MCPTT ID as defined in clause F.2.1. In the former case, the 'role' of the IDRr field is replaced with a role of IDRuidr. In the latter case, the 'role' of the IDRi field is replaced with a role of IDRuidi.

Editor's Note: 3GPP will need to ask IANA to define two new identifier roles, IDRuidr and IDRuidi.

The processing of the MIKEY-SAKKE I_MESSAGE at the initiator stays the same. If the initiator has hidden its own MCPTT ID, it shall ensure that the SIP message containing the I_MESSAGE contains the initiator's MCPTT ID encrypted to the receiver.

As a consequence of identity hiding, the receiver of the MIKEY-SAKKE I_MESSAGE will be able to check the signature based on the initiator's UID in the IDRuidi field, but initially will be unable to confirm the MCPTT ID that has been used to generate the UID. The receiver will recognize its own UID in the IDRuidr field, and be able to extract the encapsulated key.

Using the encapsulated key or otherwise, the receiver is able to extract associated metadata in the message, including the initiator's MCPTT ID. On obtaining the initiator's MCPTT ID, the receiver is able to compute the UID and ensure this matches the UID in the IDRuidi field. By performing this check, the receiver has authenticated the I_MESSAGE.

Annex F (normative): Key derivation and hash functions

F.1 KDF interface and input parameter construction

F.1.1 General

This annex specifies the use of the Key Derivation Function (KDF) specified in 3GPP TS 33.220 [5] for the current specification. This annex specifies how to construct the input string, S, to the KDF (which is input together with the relevant key). For each of the distinct usages of the KDF, the input parameters S are specified below.

F.1.2 FC value allocations

The FC number space used is controlled by 3GPP TS 33.220 [5].

F.1.3 Calculation of the User Salt for GUK-ID generation

When calculating a User Salt using the GMK for generating the GUK-ID from the GMK-ID, the following parameters shall be used to form the input S to the KDF that is specified in annex B of 3GPP TS 33.220 [17]:

- FC = 0x50.
- P0 = MCPTT ID.
- L0 = length of above (i.e. 0x00 0x17).

The GMK and MCPTT ID follow the encoding also specified in annex B of 3GPP TS 33.220 [17]. The 28 least significant bits of the 256 bits of the KDF output shall be used as the User Salt.

F.1.4 Calculation of keys for application data protection

The two keys used to protect either signalling plane confidentiality, or signalling plane integrity are derived from the XPK, using the KDF that is specified in annex B of 3GPP TS 33.220 [27].

The following parameters shall be used to form the input S to the KDF that is specified in annex B of 3GPP TS 33.220 [27]. The key used by the KDF shall be the XPK:

- FC = 0x51, (for signalling plane confidentiality), or
- FC = 0x52 (for signalling plane integrity).
- P0 = MCPTT ID.
- L0 = length of above, expressed in number of bytes (i.e. 0x00 0x17).
- P1 = XPK-ID.
- L1 = length of above, expressed in number of bytes (i.e. 0x00 0x17).

The MCPTT ID and XPK-ID follow the encoding also specified in annex B of 3GPP TS 33.220 [27].

Where the XPK is 128-bits, the output keys shall be 128-bits and hence the 128 least significant bits of the 256 bits of the KDF output shall be used as the signalling protection key. Where the XPK is 256-bits, the output keys shall be 256-bits and hence the entire output of the KDF shall be used.

F.2 Hash Functions

F.2.1 Generation of MIKEY-SAKKE UID

Section 3.2 of IETF RFC 6509 [11] defines an identifier for use in MIKEY SAKKE in section 3.2, referred to as the UID in the present document. This requires a Tel-URI as the user's URI and monthly key periods. As MCPTT IDs may not be Tel-URIs, this UID format cannot be used within MCPTT. This clause defines how the 256-bit MIKEY-SAKKE UID is generated using a generic identifier and generic key period.

The MIKEY-SAKKE UID is generated by hashing a fixed string, the identifier of the user, the identifier of the KMS, the key period length, the current key period number and their respective lengths.

The input to the hash function shall be encoded as specified in clause B.1 of 3GPP TS 33.220 [17]. The hash function shall be SHA-256 as specified in [18]. The full 256-bit output shall be used as the identifier within MIKEY-SAKKE (referred to as 'ID' in IETF RFC 6507 [9] and 'a' or 'b' within IETF RFC 6508 [10].

