ETSI TS 133 180 V14.0.0 (2017-07)

LTE;
Security of the mission critical service
(3GPP TS 33.180 version 14.0.0 Release 14)

Reference
DTS/TSGS-0333180ve00

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

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 service (MCX). The architecture includes mechanisms to protect the Common Functional Architecture and security mechanisms for mission critical applications. This includes Push-To-Talk (MCPTT), Video (MCVideo) and Data (MCData). Additionally, security mechanisms relating to on-network use, off-network use, roaming, migration, interconnection, interworking and multiple security domains are described.

This specification complements the Common Functional Architecture defined in TS 23.280 [36], the functional architecture for MCPTT defined in 3GPP TS 23.379 [2], the functional architecture for MCVideo defined in 3GPP TS 23.281 [37] and the functional architecture for MCData defined in 3GPP TS 23.282 [38].

The MC 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 23.379: "Functional architecture and information flows to support Mission Critical Push To Talk (MCPTT); 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] 3GPP TS 33.179: "Security of Mission Critical Push To Talk (MCPTT) over LTE".

[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)".

[36] 3GPP TS 23.280: "Common functional architecture to support mission critical services; Stage 2".

[37] 3GPP TS 23.281: "Functional architecture and information flows for mission critical video; Stage 2".

[38] 3GPP TS 23.282: "Functional model and information flows for Mission Critical Data".

[39] 3GPP TS 23.002: "Network Architecture".

[40] IETF RFC 2045: "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies".

[41] IETF RFC 2392: "Content-ID and Message-ID Uniform Resource Locators".

[42] NIST Special Publication 800-38D: "Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC".

[43] IETF RFC 5116: "An Interface and Algorithms for Authenticated Encryption".

[45] IETF RFC 7521: “Assertion Framework for OAuth 2.0 Client Authentication and Authorization Grants”.

[46] IETF RFC 7523: “JSON Web Token (JWT) Profile for OAuth 2.0 Client Authentication and Authorization Grants”.

[47] 3GPP TS 22.280: " Mission Critical Services Common Requirements; Stage 1".

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.

MCX: Mission critical services where “MCX” may be substituted with the term “MCPTT”, “MCVideo”, “MCData”, or any combination thereof.

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].

CMS	Configuration Management Server
CS	Crypto Session
CSB-ID	Crypto Session Bundle Identifier
CSC	Common Services Core
CSK	Client-Server Key
CSK-ID	Client-Server Key Identifier
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
JSON	JavaScript Object Notation
JWS	JSON Web Signature
JWT	JSON Web Token
KDF	Key Derivation Function
KFC	Key For Control Signalling
KFC-ID	Key for Floor Control Identifier
KMS	Key Management Server
MBCP	Media Burst Control Protocol
MCData	Mission Critical Data
MCPTT	Mission Critical Push to Talk
MCVideo	Mission Critical Video
MCX	Mission Critical Services
MKFC	Multicast Key for Floor Control
MSCCK	MBMS subchannel control key
MuSiK	Multicast Signalling Key
MKI	Master Key Identifier
OIDC	OpenID Connect
PCK	Private Call Key

PCK-ID	Private Call Key Identifier
PKCE	Proof Key for Code Exchange
PSK	Pre-Shared Key
SPK	SIP Protection Key
SRTCP	Secure Real-Time Transport Control Protocol
SRTP	Secure Real-Time Transport Protocol
SSRC	Synchronization Source
TBCP	Talk Burst Control Protocol
TGK	Traffic Generating Key
TrK	KMS Transport Key
UID	User Identifier for MIKEY-SAKKE (referred to as the 'Identifier' in RFC 6509 [11])

4 Overview of Mission Critical Security

4.1 General

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

- Protection of the signalling plane used by the MC Service, defined in clause 6.1 and 6.2.
- Protection of inter/intra domain interfaces, defined in clause 6.3.

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

- Authentication and authorisation of users to the MC Service, defined in clause 5.1.
- Protection of sensitive application signalling within the MC Service, defined in clause 9.
- Security of RTCP (e.g. floor control, transmission control) within the MC Service, defined in clause 9.
- Security of data signalling within the MCData Service, defined in clause 8.
- End-to-end security of user media within the MC Service. Defined in clause 7 for MCPTT and MCVideo services and defined in clause 8 for the MCData service.

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

4.2 Signalling plane security architecture

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

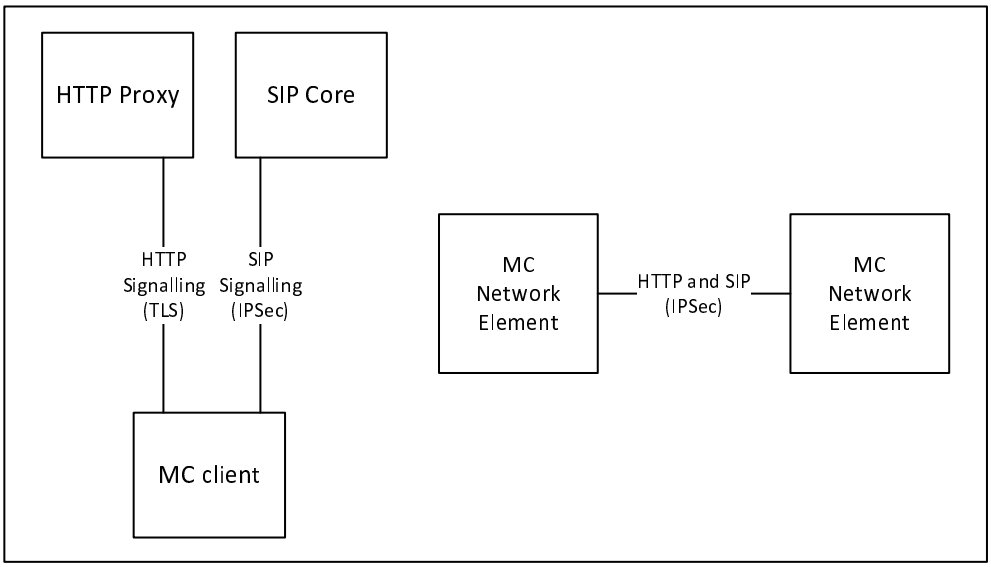


Figure 4.2-1: Signalling plane security architecture

Signalling from the MC client is passed over both HTTP and SIP. The signalling plane security mechanisms for client to server interfaces and between network elements are defined in clause 6.

4.3 MC system security architecture

4.3.1 General

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

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

Media security allow protection of MC media within the MC system. It is applied end-to-end between MC clients. Under normal operation, MC network entities such as the MCX Servers, are unable to access the content of media within the MC system.

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.

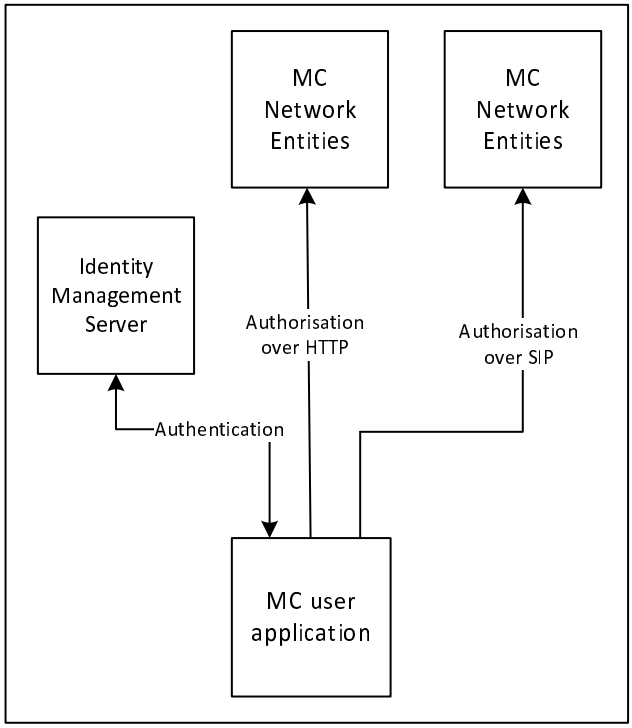


Figure 4.3.2-1: User authentication and authorisation

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 5.1.

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.

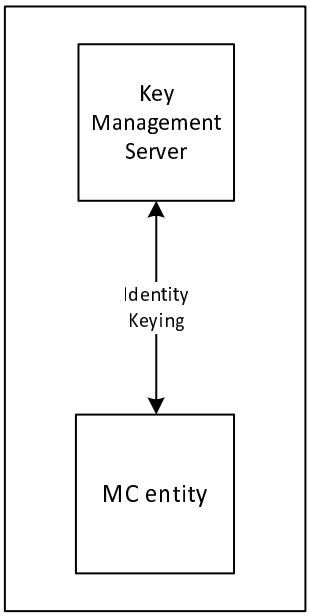


Figure 4.3.3-1: Identity keying of MC entities

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 MC network entities also require identity key material including the MCX 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 5.3.

4.3.4 Protection of application plane signalling

Application plane signalling security protects application signalling between the MC client and the MCX server. Initial 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. Following initial key distribution, the MCX server may perform a 'key download' procedure to update key material, and to key the client to allow multicast signalling to be protected.

There are a variety of types of application plane signalling, including:

- XML signalling within SIP payloads
- Control signalling (e.g. RTCP for floor control or transmission control).
- MCData signalling payloads within SIP payloads.

In each case, the same root key material is used to protect the signalling when the signalling is unicast on the uplink or downlink. Should the signalling be multicast on the downlink, the MCX Server will distribute key material for this purpose and use this key material to protect multicast signalling.

The security architecture is shown in Figure 4.3.4-1.

The mechanisms to provide application plane signalling security are defined in clause 9.

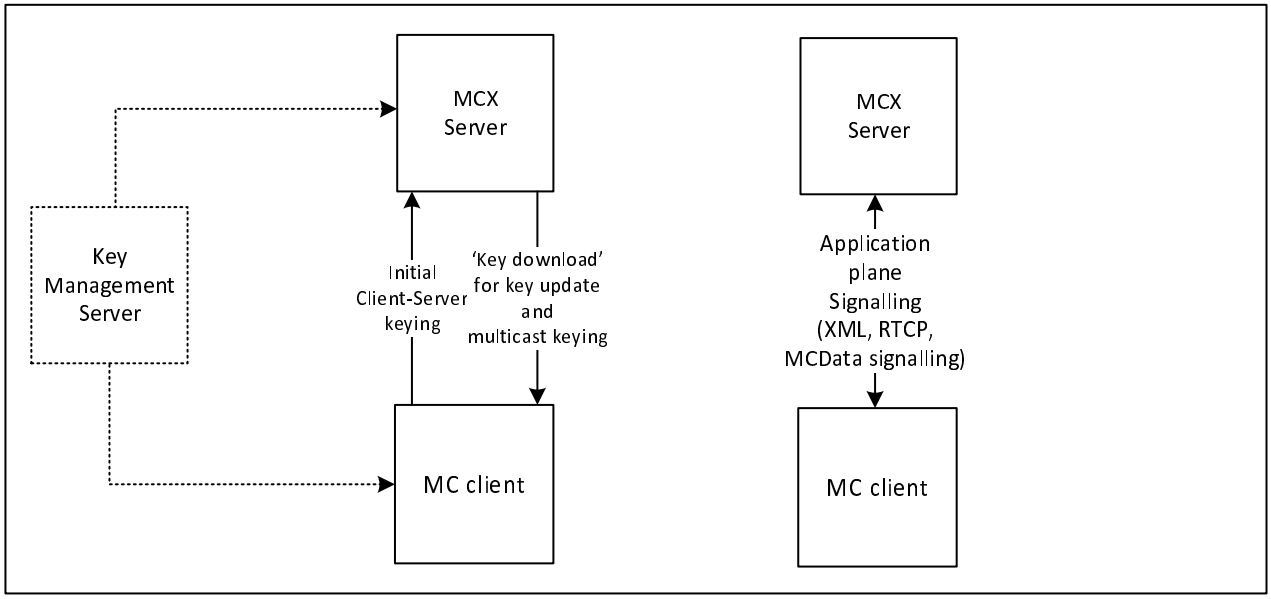


Figure 4.3.4-1: Application plane signalling security

Application plane signalling security can also be applied between MCX servers. In this case the MCX 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 MC users to support group communications and private communications for the MCPTT, MCVideo or MCData services. The intention is for media to be able to be encrypted end-to-end between MC 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 or IOPS connection.

Key distribution for groups is performed by the Group Management Server. Key distribution for private calls is performed by the initiating MC client. Once a security context is established, the media is protected using the distributed key material. Aditionally, when MC UEs are offline, the security context that is used to protect media security is also used to protect control signalling (e.g. RTCP).

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 MC users that are members of the group. The Group Management Server may choose to distribute the group key to MCX Server(s) to allow the media mixing function within the MCX Server(s) to be used. MC users and MCX servers 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 5.7.

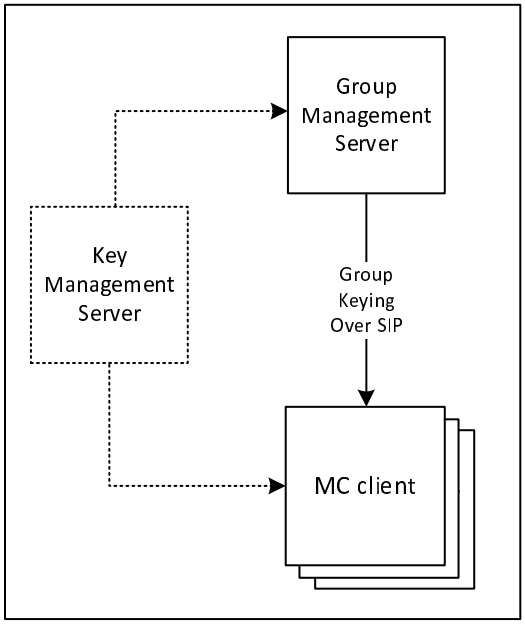


Figure 4.3.5.2-1: Group keying for media security

Once a group key has been shared with MC users, keys are derived from that group key to protect media (and control signalling when the UE is offline).

For MCPTT and MCVideo (specifically RTP), 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 in this case. For MCData, the user-specific key derivation is not required. Media is encrypted within a MCData data payload in this case.

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

Unlike media, control signalling (such as floor control) is protected differently when the UE has a network connection and when it is offline. When the UE has a network connection, control signalling traffic is encrypted to the MC Domain. When it is offline, control signalling is encrypted directly to UEs using a key derived from the root key for the group or private communication. Details of control signalling encryption is provided in clause 9.4.

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

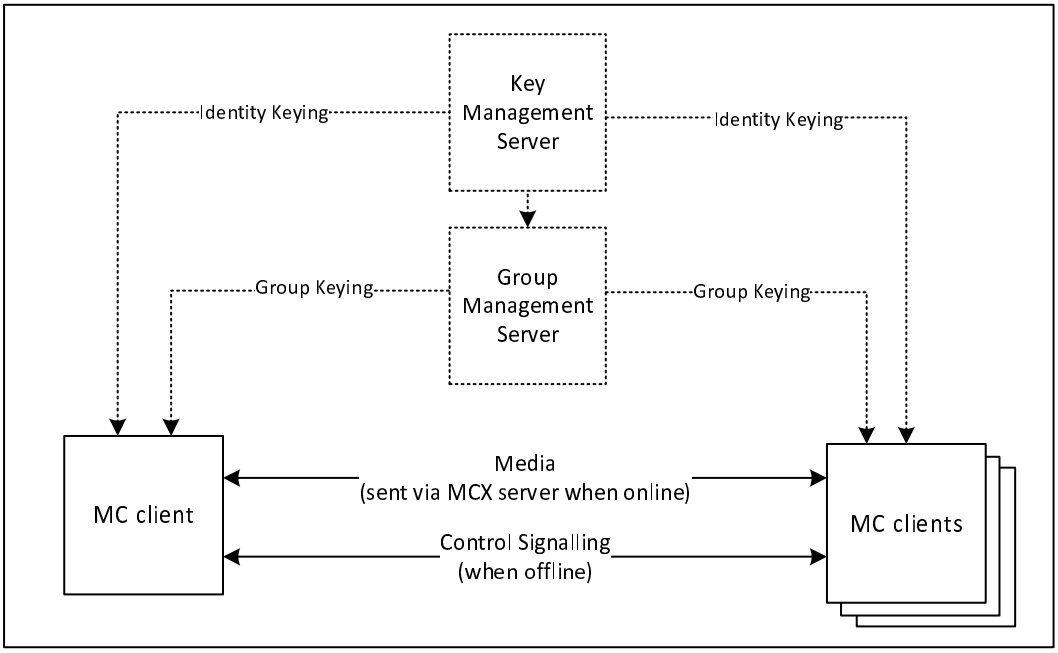


Figure 4.3.5.2-2: Group media protection

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 MC user that is currently registered on the terminating client. As a result, MC users require identity keying by a KMS prior to performing group management.