FC = 0x00

P0 = The fixed string: "MIKEY-SAKKE-UID"

L0 = Length of P0 value

P1 = Identifier (e.g. MCPTT ID)

L1 = Length of P1 value

P2 = KMS Identifier (e.g. secgroup1.kms.example.org)

L2 = Length of P2 value

P3 = Key Period length in seconds (e.g. 2592000)

L3 = Length of P3 value

P4 = Key Period offset in seconds (e.g. 0)

L4 = Length of P4 value

P5 = Current Key Period No. since 0h on 1 January 1900 (e.g. 553)

L5 = Length of P5 value

NOTE: The key derivation function defined in clause B.1 of 3GPP TS 33.220 [17] is not used, therefore the FC value should only be considered as a dummy value.

P0 is a fixed 15 character string encoded as described in annex B of 3GPP TS 33.220 [17]. P1 is the identifier, which for MCPTT would be the MCPTT ID. P2 is the identifier of the KMS, and uniquely identifies the public key used for encryption and signing. P3 is the integer representing the number of seconds in a key period. P4 is the offset from 0h on 1 January 1900 and shall be less than P3. It sets the time at which keys are changed over. Both P3 and P4 are extracted from the KMS certificate and encoded as integers as described in annex B of 3GPP TS 33.220 [17]. P5 is the integer representing the current key period number since 0h on 1 January 1900, which may be calculated as:

P5 = Floor ((TIME - P4) / P3)

Where TIME is a NTP timestamp, i.e., a number in seconds relative to 0h on 1 January 1900. P4 is encoded as described in annex B of 3GPP TS 33.220 [17].

NOTE 1: When used to generate a UID for encrypting using a MIKEY payload, P1 will commonly be the 'ID Data' from the IDRr payload, P2 will be the encoded 'ID Data' from the IDRkmsr payload, and TIME will be the NTP timestamp within the MIKEY payload.

NOTE 2: When used to generate a UID for signing a MIKEY payload, P1 will commonly be the 'ID Data' from the IDRi payload, P2 will commonly be the 'ID Data' from the IDRkmsi payload, and TIME will be the NTP timestamp within the MIKEY payload.

Annex G (informative): Change history

Change history							
Date	**Meeting**	**TDoc**	**CR**	**Rev**	**Cat**	**Subject/Comment**	**New version**
2016-06	SA#72	SP-160383	0001	2	F	Architectural clarifications and corrections	13.1.0
2016-06	SA#72	SP-160383	0002	2	F	Fix KMS reference points	13.1.0
2016-06	SA#72	SP-160383	0004	1	F	Technical text clarifications	13.1.0
2016-06	SA#72	SP-160383	0005	1	C	Fixing Floor Control Key Management	13.1.0
2016-06	SA#72	SP-160383	0006	1	F	Addition of security overview	13.1.0
2016-06	SA#72	SP-160383	0007	-	C	Sending GMK to unaffilliated users	13.1.0
2016-06	SA#72	SP-160383	0010	-	F	Correction to integrity protection	13.1.0
2016-06	SA#72	SP-160383	0011	2	F	Fix off-network provisioning	13.1.0
2019-09	SA#73	SP-160578	0012	-	F	Correction of some implementation errors	13.2.0
2019-09	SA#73	SP-160578	0013	1	F	Signing of Access Tokens	13.2.0
2019-09	SA#73	SP-160578	0014	1	F	Fix IdM interfaces	13.2.0
2019-09	SA#73	SP-160578	0015	1	F	Corrections to 7.2.2.1, 7.2.2.2, 7.2.2.3, 9.1.3.1, 9.1.3.3, 9.1.3.4, 9.3.1	13.2.0
2019-09	SA#73	SP-160578	0016	2	F	Clarification on floor control signalling protection	13.2.0
2019-09	SA#73	SP-160578	0017	-	F	Clarifications to 33.179	13.2.0
2019-09	SA#73	SP-160578	0018	-	F	Correcting GMK revokation	13.2.0

History

Document history		
V13.0.0	May 2016	Publication
V13.1.0	August 2016	Publication
V13.2.0	October 2016	Publication