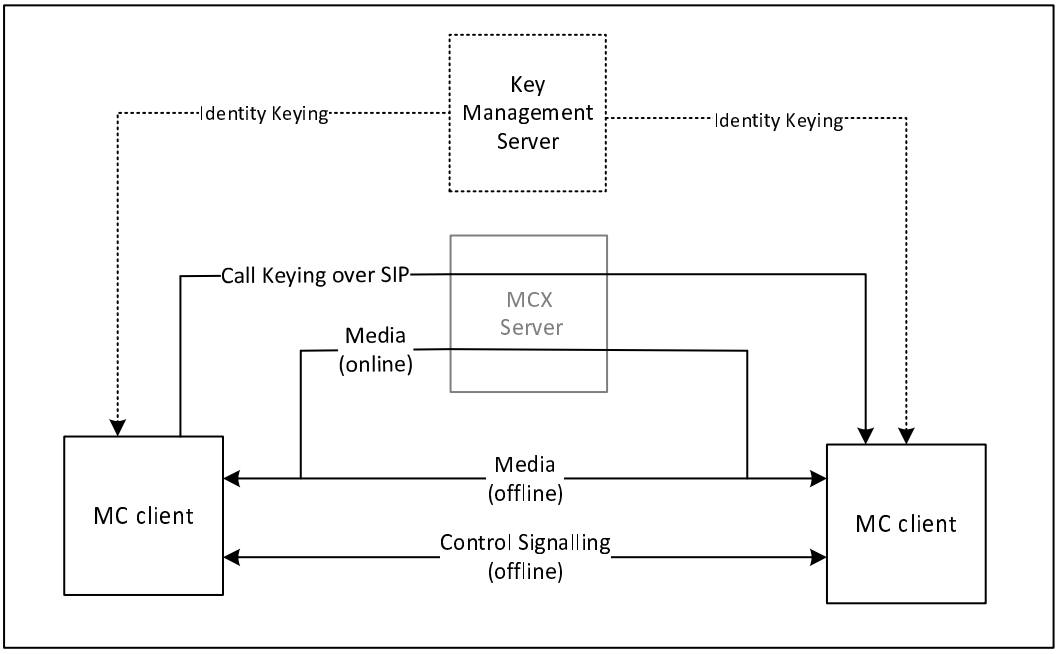


Figure 4.3.5.3-1: Media security for private calls

Figure 4.3.5.3-1 provides an overview of media protection for private calls. For clarity, MC 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 5.6, specific MCPTT and MCVideo procedures are described in clause 7 and specific MCData procedures are in clause 8.

Once private call key distribution has been completed, media and application signalling is protected as described for group communications in clause 4.3.5.2. Media will be routed via the media distribution function in the MCX Server when the UE is online, and directly when the UE is offline. The media security context shall also be used to protect control signalling (e.g. floor control) when the MC UE is offline. Details of media protection may be found in clauses 7 and 8, and control signalling protection is defined in clause 9.4.

5 Common mission critical security framework

5.1 User authentication and authorization

5.1.1 General

The generic steps for MCX user authentication and authorisation is shown in figure 5.1.1-1.

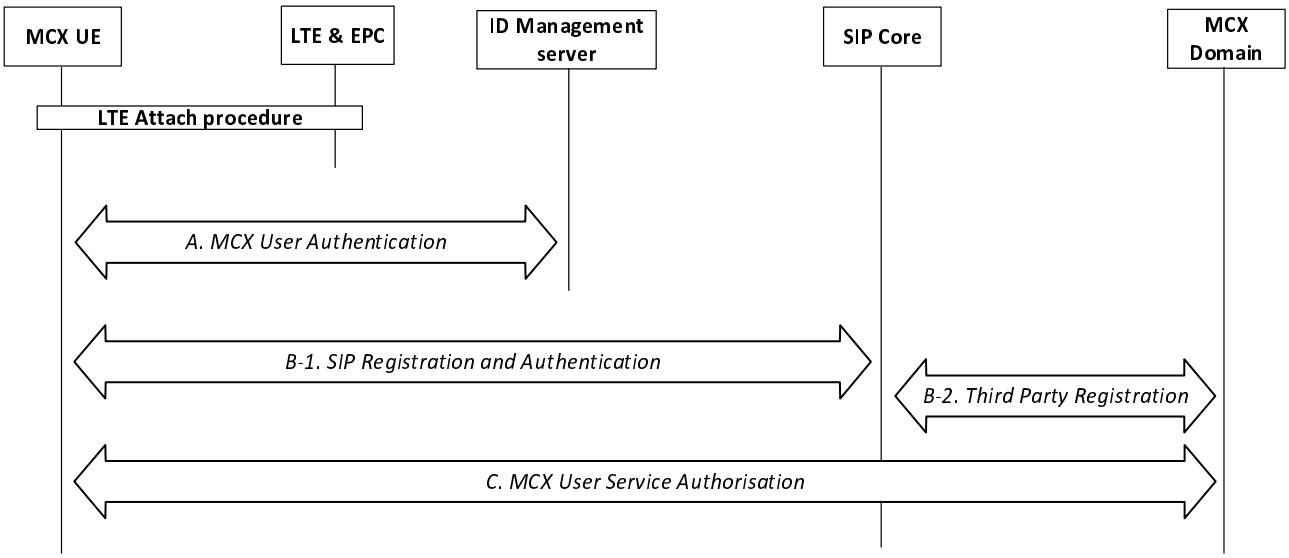


Figure 5.1.1-1: MCX authentication and authorisation

At UE power-on, the MCX UE performs LTE authentication as specified in TS 33.401 [14]. The MCX UE then performs the following steps to complete authentication of the user, authorisation of the user, MCX service registration, and identity binding between signalling layer identities and the MC service ID(s).

- A: MCX user authentication.
- B: SIP Registration and Authentication.
- C: MCX 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 MC service ID(s), a re-registration (Step B) to the SIP Core may be performed to update the registered signalling layer identity.

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

5.1.2 User authentication

5.1.2.1 Identity management functional model

The mission critical Identity Management functional model is shown in figure 5.1.2.1-1 and consists of the identity management server located in the MCX common services core and the identity management client located in the MCX UE. The IdM server and the IdM client in the MCX UE establish the foundation for MCX user authentication and user authorization.

The CSC-1 reference point, between the 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 MCX shall be implemented as defined in annex B. MCX user authentication, MCX user service authorization, OpenID Connect 1.0, and the OpenID Connect profile for MCX shall form the basis of the 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.

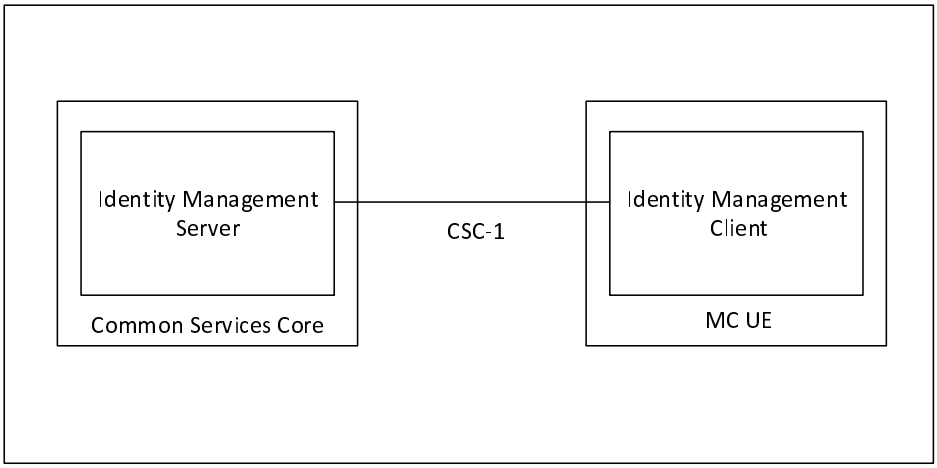


Figure 5.1.2.1-1: Functional Model for MC Identity Management

To support MCX user authentication, the IdM server (IdMS) shall be provisioned with the user's MC ID and MC service IDs (the MC service ID may be the same as the MC ID). A mapping between the MC ID and MC service ID(s) shall be created and maintained in the IdMS. When an MCX user wishes to authenticate with the MCX system, the MC ID and credentials are provided 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 MC ID and credentials, and if valid returns an ID token, refresh token, and access token to the UE IdM client specific to the credentials. The IdM client learns the user's MC service ID(s) from the ID token. Table 5.1.2.1-1 shows the MCX tokens and their usage.

Table 5.1.2.1-1: MC tokens

Token Type	Consumer of the Token	Description (See Annex B for details)
ID token	UE client(s)	Contains the MC service ID for at least one authorised service (MCPTT ID, MCVideo ID, MCData ID). Also may contain other 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 MC service ID for at least one authorised service (MCPTT ID, MCVideo ID, MCData 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 MCX user authorization, the access token(s) obtained during user authentication is used to gain MCX services for the user. MCX user service authorisation is defined in clause 5.1.3.

To support the MCX identity functional model, the MC service ID(s) 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 MCX user database and mapped to a user profile; and
- Provisioned into the GMS(s) and mapped to Group IDs.

Further details of the user authorization architecture are found in clause 5.1.3.

5.1.2.2 User authentication framework

The framework utilises the CSC-1 reference point as depicted in Figure 5.1.2.2-1.

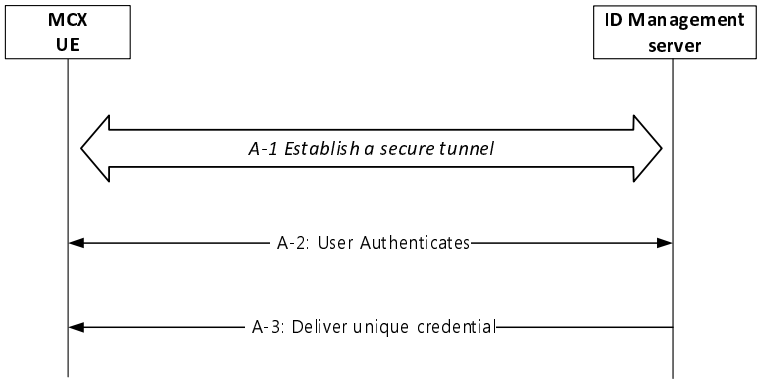


Figure 5.1.2.2-1: MCX User Authentication Framework

The User Authentication procedure in Step A of Figure 5.1.1-1 is further detailed into 3 sub steps that comprise the MCX user authentication framework:

- A-1 - Establish a secure tunnel between the MCX 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(s) that uniquely identifies the MCX user to the MCX client.

Following step A-3, the MCX client uses the credential(s) obtained from step A-3 to perform MCX user service authorization as per procedure C in figure 5.1.1-1.

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

NOTE: MCX service authorization in step C of Figure 5.1.1-1 is outside the scope of the User Authentication framework.

5.1.2.3 OpenID Connect (OIDC)

5.1.2.3.1 General

Figure 5..1.2.3.1-1 describes the MCX User Authentication Framework using the OpenID Connect protocol. Specifically, it describes the steps by which an MCX user authenticates to the Identity Management server (IdMS), resulting in a set of credentials delivered to the UE uniquely identifying the MC service ID(s). The means by which these credentials are sent from the UE to the MCX services are described in clause 5.1.3. The authentication framework supports extensible user authentication solutions based on the MCX 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.

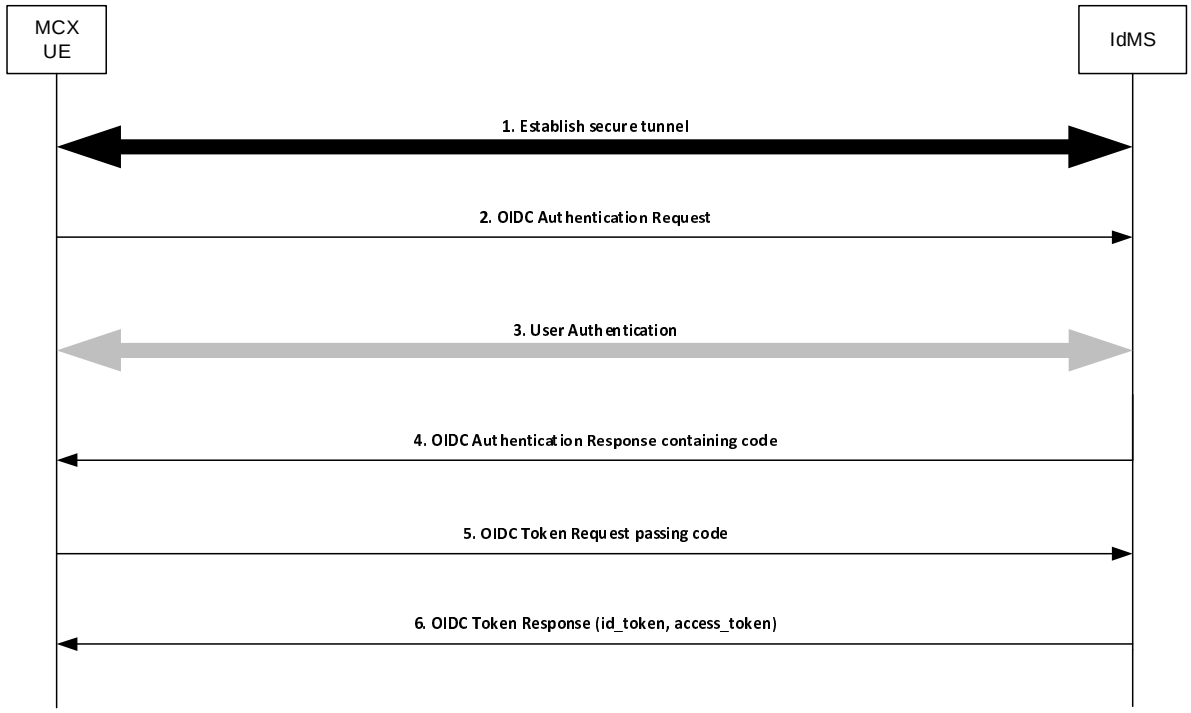


Figure 5.1.2.3.1-1: OpenID Connect (OIDC) flow supporting MCX user authentication

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 MCX service provider policy. The method chosen by the MCX service provider is neither 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 MCX service). The ID token is consumed by the UE to personalize the MCX client for the MCX user, and the access token is used by the UE to communicate the identity of the MCX user to the MCX server(s).

5.1.2.3.2 User authentication example using username/password

Figure 5.1.2.3.2-1 shows the OIDC flow when Username/Password is used as the user authentication method.

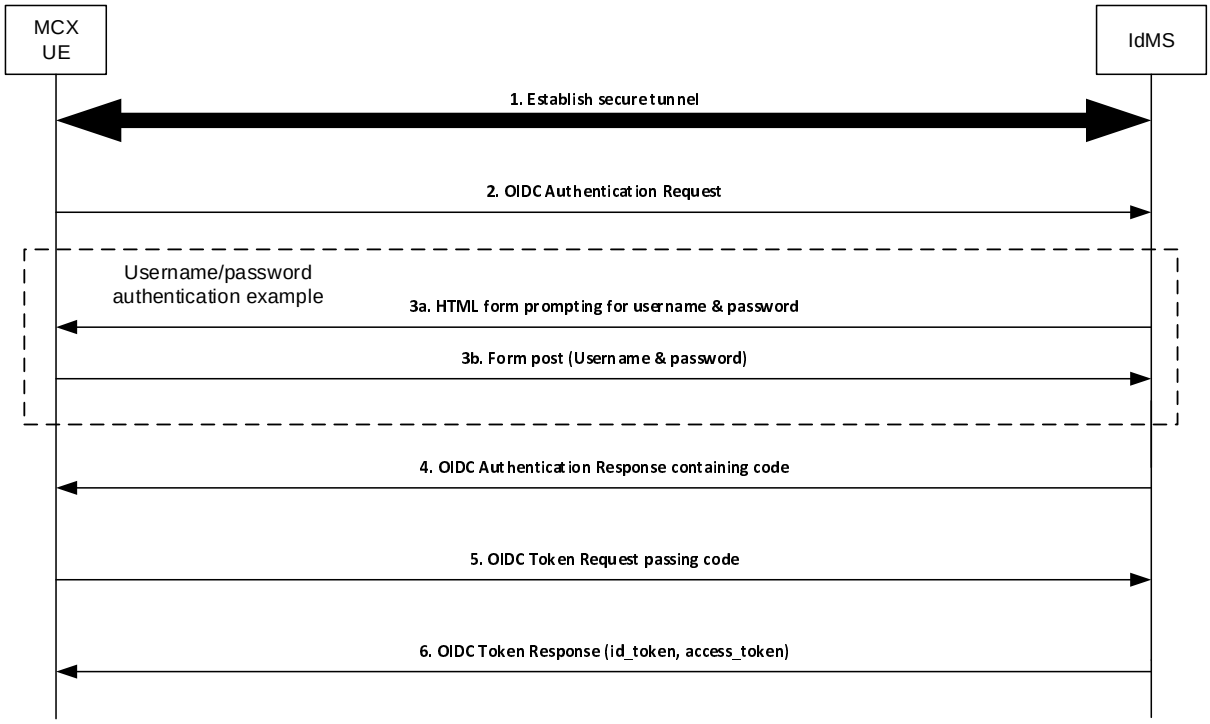


Figure 5.1.2.3.2-1: OpenID Connect (OIDC) Example Using Username/Password

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 MCX service). The ID token is consumed by the UE to personalize the MCX client for the MCX user, and the access token is used by the UE to communicate the identity of the MCX user to the MCX server(s).

5.1.3 MCX user service authorisation

5.1.3.1 General

This clause expands on the MCX user service authorization step shown in figure 5.1.1-1 step C.

MCX User Service Authorization is the function that validates whether or not a MCX user has the authority to access certain MCX services. In order to gain access to MCX services, the MCX client in the UE presents an access token

(acquired during user authentication as described in subclause 5.1.2) to each service of interest (i.e. Key Management, MCX server,Configuration Management, Group Management, etc.). If the access token is valid, then the user is granted the use of that service. Figure 5.1.3.1-1 shows the flow for user authorization which covers key management authorization, MCX 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 one or more sets of user specific key material back to the UE KM client based on the MC service ID(s) present in the access token (MCPTT ID, MCVideo ID and/or MCData ID). User specific key material includes identity based key information for media and signalling protection. This key management authorisation may be repeated for each KM service the user is authorised to use (MCPTT, MCVideo, MCData).

Editor's Note: It is ffs whether the increase in key material and keys needed per MC service ID might create storage or computational concerns for the MCX UE, the MCX servers, or both.

For MCPTT 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.

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

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

The UE can now perform configuration management authorization and download the user profile for the service(s) (MCPTT, MCVideo, MCData). Following the flow described in subclause 10.1.4.3 of 3GPP TS 23.280 [36] " MC service user obtains the MC service user profile(s) from the network ", 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 identity from the access token (MCPTT ID, MCVideo ID, MCData ID) to obtain the user profile from the MCX user database. The CM server then sends the user profile back to the CM client over HTTP. This configuration management authorisation may be repeated for each CM service the user is authorised to use (MCPTT, MCVideo, MCData).

Upon receiving each user 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 profile, and following the flow shown in clause 10.1.5.2 of 3GPP TS 23.280 [36] "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 5.7 of the present document. This group management authorisation may be repeated for each GM service the user is authorised to use (MCPTT, MCVideo, MCData).

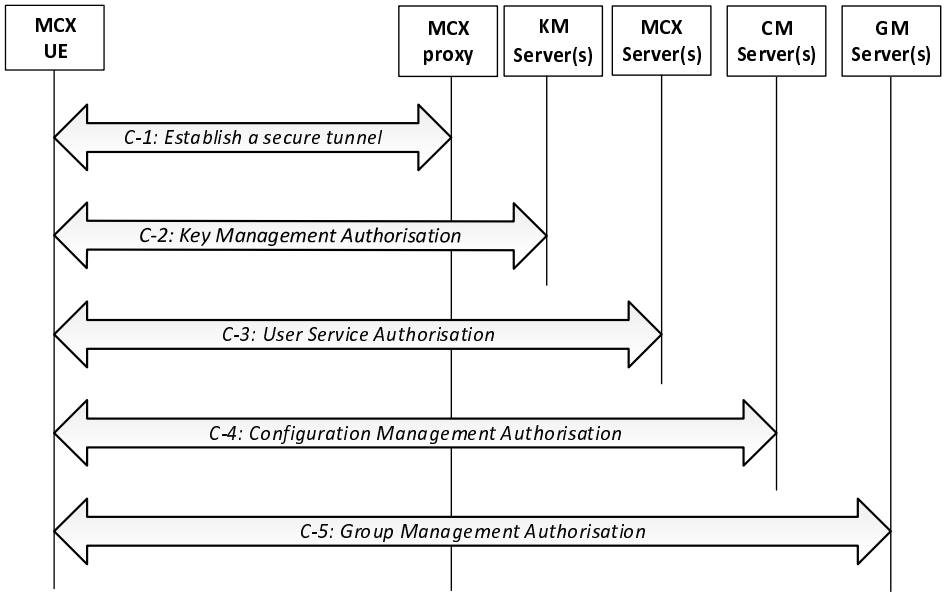


Figure 5.1.3.1-1: MCX user service authorization

The user authorization procedure in Step C of Figure 5.1.1-1 is further detailed into 5 sub steps that comprise the MCX user service authorization process:

Step C-1: If not already done, establish a secure HTTP tunnel using HTTPS between the MCX UE and MCX proxy server. Subsequent HTTP messaging makes use of this tunnel (with the possible exception of the KMS client to KMS server interface).

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 based upon the MC service ID(s) provided and replies to the client with identity specific key information. This step may be repeated to authorise the user with additional KM services (MCPTT, MCVideo, MCData) as necessary.

Step C-3: The MCX client in the UE presents an access token to the MCX server over SIP as defined in clause 5.1.3.2 of the present document. This step may be repeated to authorise the user with additional MCX services (MCPTT, MCVideo, MCData) as necessary.

Step C-4: The CM client in the UE follows the "MCX user obtains the user profile (UE initiated)" flow from clause 10.1.4.3 of 3GPP TS 23.280 [36], presenting an access token in the Get MCX 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. This step may be repeated as necessary to obtain the user profile for additional services (MCPTT, MCVideo, or MCData).

Step C-5: The GM client in the UE follows the "Retrieve group configurations at the group management client" flow as shown in clause 10.1.5.2 of 3GPP TS 23.280 [36], presenting an access token in the Get group configuration request over HTTP. If the token is valid, the GMS 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. This step may be repeated to authorise the user for additional group services (MCPTT, MCVideo, MCData) as necessary.

5.1.3.2 MCX user service authorization with MCX Server

5.1.3.2.1 General

Depending on implementation, MCX user service authorization may be performed by sending the access token to the MCX server over the SIP-1 and SIP-2 reference points using either a SIP REGISTER message or a SIP PUBLISH message. Clause 5.1.3.2.2 describes how to use the SIP REGISTER message to transport the access token to the MCX server and clause 5.1.3.2.3 describes how to use the SIP PUBLISH message to transport the access token to the MCX 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 MCX server as per Step C-3 in figure 5.1.3.1-1.

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 MCX server to re-bind an IMPU and MC service ID (MCPTT ID, MCVideo ID or MCData 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).

5.1.3.2.2 Using SIP REGISTER

The use of a SIP REGISTER message to provide the access token to the MCX server is shown in figure 5.1.3.2.2-1. The inclusion of an access token in any particular SIP REGISTER message is optional.

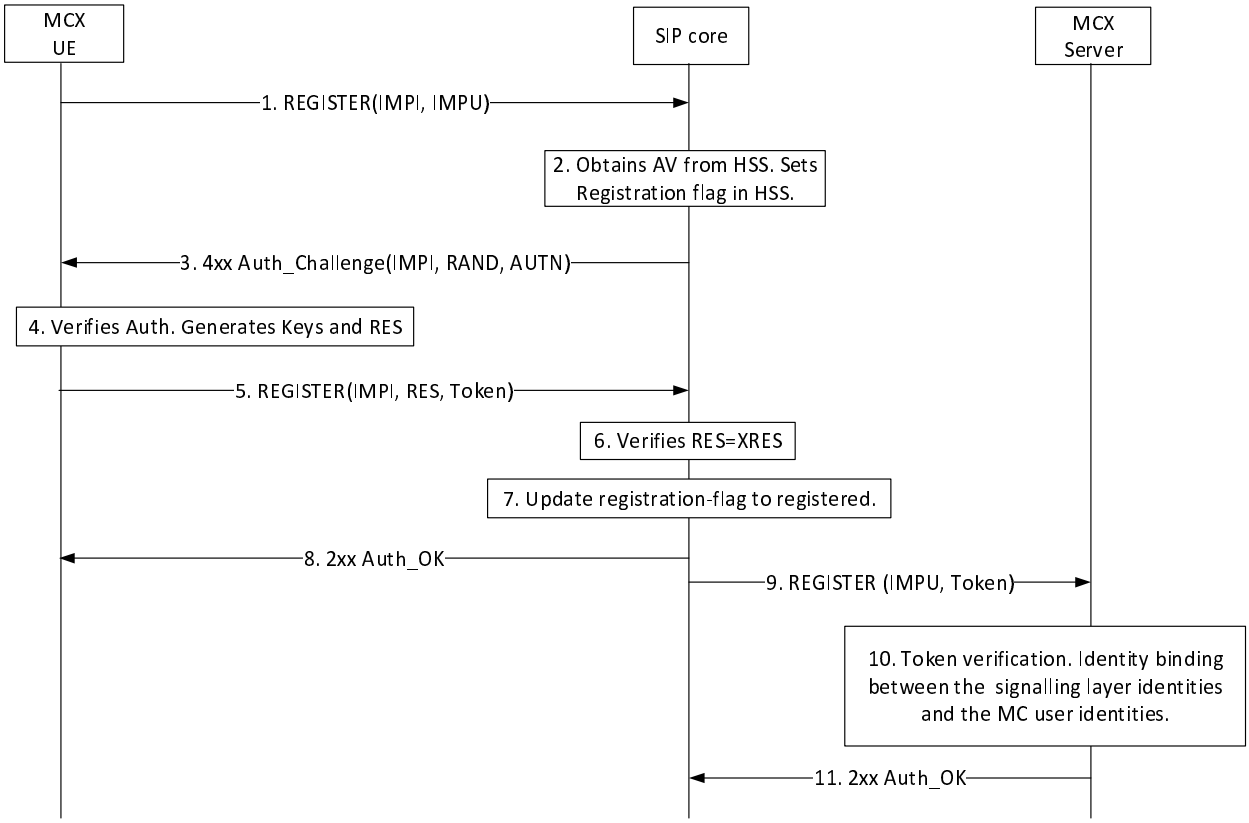


Figure 5.1.3.2.2-1: MCX User Service Authorization using SIP REGISTER message

Step 5 of figure 5.1.3.2.2-1 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 MCX server in the third part registration request message (Step 9).

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

5.1.3.2.3 Using SIP PUBLISH

The use of a SIP PUBLISH message to provide the access token to the MCX server is shown in figure 5.1.3.2.3-1. The inclusion of an access token in any particular SIP PUBLISH message is optional.

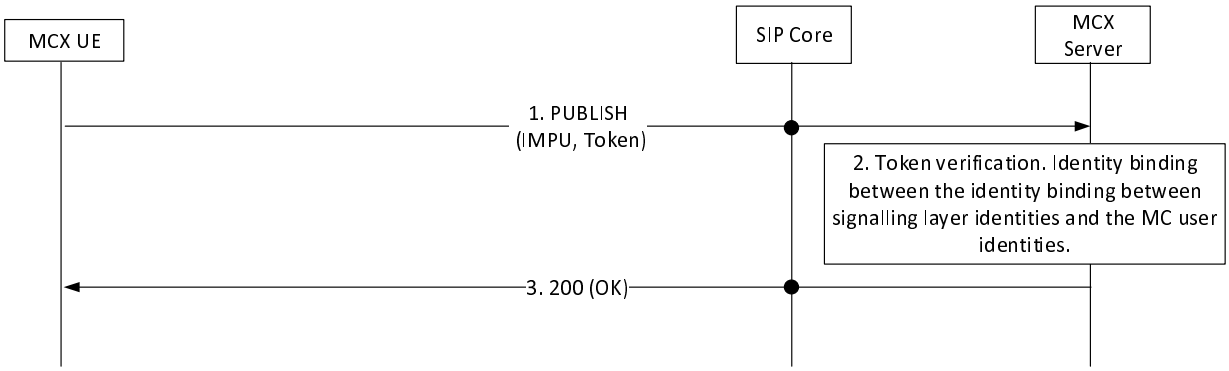


Figure 5.1.3.2.3-1: MCX User Service Authorization using SIP PUBLISH message

As shown in Step 1 of figure 5.1.3.2.3-1, the SIP PUBLISH message carries the access token through the SIP core to the MCX server.

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

5.1.4 Inter-domain MCX user service authorization

5.1.4.1 General

When a MCX User requires service authorisation to a service that is located in a domain different from the primary domain of the user, coordination between the identity management services of the primary domain and the partner domain is required. For example, a MCX User from domain A may be a member of a group that is home to domain B.

This sub-clause describes the method for authorizing a user that is home to domain A with a group service that is located in domain B.

5.1.4.2 Inter-domain identity management functional model

The inter-domain identity management functional model is shown in Figure 5.1.4.2-1.

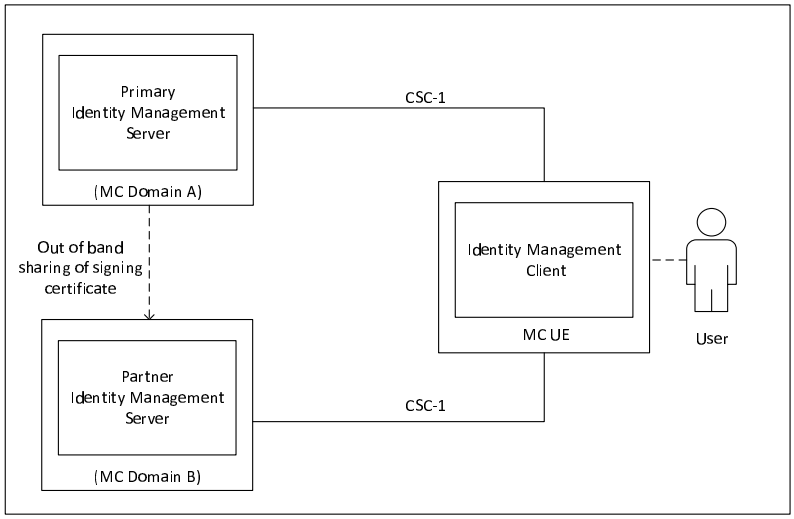


Figure 5.1.4.2-1: Functional Model for Inter-Domain MC Identity Management

In figure 5.1.4.2-1, the IdMS located in the primary domain (MCX Domain A) is the home identity management server for the user. The partner IdMS is located in a second domain (MCX Domain B) and is home to the service where the primary user requires group authorization.

The CSC-1 reference point between the UE IdM client and the partner IdM server endpoints shall be a direct connection and shall be protected with HTTPS (TLS).

The primary IdMS certificate(s) used to validate the user credentials at the partner IdMS are provisioned into the partner IdMS using an out of band mechanism beyond the scope of this document.

As defined in clause 5.1.2 an access token is required for user service authorisation. The same principle applies for inter-domain user service authorisation, in that the user must present a valid access token issued from the partner IdMS in MCX Domain B for authorisation to any group services located in MCX Domain B.

The MCX UE, after performing user service authorisation within the primary domain, may determine that the user is a member of a group service that is located in a partner domain (as indicated in the user profile).

In order for the UE to obtain this MCX Domain B access token, the token exchange procedure with the primary IdM service (MCX Domain A) shall be used to obtain a security token that identifies the user to the partner IdM service. This security token shall be specific to the partner IdM service, signed by the primary IdM service per IETF RFC 7515 [35]. Upon validation of the security token, the partner IdM service shall provide an access token to the UE specifically scoped for that user. This access token shall provide the user with authorisation to the group service(s) in the partner domain (MCX Domain B).

Figure 5.1.4.2-2 shows the token exchange and authentication procedure.

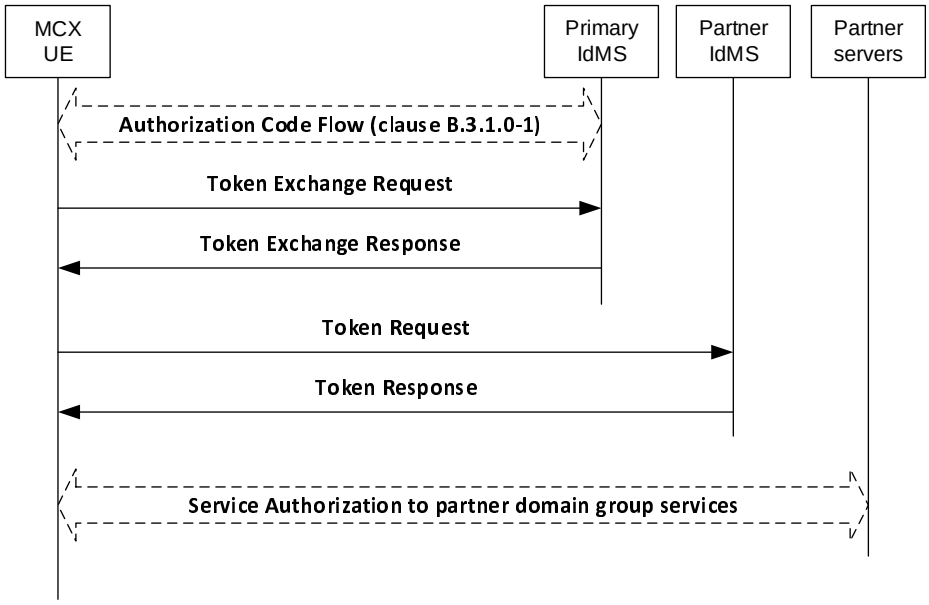


Figure 5.1.4.2-2: Token exchange procedure

The token exchange profile for accessing the partner identity management service shall consist of [45] and [46] and shall be profiled as defined in Annex B.7.

NOTE: A specific and independent security token is required for each partner identity management domain.

Once the UE obtains the access token specific to the partner group service(s), the UE shall follow the user service authorisation procedure defined in clause 5.1.3 to access the group services within the partner domain.

The token exchange procedure shall be repeated for each partner identity management domain where the UE requires access and authorisation to group service(s) within that partner domain.

Annex C.2 shows the detailed flow for inter-domain MC user service authorization using the OAuth 2.0 token exchange procedure.

5.2 Key management common elements

5.2.1 Overview of key management

This clause details the key management procedures for MCX users. It allows entities in MCX systems to establish a security association to support future communications.

The primary purpose of these procedures is to allow MCX entities to communicate with each other using end-to-end security. End-to-end security provides assurance to MCX users that no unauthorized access to communications is taking place within the MCX network. End-to-end communication security may be applied to media when operating on-network and media, floor control and media control when operating off-network.

A security domain is managed by a Key Management Server (KMS). The KMS is a component of the Common Services Core within the MCX system architecture. For any end-point to use or access end-to-end secure communications, it needs to be provisioned with key material associated to its identity by the KMS. Through the use of the KMS, MC administrators are able to manage the use of, and access to, secure communications within the MCX network.

Key provisioning for groups is performed by a Group Management Server (GMS), authorized and provisioned by the KMS. The Group Management Server is responsible for distributing the key material to MCX users within the group. This establishes a group security context. With the group security context established, MCX users can communicate using end-to-end security.

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

NOTE 1: It is a deployment option whether the MCX Server is included in the end-to-end security context. Where the MCX Server is not included in the security context, it will be unable to mix content on behalf of the users.

Key provisioning for private communications is performed by the initiating UE as the communication 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 when on-network and, when off-network, media, floor control and media 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.

The key provisioning procedures described in this specification use common security methodologies for key distribution.

5.2.2 Common key distribution

The security mechanism described in this clause allows a key, K, to be distributed from an initiating party to a receiving party. It provides confidentiality of the key, and integrity and authenticity of the payload. It is used within a number of different security procedures in this specification.

The key, K, is distributed encrypted specifically to the receiving entity and signed by the initiating entity. Prior to call commencement, both MCX UEs shall be provisioned by the KMS with time-limited key material associated with the MCX entity’s URI. The key is distributed with a 32-bit Key Identifier (K-ID). This payload is a MIKEY-SAKKE I_MESSAGE, as defined in IETF RFC 6509 [11], which ensures the confidentiality of the key, plus integrity and authenticity of the payload.

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

The payload includes the encrypted key and the key identifier (K-ID). The key is unique within the MC domain. On creating the key, the initiator generates a 32-bit key identifier (K-ID). The 4 most significant bits of the K-ID shall indicate the purpose of the key, the other 28-bits shall be randomly generated. The key identifier (K-ID) is stored in the CSB-ID field of the MIKEY I_MESSAGE.

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

NOTE: This solution is for the end-to-end protection of keys and does not protect the identities transmitted. Identities may be masked by transmitting the UID within the MIKEY ID fields as described in Annex E.7.

The security processes are summarized in figure 5.2.2-1.

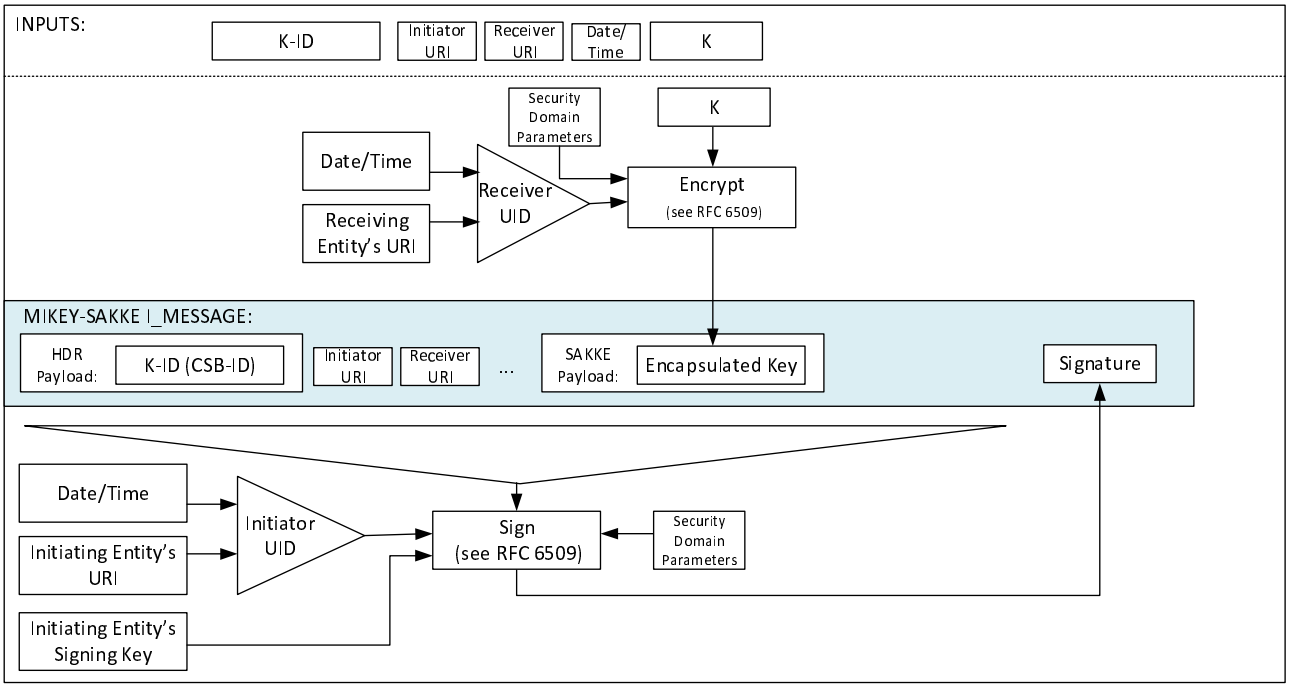


Figure 5.2.2-1: Common key distribution mechanism

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

At the receiving MCX entity, the initiating entity'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 key, K, using the (KMS-provisioned) entity's UID key. The MCX entity also extracts the K-ID. This process is shown in figure 5.2.2-2.

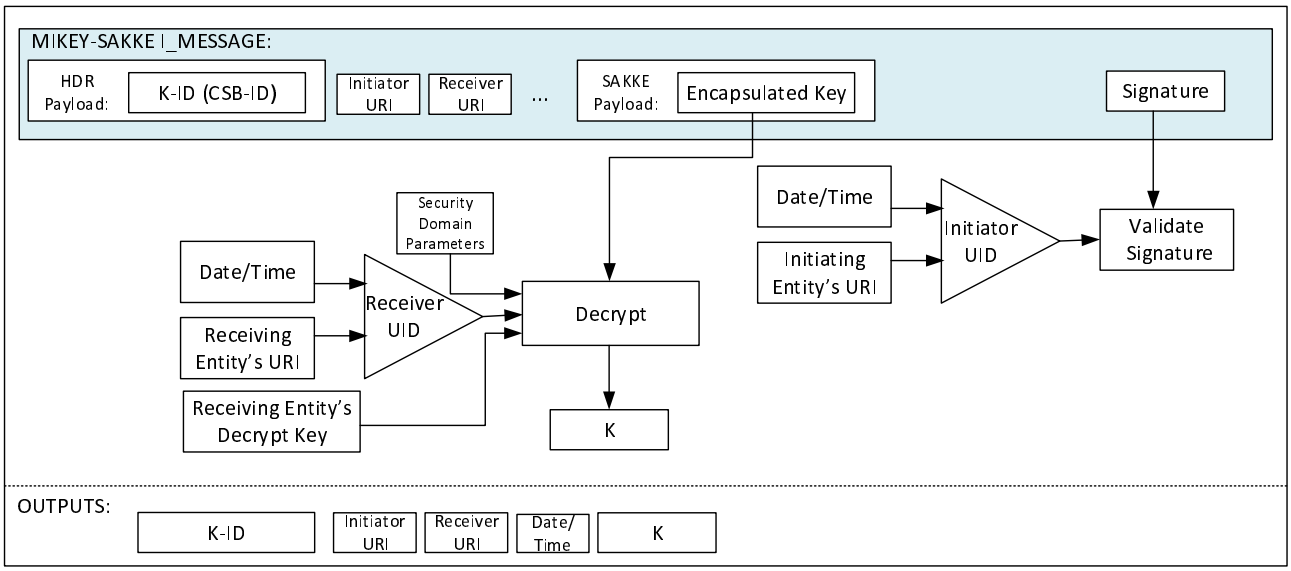


Figure 5.2.2-2: Common key extraction mechanism

With the key successfully shared between the two MCX entities, the entities are able to use the shared security context to protect communications.

5.2.3 Key distribution with end-point diversity

The security mechanism described in this clause extends that defined in clause 5.2.2 to provide end-point key diversity. The mechanism is identical to that described in clause 5.2.2, except for the distribution of K-ID. Contrary to clause

5.2.2, the key is distributed with an end-point-specific key identity (UK-ID) derived from the key id (K-ID). This allows the receiving entity to diversify the shared key for end-point-specific use.

Specific types of key require use of end-point key diversity. The type of key is defined by the 'purpose tag' within the key identifier stored in the CSB-ID field of the MIKEY payload. Hence on receipt of a key, the contents of the CSB-ID field instruct the receiving entity whether end-point diversity should be applied to the key.

The key, K, is distributed encrypted specifically to the receiving entity and signed by the initiating entity as described in clause 5.2.2. The key is distributed with a 32-bit entity-specific Key Identifier (UK-ID) derived from a common key id (K-ID) and a salt (which is derived from the receiving entity's MCX URI).

The payload includes the entity-specific Key Identifier (UK-ID) within the CSB-ID field. The key, K, is identified by a Key Identifier (K-ID) from which the UK-ID is derived. On creating the key, K, the initiating entity generates a K-ID as follows. The 4 most significant bits of the K-ID is the 'purpose tag' which defines the purpose of the key. The 28 least significant bits of the K-ID is a 28-bit randomly-generated value.

For each receiving entity, the initiating entity creates a 28-bit Salt by hashing the receiving entity's URI through a KDF using the key, K, as the key (as defined in Annex F.1.3). The Salt is xor'd with the 28 least-significant bits of the K-ID to create the 32-bit UK-ID.

NOTE: Knowledge of the UK-ID, K-ID and Salt does not reveal the receiving entity URI to those without the key K.

The process for generating the UK-ID is summarized in figure 5.2.3-1.

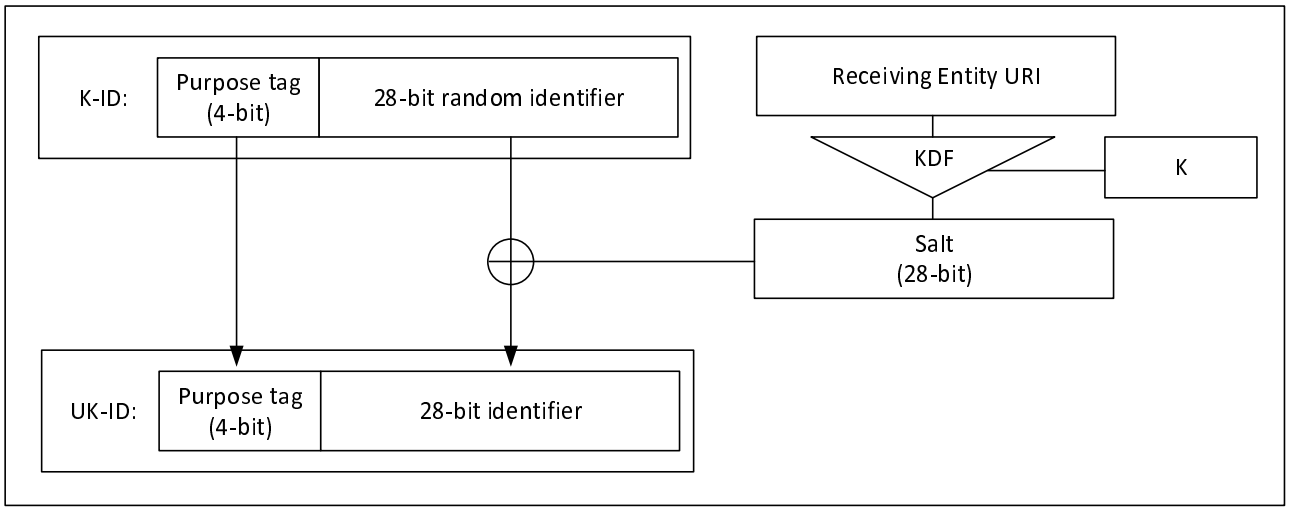


Figure 5.2.3-1: Generating the UK-ID

The UK-ID is placed in the CSB ID field within the header of the I_MESSAGE. The security processes are summarized in figure 5.2.3-2.

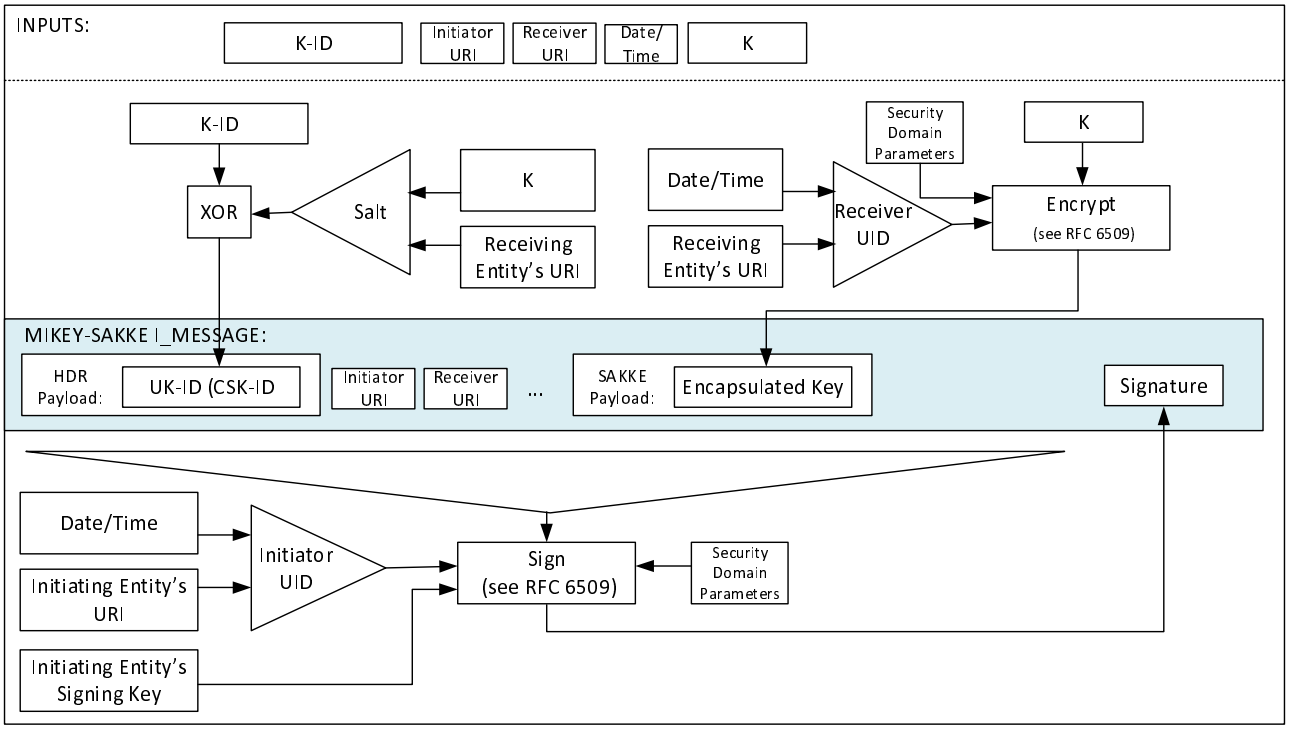


Figure 5.2.3-2: Common key distribution mechanism with end-point diversity

At the receiving MCX entity, the initiating entity'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 payload. If valid, the receiving entity extracts and decrypts the encapsulated key, K, using the (KMS-provisioned) entity's UID key.

The receiving MCX entity also extracts UK-ID from the CSB-ID field of the I_MESSAGE. If the 'purpose tag' of the UK-ID indicates that end-point diversity is applied, the receiving entity generates the Salt using its URI and the decrypted key, K. The receiving entity xors the UK-ID and Salt together to obtain the K-ID. The K-ID and UK-ID are stored

The extraction procedure is described in figure 5.2.3-3.

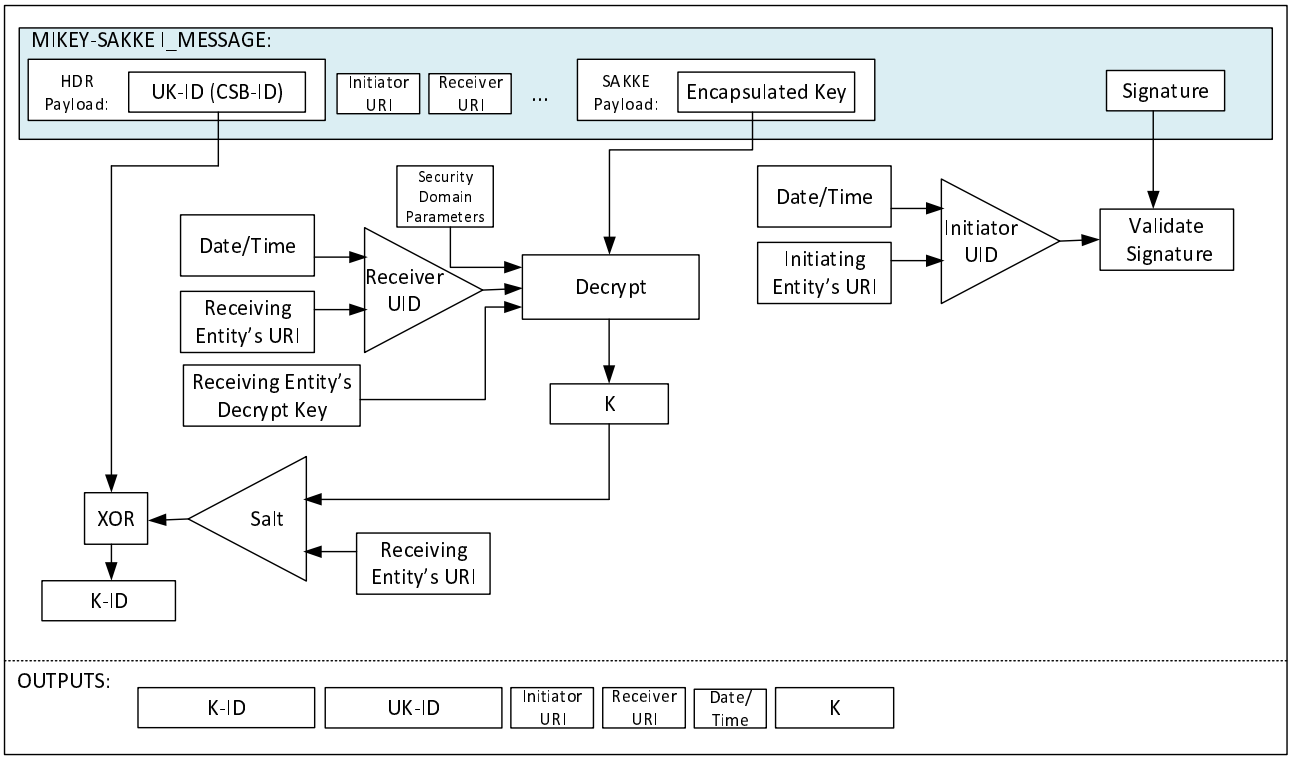


Figure 5.2.3-3: Common key extraction mechanism with end-point diversity

5.2.4 Key distribution with associated parameters

The key distribution mechanisms described in Clause 5.2.2 and clause 5.2.3 may be extended to include data associated with the key in the MIKEY I_MESSAGE. This data is stored within a format known as 'associated parameters' and defined in Annex E.6.

The associated parameters are encrypted using K, the key distributed within the MIKEY I_MESSAGE. The security mechanism is summarised in Figure 5.2.4-1.

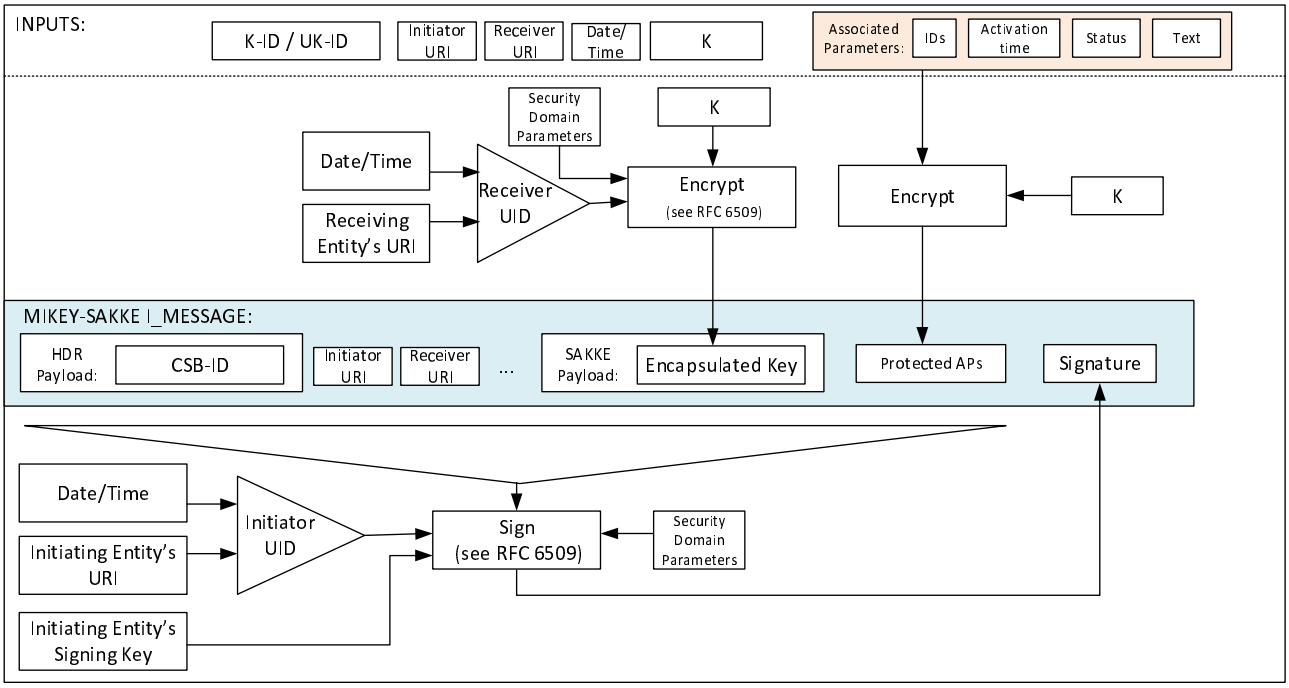


Figure 5.2.4-1: Common key distribution mechanism with associated parameters

At the receiving MCX entity, the initiating entity'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 payload. If valid, the receiving entity extracts and decrypts the encapsulated key, K, using the (KMS-provisioned) receiving entity's decryption key.

The receiving MCX entity also extracts 'associated parameters' payload from the I_MESSAGE. The receiving entity uses the decrypted key, K, to decrypt these associated parameters. The receiving entity stores these parameters with the distributed key, K. If the Status field within the ‘associated parameters’ payload indicates the key has been revoked, the distributed key, K, and the K-ID shall not be used. If the decryption process for the encapsulated associated parameters fails, the key is rejected.

The security mechanism is summarised in Figure 5.2.4-2.

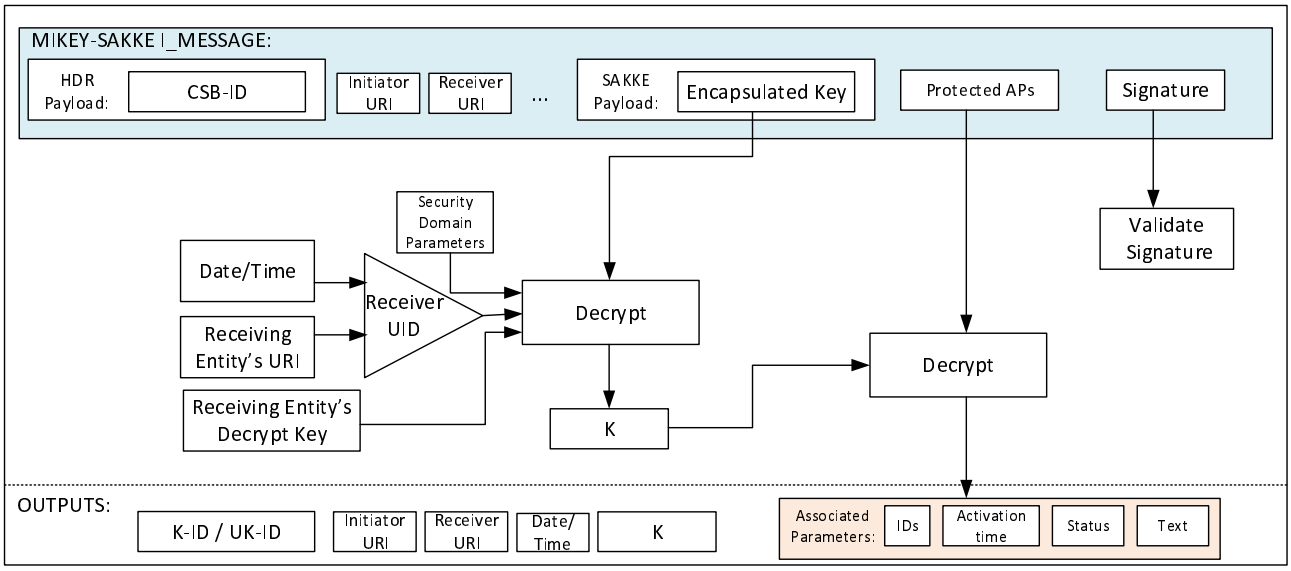


Figure 5.2.4-2: Common key extraction mechanism with associated parameters

5.2.5 Key distribution with SAKKE-to-self payload

The key distribution mechanism defined in clauses 5.2.2, 5.2.3 and 5.2.4 may be extended to allow the initiating entity to be able to decrypt the distributed key, K contained within the payload.

NOTE: Where the initiating entity is an MCX user logged into multiple devices, this extension is necessary to allow all devices to obtain the key, K and decrypt any subsequent communication.

In addition to encrypting the key, K, to the receiving entity, the key is also encrypted to the initiating entity. The UID used to encrypt the data is derived from the initiating entity's URI (e.g. user.002@mcptt.example.org) and a time-related parameter (e.g. the current year and month). The encapsulated key is added to a SAKKE-to-self payload within the MIKEY I_MESSAGE. No other payloads (e.g. IDRr) are affected.

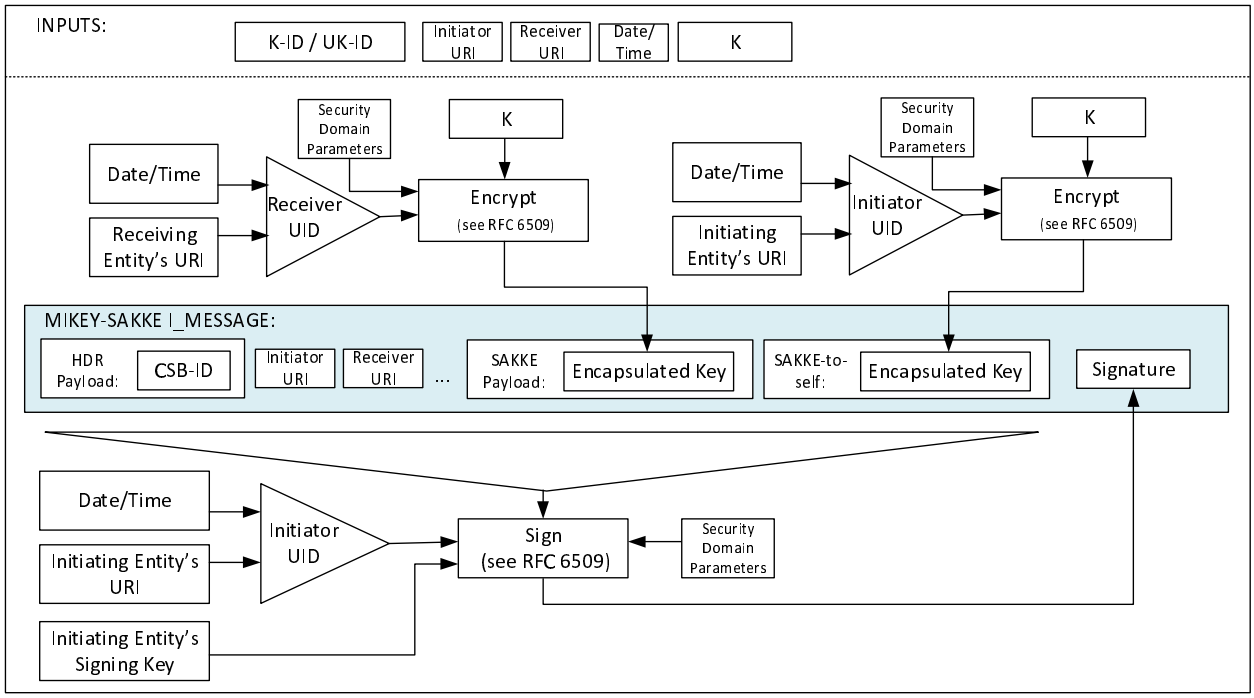


Figure 5.2.4-1: Common key distribution mechanism with SAKKE-to-self payload

5.2.6 Key distribution with identity hiding

The key distribution mechanism defined in clauses 5.2.2, 5.2.3, 5.2.4 and 5.2.5 may be extended to allow identities to be masked within the MIKEY payload. This is achieved by adding the UID, rather than the URI to the payload as described in Annex E.7 and shown in figure 5.2.6-1.

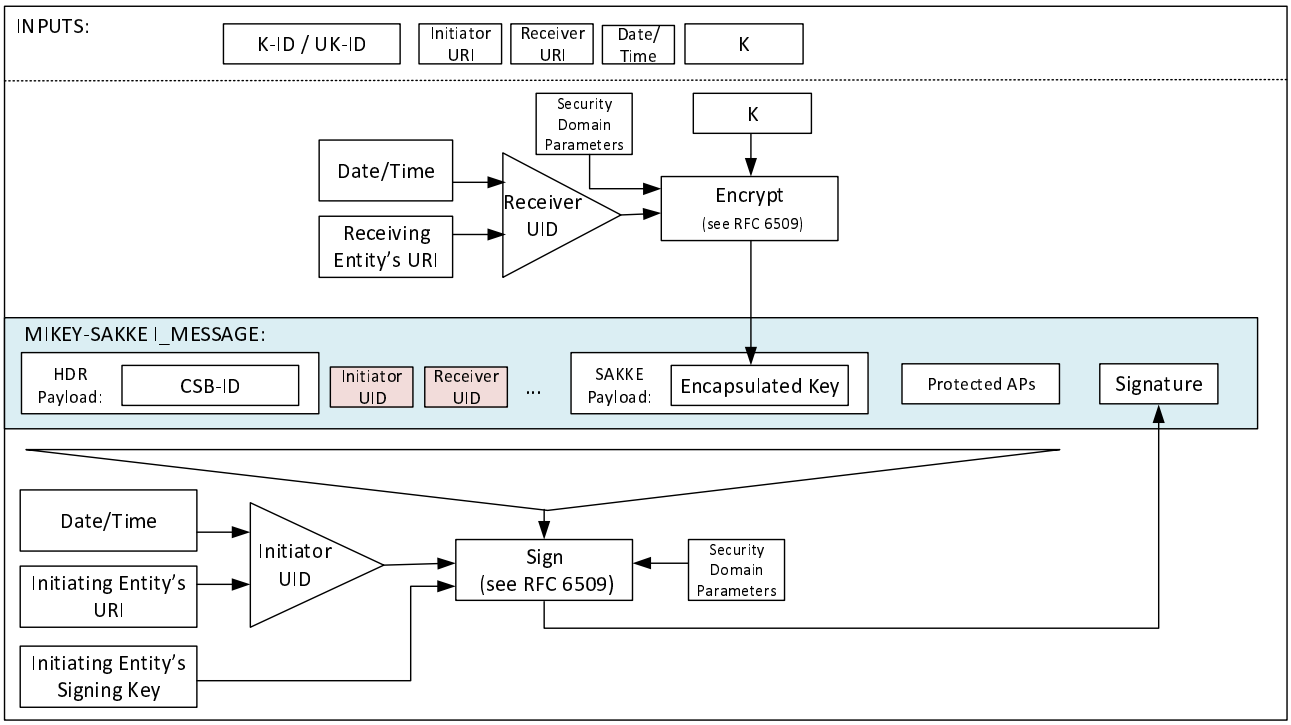


Figure 5.2.6-1: Common key distribution mechanism with identity hiding

On receipt of a MIKEY payload with identities hidden, the receiving entity should recognise the receiver UID in the packet. If not, the I_MESSAGE shall be rejected. Based on the initiator UID, the receiver checks the validity of the

I_MESSAGE signature. At this point the initiator is anonymous to the receiver. If this check fails, the I_MESSAGE shall be rejected. The receiver then extracts the key K. This may be used to decrypt other parts of the packet and extract the initiator URI. Once the initiator URI is extracted, this shall be used to generate the initiator UID and check that it is the one provided in the I_MESSAGE. If not, the I_MESSAGE shall be rejected. This procedure is shown in figure 5.2.6-2

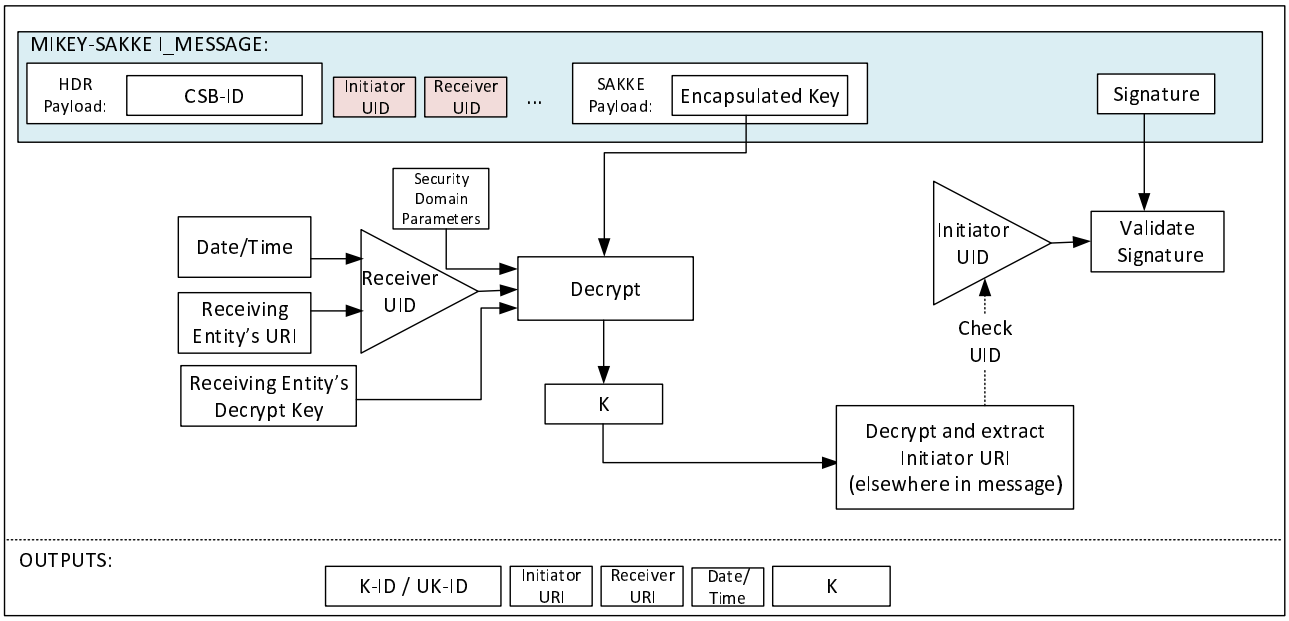


Figure 5.2.6-2: Common key extraction mechanism with identity hiding

5.2.7 Key distribution across multiple security domains

5.2.7.1 General

5.2.7.2 Identification of External Security Domains

To support multiple security domains, the security domain used by each user is recorded alongside the user's MC Service ID within configuration parameters in the MC system. Furthermore, the security domain of the GMS is recorded alongside the GMS FQDN and the security domain of the MCX Server is recorded alongside the MCX Server FQDN. Security domains are identified by a unique identifier, the 'KMSUri'. Specifically, the following describes the situations where security domain information is needed:

1) The MCX Server(s) requires knowledge of the security domain (KMSUri) of users connected to the server.

2.1) On initiating a MCPTT private call, the initiating UE requires knowledge of the security domain (KMSUri) of the receiving user.

2.2) On receiving a MCPTT private call, the receiving UE requires knowledge of the security domain (KMSUri) of the initiating user.

3.1) On initiating a MCVideo private call, the initiating UE requires knowledge of the security domain (KMSUri) of the receiving user.

3.2) On receiving a MCVideo private call, the receiving UE requires knowledge of the security domain (KMSUri) of the initiating user.

4.1) On initiating a MCData one-to-one SDS or file transfer, the initiating UE requires knowledge of the security domain (KMSUri) of the receiving user.

4.2) On receiving a MCData one-to-one SDS or file transfer, the receiving UE requires knowledge of the security domain (KMSUri) of the initiating user.

5) The Group Management Server requires knowledge of the security domain (KMSUri) of each member of the group.

6) Group members require knowledge of the security domain (KMSUri) of the group management server.

7) MC users require knowledge of the security domain (KMSUri) of the MCX Server(s) to which they connect.

NOTE: In most cases, the required security domain will be the Home security domain, meaning that the required KMSUri will be the user's Home KMSUri. It may be more space efficient to only keep a record where the KMSUri is not the Home KMSUri.

5.2.7.3 Using multiple security domains

On encrypting to an entity within the MC System using an I_MESSAGE, the client shall lookup the KMSUri from the appropriate configuration data, then lookup the appropriate KMS Certificate with that KMSUri from the certificate cache downloaded from it's home KMS. The security parameters within the KMS Certificate are used to perform encryption. The KMSUri is added to the I_MESSAGE within the IDRkmsr field.

Equivalently, when verifying a received I_MESSAGE, the receiving client shall extract the KMSUri from the I_MESSAGE (if present) and check this matches the KMSUri from the appropriate configuration data. The client shall then lookup the appropriate KMS Certificate with that KMSUri from the certificate cache downloaded from it's home KMS. The security parameters within the KMS Certificate are used to perform verification.

Should a matching certificate not be found, the client may request the certificate based on the KmsUri from it's home KMS using an appropriate KMSCertCache request.

5.3 User key management (KMS)

5.3.1 General

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

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

5.3.2 Functional model for key management

5.3Within the mission critical architecture, the Key Management Server (KMS) provisions key material associated with a specific MC identity (e.g. MCPTT ID). The 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 MC UE, in the MCX Server(s) and in the Group Management Server(s).

The reference points for the KMS are shown in figure 5.3.2-1.

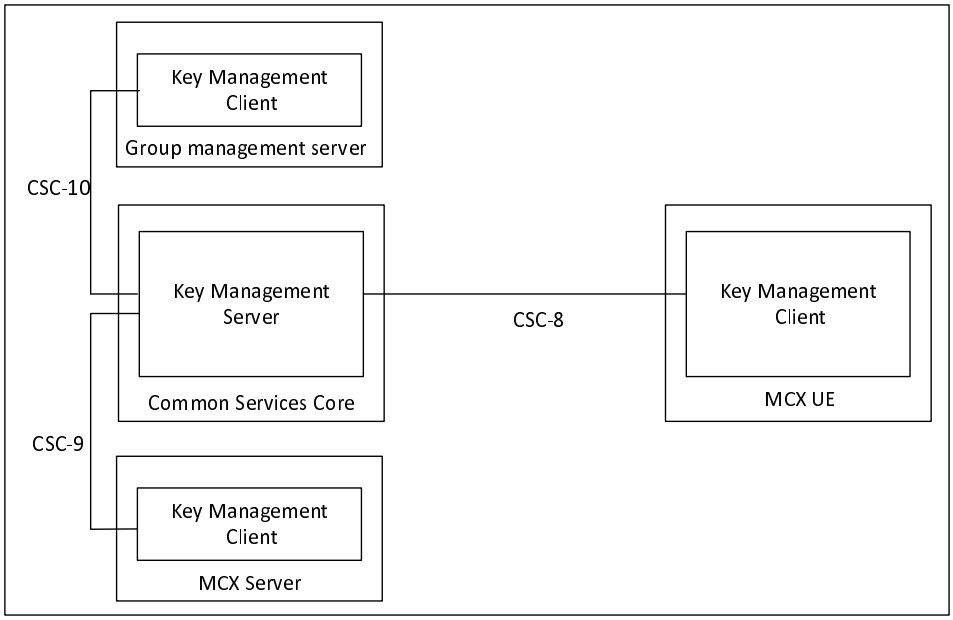


Figure 5.3.2-1: Reference Points for Key Management Server

Figure 5.3.2.1-1 shows the CSC-8, CSC-9 and CSC-10 reference points for the Key Management Server within a MC domain.

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

If the KMS does not make use of the HTTP proxy, then a secure HTTP connection (HTTPS) shall be established directly between the KMS server and the KMS client. In this case, each of CSC-8, CSC-9 and CSC-10 is a direct HTTP connection between the KMS Server and KMS client in the MC UE, MCX Server or GMS (resp). The use of the TrK as defined in clause 9.3.3 may be used to protect the key material content in this configuration.

If the KMS does connect to and employ the use of the HTTP proxy, then for public safety users the TrK shall be used as defined in clause 9.3.3 to protect the key material content. In this case, each of CSC-8, CSC-9 and CSC-10 uses HTTP-1 and HTTP-2 between the KMS Server and KMS client in the MC UE, MCX Server or GMS (resp).

5.3.3 Security procedures for key management

The procedure for the provision of identity-specific key material when the HTTP proxy is supported between the KMS and the KMS client is described in figure 5.3.3-1. The procedure is the same whether the key management client in the MC UE, an MCX Server or a Group Management Server is making the request.

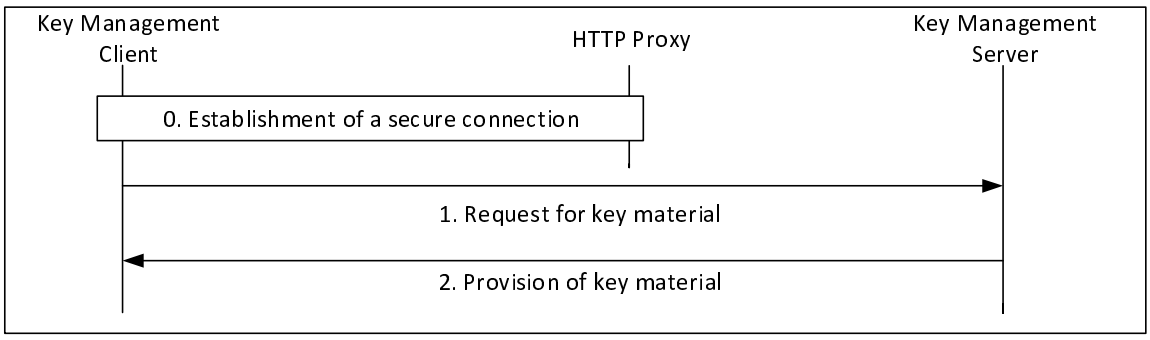


Figure 5.3.3-1: Provisioning of key material via the HTTP proxy

The procedure in figure 5.3.3-1 is now described step-by-step.

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

NOTE: Additionally, the connection between the KMS and the HTTP Proxy is secured according to clause 6.1.

1) The key management client makes a request for user key material from the KMS. The request contains an access token to authenticate the user as defined in clause 5.1. There are three types of request (as defined in Annex D):

   a) KMSInit Request. This request is the first request sent to the KMS to setup the user.

   b) KMSKeyProv Request: This request is to obtain new key material from the KMS. The request may contain details of a specific identity (e.g. MCPTT ID) required for key management, and may contain a specific time for which the key material is required.

   c) KMSCertCache Request: This request is to obtain external KMS certificates associated with external security domains (managed by another KMS). The request may contain details of the latest version of the cache received by the client.

2) The KMS provides a response based upon the authenticated user and the user's request. For public safety use, the key material itself shall be encrypted using a 256-bit transport key (TrK). The response may also be signed by the TrK. The TrK is distributed via an out-of-band mechanism along with a 32-bit identifier, TrK-ID. The responses are:

   a) KMSInit Response. This response contains domain parameters and, optionally, a new TrK.

   b) KMSKeyProv Response: This response provides new key material to the user and, optionally, a new TrK.

   c) KMSCertCache Response: This response contains new or updated home KMS certificates and/or external KMS certificates required by the user for communications with external security domains.

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 5.3.3-2.

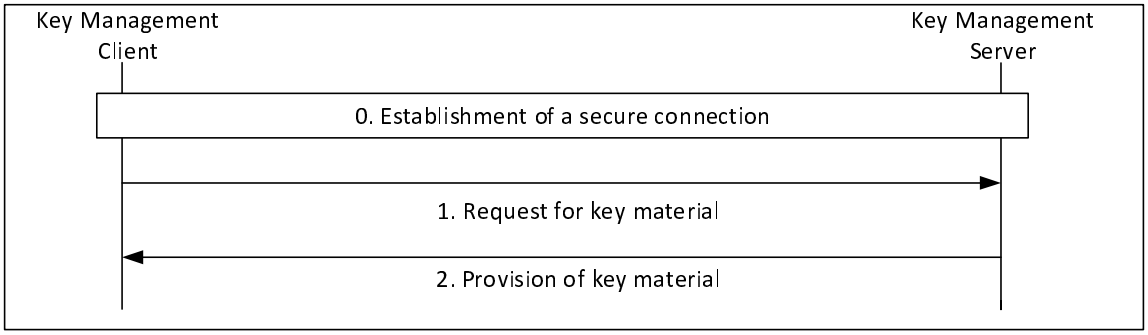


Figure 5.3.3-2: Provisioning of key material without a proxy

The procedure in Figure 5.3.3-2 is now described step-by-step:

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

1) The key management client makes a request to the KMS. The same requests can be made as defined above with a proxy.

2) The KMS provides a response based upon the authenticated user and the user's request. Optionally, the key material itself may also be encrypted using a 256-bit transport key (TrK). The response may also be signed using the TrK. The 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 MC system.

5.3.4 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 provisioned to each user is listed below:

- A KMSInit Response contains the Home KMS Certificate (domain specific key material associated to the KMS), and may contain:
  - An updated TrK for the user (to replace the offline-provisioned, bootstrap TrK).
- A KMSKeyProv Response contains zero, or more, KMSKeySets and may contain:
  - An updated TrK for the user (to replace existing TrK).
- A KMSCertCache Response may contain:
  - Home KMS Certificate(s) (current, updated or future).
  - External KMS Certificates. This is domain specific key material associated with other KMSs. It is required to enable secure communications across security domains.

5.3.5 KMS Certificate

A KMS Certificate is defined in Annex D.3.2. A KMS Certificate contains the following:

- A Role of 'Home' or 'External', depending on whether the certificate is the issuing KMS's or is provided by another external KMS.
- The KMS Public Authentication Key (KPAK in IETF RFC 6507 [9]).
- The 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.

Certificates are identified by the KMS (KMSUri) and a unique identifier (CertUri). A (logical) KMS should only have a single KMS certificate active at any one time (based upon the KMSUri). Certificates may be updated using the CertURI. Should a client receive a certificate with a CertURI of an existing certificate, the client shall replace this existing certificate with the newly provisioned certificate.

The UID conversion mechanism defines how UIDs are generated. Using this information a MC 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 MC Service IDs and UIDs during each time period.

5.3.6 KMS provisioned Key Set

KMSKeySet(s) contain the following:

- A user signing key for each UID for the current time period (SSK and PVT in IETF RFC 6507 [9]).
- A user decryption key for each UID for the current time period (RSK in IETF RFC 6508 [10]).
- The key period number associated with the current keys.
- Optionally, the time period, for which the user key material is valid (e.g. month).

5.4 Key management from MC client to MC server (CSK upload)

The key (CSK) is distributed from the MCX client to the MCX Server(s) using the ‘CSK upload’ procedure. The procedure shall use the common key distribution mechanism described in clause 5.2.2, transported over the SIP bearer. Identity hiding may be supported as defined in clause 5.2.6.

The initiating entity of the CSK upload procedure shall be the MCX UE and the receiving entity shall be the MCX Server. With respect to the common key distribution procedure, the initiating entity URI shall be the MCX Service user ID of the user andthe receiving entity URI shall be the MCX Server Domain Security Identifier (MDSI). The MDSI is added to the recipient field (IDRr) of the message. The distributed key, K, shall be the CSK and the distributed identifier K-ID shall be the CSK-ID.

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

Before the CSK upload procedure can be used by the client to securely share the encryption key, the MC user shall first be authorized by KMS for key management services. Once the MC user is authorized, the KMS distributes the user's key material to the client as specified in clause 5.3.3.

The server receives the SIP message with the protected CSK and retrieves it from the message. It associates the MC User's SIP Core identity (IMPU), MC Service user ID (e.g. 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 a MC domain.

5.5 Key management between MCX servers (SPK)

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

To protect information from all other entities outside of the MC 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 Annex G.

The SPK and associated values shall be directly provisioned into the communicating servers, along with the SPK-ID. With the SPK provisioned, RTCP and XML content (within SIP) may be protected.

5.6 Key management for one-to-one (private) communications (PCK)

The purpose of this procedure is to allow two MCP UEs to create an end-to-end security context to protect an MCX private communication. To create the security context, the initiating MCX UE generates a Private Communication Key (PCK) and securely transfers this key, along with a key identifier (PCK-ID), to the terminating MCX UE. Prior to key distribution, both MCX UE shall be provisioned by the Key Management Server (KMS) with time-limited key material associated with the MCX user as described in clause 5.3.

The PCK is distributed between the MCX clients using the security mechanism described in clause 5.2.2, transported over the SIP bearer within the SDP content of a SIP INVITE. The SAKKE-to-self extension may be included as defined in clause 5.2.5. Identity hiding may be supported as defined in clause 5.2.6.

The initiating entity shall be the initiating MCX user. The initiating entity URI shall be the MCX ID of the initiating user. The receiving entity shall be the terminating MCX user. The receiving entity URI shall be the MCX ID of the terminating user. The distributed key, K, shall be the PCK and the distributed identifier K-ID shall be the PCK-ID.

Clause E.2 provides MIKEY message structure for PCK distribution.

5.7 Key management for group communications (GMK)

5.7.1 General

To create the group's security association, a Group Master Key (GMK) and associated identifier (GMK-ID) is distributed to MCX 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 MCX UE within the group shall be provisioned by the MCX Key Management Server (KMS) with time-limited key material associated with the MCX user as described in clause 5.3. The Group Management Server shall also be provisioned by the MCX KMS with key material for an identity which is authorized to create groups.

The GMK is distributed from the GMS to a MCX client using the security mechanism described in clause 5.2.2, transported over the SIP bearer. For GMKs, end-point diversity is required and hence the extension in clause 5.2.3 is applied. Additional parameters may be included as defined in clause 5.2.4. The SAKKE-to-self extension may be included as defined in clause 5.2.5. Identity hiding may be supported as defined in clause 5.2.6.

The initiating entity shall be the initiating GMS. The initiating entity URI shall be the URI of the GMS (e.g. gp.manager@mcptt.example.org). The receiving entity shall be the terminating MCX user. The receiving entity URI shall be the MCX ID of the terminating user. The distributed key, K, shall be the GMK, the key identifier K-ID shall be the GMK-ID and the end-point-specific key identifier, UK-ID shall be the GUK-ID.

Clause E.3 provides MIKEY message structure for GMK distribution.

5.7.2 Security procedures for GMK provisioning

This procedure uses a MIKEY payload to distribute a GMK from the GMS to the MC UEs within the group. The payload is transported as part of the 'Notify group configuration request' message defined in clause 10.1.2.7 of 3GPP TS 23.280 [36].

Figure 5.7.2-1 shows the security procedures for creating a security association for a group.

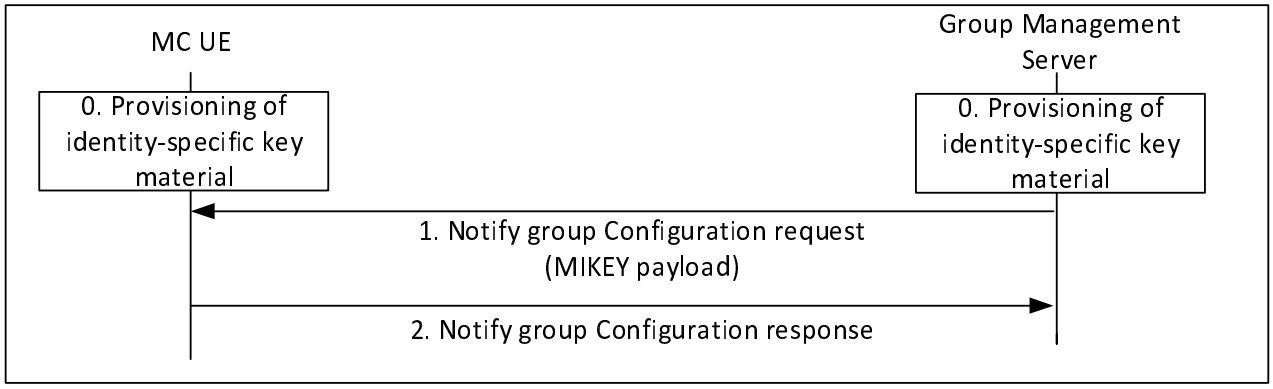


Figure 5.7.2-1: Security configuration for groups

A description of the procedures depicted in figure 5.7.2-1 follows. For clarity, figure 10.1.5.3-2 in clause 10.1.5.3 of 3GPP TS 23.280 [36] is referenced.

0) Prior to beginning this procedure the MC client shall be provisioned with identity-specific key material by a MC KMS as described in clause 5.3. 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 MC 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 (MCX service user ID) associated to the MC 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 5.2.4-1.

2) On receipt of a MIKEY message, the MC 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 MC 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 5.2.4-2. Should any of these checks fail, an error shall be returned to the GMS. Upon successful receipt and processing, the MC UE shall store the GMK, GMK-ID and GUK-ID and respond to the GMS with a 'Notify group configuration response' message.

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.

5.8 Key management from MC server to MC client (Key download)

5.8.1 General

The 'key download' procedure is used to send keys from the MCX server to the MC client. It is used to distribute Multicast Signalling Keys (MuSiKs) to the MC clients, and it is used to update both the CSKs and MuSiKs.

Within the 'key download' procedure, keys (CSK or MuSiKs) are encrypted specifically to the MC user and signed using an identity representing the MC Server. Prior to group key distribution, each MC client shall be provisioned by the KMS with time-limited key material associated with the MC User as described clause 5.3. The MC Server shall also be provisioned by the MC KMS with key material for an identity which is authorised to act as an MCX Server.

The key (CSK or MuSiK) is distributed from the MCX Server to a MC client using the security mechanism described in clause 5.2.2, transported over the SIP bearer. End-point diversity is not required as end-points do not encrypt data, hence the extension in clause 5.2.3 is not applied. Additional parameters may be included as defined in clause 5.2.4. The SAKKE-to-self extension may be included as defined in clause 5.2.5. Identity hiding may be supported as defined in clause 5.2.6.

The initiating entity shall be the initiating MCX Server and the receiving entity shall be the terminating MC user. The initiating entity URI shall be the FQDN of the MCX Server (e.g. MDSI of the MC Domain) and the receiving entity URI shall be the MC Service ID of the terminating user. The distributed key, K, shall be the CSK or MuSiK and the key identifier K-ID shall be the CSK-ID or MuSiK-ID (respectively).

As a result of this 'key download' mechanism, the MC clients receive a new signalling key, CSK or MuSiK, identified by the 4 most significant bits of the key ID.

5.8.2 'Key download' procedure

The procedure for key download is described in figure 5.8.2-1:

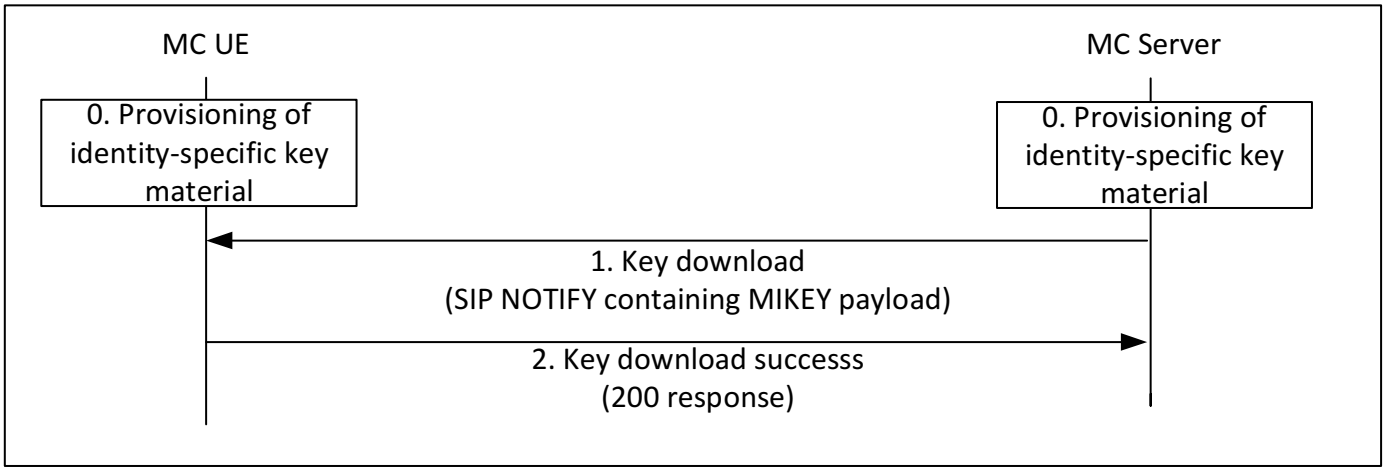


Figure 5.8.2-1: Procedures for key download

0. The MCX UE has been provisioned by a KMS with key material associated with the MC user. The MC UE has also registered with an MCX Server. As a consequence of this registration, the MC UE is subscribed to key download notifications from the MCX Server.

1. The MCX Server sends a key download message (SIP NOTIFY) to the MC UE. The MC UE extracts the signalling key from the key download message.

2. Upon successful extraction of the signalling key, the MC UE returns a key download success message (200 OK response) to the MCX Server. Upon receipt of a notification of success, the MCX Server is able to begin to use the key for protection of signalling traffic.

5.9 Key management during MBMS bearer announcement

When MBMS is used, the MCX Server and MC clients require a shared MuSiK to protect multicast signalling on the multicast bearer between the MCX Server and MC client. The MuSiK shall be distributed over a MBMS bearer announcement message where a MuSiK requires distribution at the same time as a MBMS bearer is established or when a participating UE roams into the MBMS area.In this case, the encapsulated key is distributed within the SDP content of the bearer announcement message.

The MBMS bearer announcement message may also be used to distribute a MSCCK as described in Annex H.

The security procedures for key distribution via an MBMS bearer announcement message are identical to those used for 'key download' messages, described in clause 5.8.

6 Supporting security mechanisms

6.1 HTTP

6.1.1 Authentication for HTTP-1 interface

For authentication of the HTTP-1 reference point, one of the following authentication mechanisms shall be performed between the HTTP client in the MC UE and the HTTP server endpoint (HTTP proxy, IdM server or KMS):

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

Certificate based authentication shall follow the profiles given in 3GPP TS 33.310 [5], clauses 6.1.3a and 6.1.4a. 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-PSK) is specified in the TLS profile given in 3GPP TS 33.310 [5], annex E.

6.1.2 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 MC 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.

6.2 SIP

6.2.1 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 MC service provider's SIP database depending on the SIP core deployment scenarios specified in 3GPP TS 23.379 [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].

6.2.2 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.

6.3 Network domain security

6.3.1 LTE access authentication and security

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

6.3.2 Inter/Intra domain interface security

To ensure security of the interfaces between network elements within a trusted domain and between 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.

7 MCPTT and MCVideo

7.1 General

This clause described the security procedures for both MCPTT and MCVideo.

7.2 Private communications

7.2.1 Key management

7.2.2 Security procedures (on-network)

The following private communication 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 23.379 [2] describes manual and automatic commencement for private MCPTT communications in both a single MC system and across multiple MC systems, while 3GPP TS 23.281 [37] describes manual and automatic commencement for private MCVideo communications within a single MC system.

Securing of on-network private MCPTT or MCVideo communications is summarized in the following sub clauses and applies to the aforementioned MCPTT and MCVideo private call use cases.

The private call setup message used to establish these security procedures may be pre-generated to increase the efficiency of the communication. Additionally, the MC UE may attach a second SAKKE component which encrypts the PCK to the initiating user (in addition to the terminating user) for use in the 'SAKKE-to-self' procedure.

The security procedure for an on-network MCPTT or MCVideo private call within a single MC system is summarized in figure 7.2.2-1, The security procedure for securing an on-network MCPTT private call between multiple MC systems is summarized in figure 7.2.2-2. The intent of these on-network security procedures is to transfer a PCK and PCK-ID to the terminating UE in order to provide end-to-end security of the media.

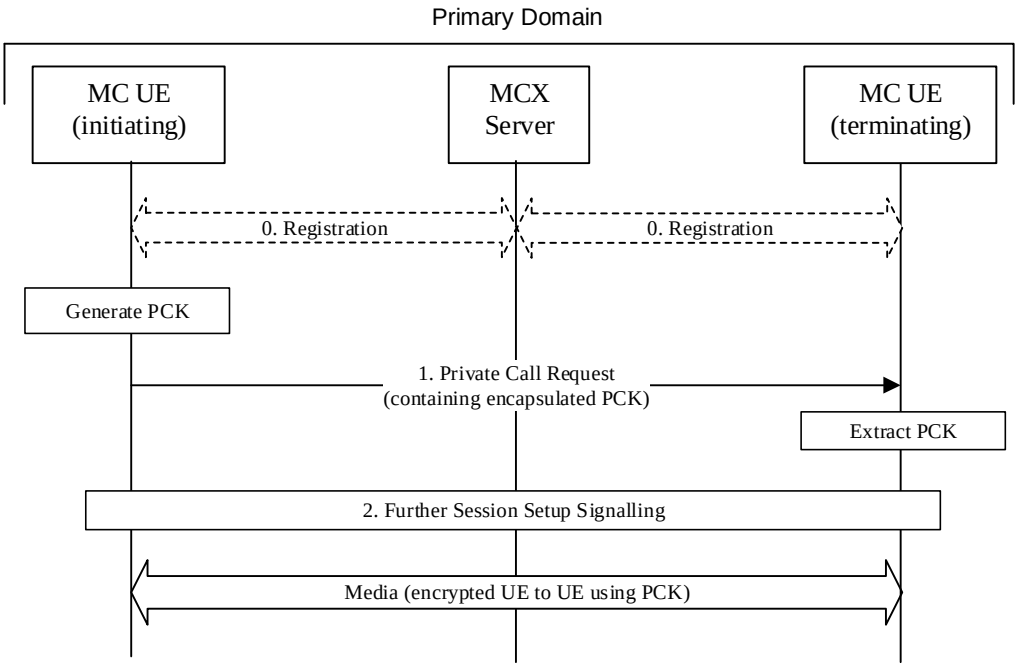


Figure 7.2.2-1: Private call security procedure for on-network PCK distribution for single domain

The procedure in figure 7.2.2-1 is now described step-by-step.

0. Prior to beginning this procedure it is assumed that the MC UEs have an authenticated MC user and that the MC UEs have been provisioned with key material associated with a user's MC service ID by a KMS as described in clause 5.3.

1. The initiating MC UE generates the PCK and sends a private call request to the terminating MC UE. The message is sent to the primary MC server of the initiating UE where it is forwarded to the intended receipient UE. Within this message includes an SDP offer which contains a MIKEY-SAKKE I_MESSAGEs as defined in

IETF RFC 6509 [11]. The I_MESSAGE encapsulates the PCK for the terminating MC 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.

2. Further session signalling occurs as defined in 3GPP TS 23.379 [2] for MCPTT and 3GPP TS 23.281 [39] for MCVideo.

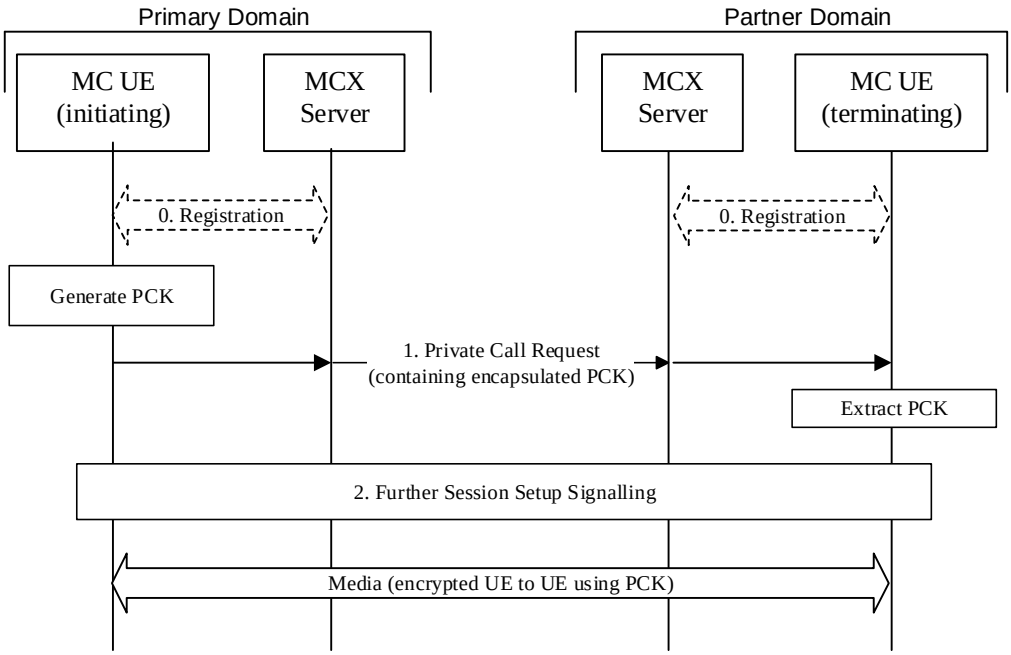


Figure 7.2.2-2: Private call security procedure for on-network PCK distribution between multiple domains

The procedure in figure 7.2.2-2 is now described step-by-step.

0. Prior to beginning this procedure it is assumed that the MC UEs have an authenticated MC user and that the MC UEs have been provisioned with key material associated with a user's MC service ID by a KMS as described in clause 5.3.

1. The initiating MC UE generates the PCK and sends a private call request addressed to the terminating MC UE. The message is first routed to the primary MC server of the initiating UE. The primary MC server routes the private call request to the partner server (home of the intended receipient UE), which is then routed to the receipient UE. The private call request message includes an SDP offer which contains a MIKEY-SAKKE I_MESSAGE as defined in IETF RFC 6509 [11]. The I_MESSAGE encapsulates the PCK for the terminating MC 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.

2. Further session signalling occurs as defined in 3GPP TS 23.379 [2].

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

7.2.3 Security procedures (off-network)

3GPP TS 23.379 [2] describes manual and automatic commencement for private off-network MCPTT communications, while 3GPP TS 23.281 [37] describes manual and automatic commencement for private off-network MCVideo communications.

Securing off-network private MCPTT or MCVideo communications is summarized in the following sub clauses and applies to the aforementioned MCPTT and MCVideo off-network private call use cases.

The private call setup message used to establish these security procedures may be pre-generated to increase the efficiency of the communication. Additionally, the MC UE may attach a second SAKKE component which encrypts the PCK to the initiating user (in addition to the terminating user) for use in the 'SAKKE-to-self' procedure.

The security procedure for securing an off-network private call is summarized in figure 7.2.3-1. As part of this process, the PCK and PCK-ID are securely transferred to the terminating UE.

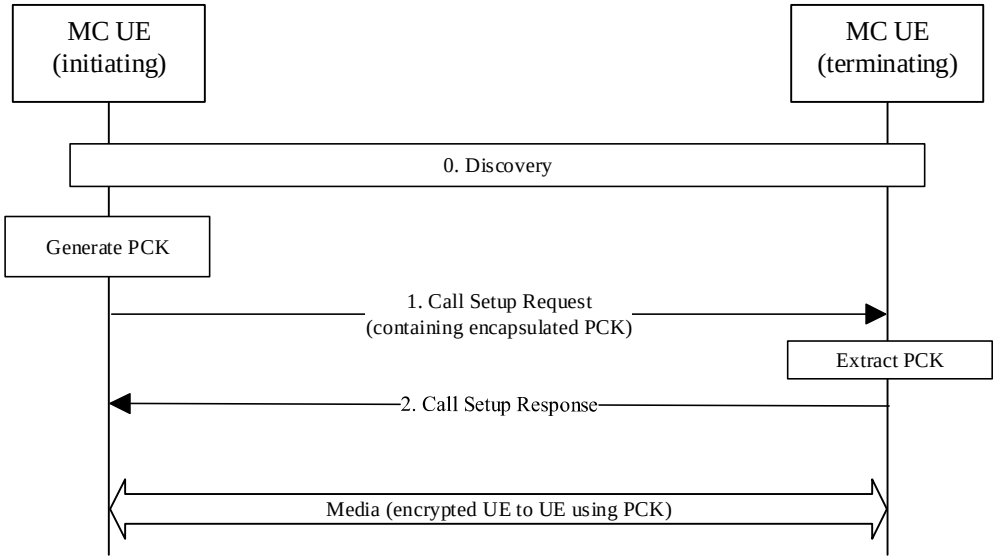


Figure 7.2.3-1: Private Call security procedure for off-network PCK distribution

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

0. Prior to beginning this procedure the MC UEs may have performed a discovery procedure. Additionally, the MC UEs have been provisioned with key material associated with a user's MC Service user IDs by a KMS as described in clause 5.3.

1. The initiating UE generates the PCK and sends a Call Setup Request to the terminating UE. Within this message, the initiating UE includes a MIKEY-SAKKE I_MESSAGEs as defined in IETF RFC 6509 [11]. The I_MESSAGE encapsulates the PCK for the terminating UE, 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.

2. A Call Setup Response is returned to the initiating UE as defined in 3GPP TS 23.379 [2] for MCPTT and as defined in TS 23.281 [37] for MCVideo.

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.

7.2.4 First-to-answer security and key management

7.2.4.1 Overview

A 'first-to-answer' call as defined in clause 10.15 of TS 23.379 [2], is a call request sent to multiple users inviting them into a private call, and where the first user to answer the request is brought into the private call with the initiator while the rest of the invited users are subsequently rejected. Consequently, a specific key management solution is required.

The following defines a method for performing key distribution for a first-to-answer call. From a security point-of-view, the approach is to perform a private call key distribution from the answering client to the initiating client of the call.

The first-to-answer messages are routed over the signalling reference points. Consequently, the security mechanisms for protecting signalling between the MC Domain and the MC UE are applied to these messages. This includes the security mechanisms defined in clause 6. Where application signalling security is supported, the security mechanisms defined in clause 5.3 are used, ensuring that the user identities (MCPTT IDs) are confidentiality protected with the CSK or SPK as per clause 5.3.

7.2.4.2 First-to-answer request and response

The first-to-answer request (containing the list of target MCPTT IDs) is sent by an initiating UE to the MCPTT server. No key material is provided in the first-to-answer request.

The first-to-answer response is sent by a target UE in response to a first-to-answer request. The first-to-answer response contains both an encapsulatedPCK for the private call and a pair of MCPTT IDs corresponding to the participants (intiator and target) of the private call.

The PCK is encapsulated as defined in clause 5.6. In this case, the 'initiating entity' shall be the MC user who provides the first-to-answer response. The initiating entity URI shall be the MC Service user ID of the user who made the first-to-answer response. The receiving entity shall be the MC user who made the first-to-answer request. The receiving entity URI shall be the MC Service user ID of the user who made the first-to-answer request.

7.2.4.3 First-to-answer call setup with security

Figure 7.2.4.3-1 below illustrates the first-to-answer call setup procedure with security.

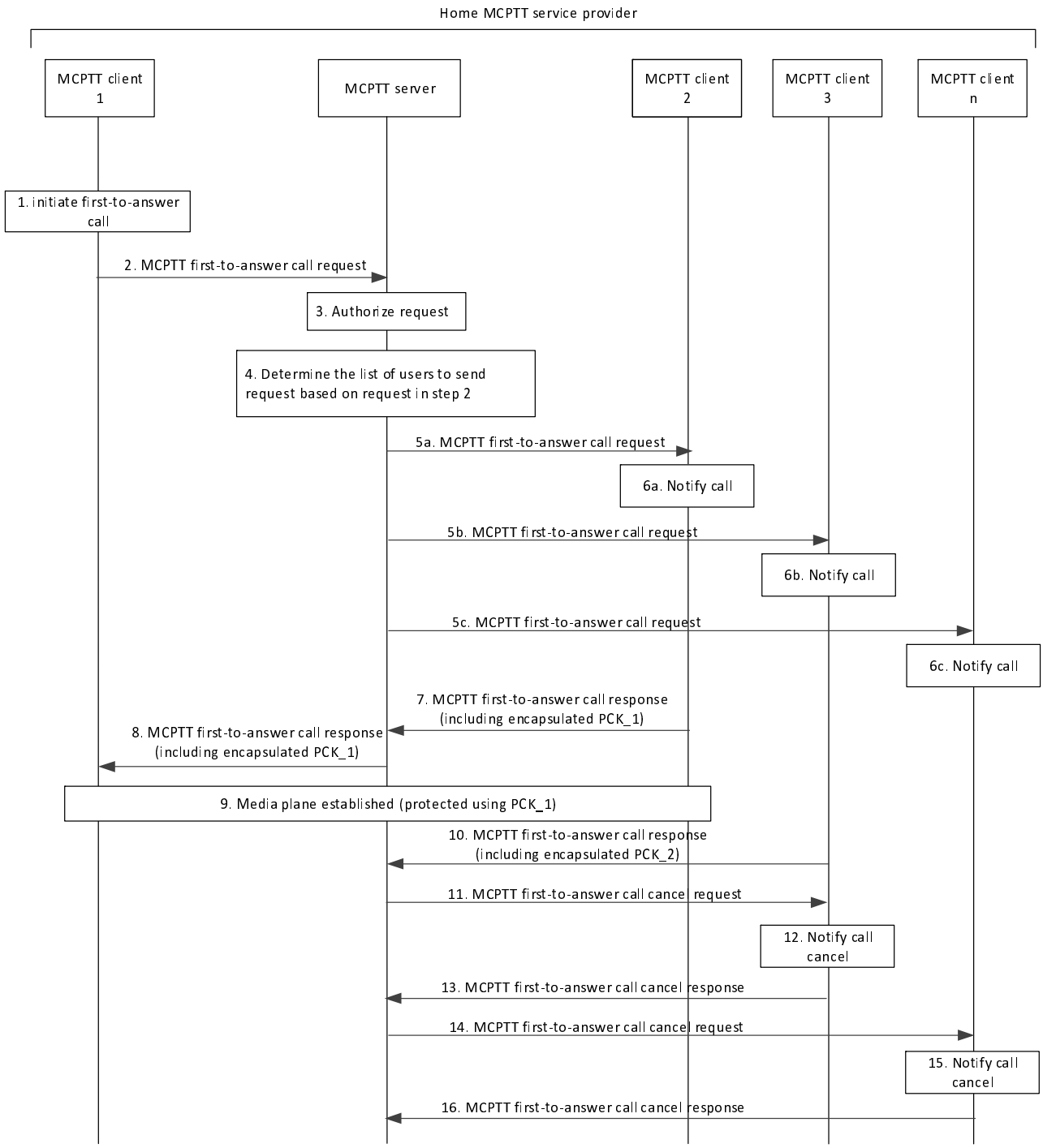


Figure 7.2.4.3-1: First-to-answer call setup and key management

1 to 6. First-to-answer call signalling occurs as defined in TS 23.379 [2]. These messages do not contain security-related key material.

7. MCPTT user at MCPTT client 2 accepts the call, which causes the MCPTT client 2 to send a first-to-answer call response to the MCPTT server. Included in the response, is the PCK (PCK_1) encapsulated to the user associated with the initiating client, MCPTT client 1. The PCK is then included in the SDP content of the response.

8. The MCPTT server forwards the first-to-answer call response to MCPTT client 1 indicating that the MCPTT user at MCPTT client 2 has accepted the call. MCPTT client 1 extracts the PCK from the message.

9. The media plane for communication is now established and protected with the shared PCK.

10. MCPTT user at MCPTT client 3 accepts the call and sends a first-to-answer call response to the MCPTT server. MCPTT client 3 also includes an encapsulated PCK (PCK_2) in the response.

11. Since the first-to-answer call response from MCPTT client 2 has already been accepted, the MCPTT server sends a MCPTT first-to-answer call cancel request to MCPTT client 3. The encapsulated PCK provided by MCPTT client 3 (PCK_2) is discarded.

12-16. First-to-answer call signalling occurs as defined in TS 23.379 [2]. These messages do not contain security-related key material.

7.2.4.4 First-to-answer media protection

The first-to-answer media plane shall be protected as for a private call. Clause 7.4.1 is applied to convert the PCK into the SRTP Master Key/Salt, and clause 7.5 is applied for the protection of the first-to-answer media.

7.3 Group communications

7.3.1 General

To support MCPTT and MCVideo group communications, group security procedures are used to establish and distribute keys to the members of predefined or dynamically defined groups.

Key material (GMK and GMK-ID) for a predefined group is created and distributed by the GMS to the members of the group via the common key distribution mechanisms defined in clause 5.7.

Key material for dynamically created groups is created and distributed by the GMS to the members of the group via the security mechanisms defined in the following sub clauses.

7.3.2 Group creation security procedure

The group creation procedure is described in clause 10.2.3 of 3GPP TS 23.280 [36] and applies to the MCPTT scenario of normal group creation by an MC administrator and user regrouping operations by an authorized user/dispatcher. To establish the security context for the group, the GMS follows the procedures in clause 5.7 to create a new GMK and GMK-ID.

The encapsulated GMK and GUK-ID is sent to group members by the GMS within a notification message (step 4 in clause 10.2.3 of 3GPP TS 23.280 [36]). The procedure is equivalent to that described in clause 5.7 of this specification.

7.3.3 Dynamic group keying

7.3.3.1 General

In the GMK distribution procedures described in this clause, the GMS is provisioned with the same information as any MC UE by the KMS as described in clause 5.3; the only distinguishing feature is that the GMS's identity is authorized to create groups.

NOTE 1: 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 MCPTT group procedures within 3GPP TS 23.280 [36].

NOTE 2: The dynamic group keying mechanisms may not support off-network scenarios.

7.3.3.2 Group regrouping security procedure (within a single MC domain)

Group Regroup procedures for the MC system are described in clause 10.2.4.1 of 3GPP TS 23.280 [36]. To create the security context for the temporary group, the GMS follows the procedures in clause 5.7, creating a new GMK and GMK-ID for the temporary group.

An encapsulated GMK and GUK-ID is sent to the temporary group members by the GMS within a notification message (step 5 in clause 10.2.4.1 of 3GPP TS 23.280 [36]). The procedure is equivalent to that described in clause 5.7.

7.3.3.3 Group regrouping security procedure (involving multiple MC domains)

The MCPTT group regroup security procedure (shown below in figure 7.3.3.3-1) involves multiple MC users from multiple MC domains and is an integrated component of the MCPTT group regrouping procedure described in clause 10.2.4.2 of 3GPP TS 23.280 [36].

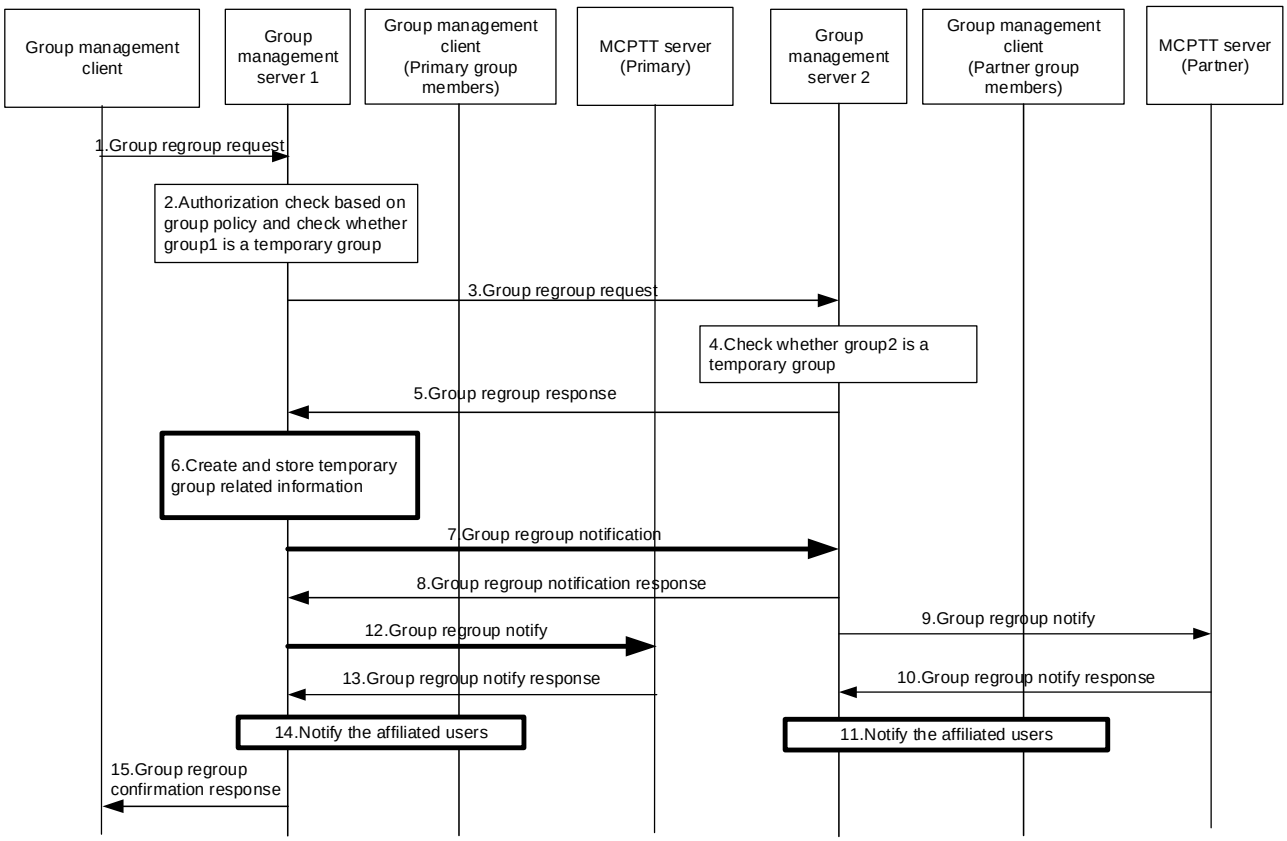


Figure 7.3.3.3-1: Group regroup security procedure (multiple MC domains)

Prior to beginning the procedure, the MC UEs, primary GMS and partner GMS are provisioned by a KMS as described in clause 5.3.

1-5: These steps are as defined in clause 10.2.4.2 of 3GPP TS 23.280 [36].

6: To create the security context for the temporary group, the primary GMS creates a new GMK and GMK-ID for the temporary group along with other group related information.

7,8: The primary GMS notifies the partner GMS of the group regroup operation. The primary GMS includes a Group Key Transport payload following the procedures in clause 5.7, 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.

9,10: These steps are as defined in clause 10.2.4.2 of 3GPP TS 23.280 [36].

11: The partner GMS extracts the GMK and GMK-ID from the notification. The partner GMS then notifies the affiliated users within the partner MC domain. 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 5.7.

12,13: The primary GMS notifies the primary MCPTT server of the group regroup. The primary GMS includes a Group Key Transport payload including a GMK and GUK-ID following the procedures in clause 5.7. The GMK is encrypted to the identity of the primary server and is signed using the identity of the primary GMS.

14: The primary GMS notifies the affiliated users within its own MC domain. The primary GMS includes a Group Key Transport payload including a GMK and GUK-ID following the procedures in clause 5.7. The GMK is encrypted to the identity of the MC user and is signed using the identity of the primary GMS.

15: This step is as defined in clause 10.2.4.2 of 3GPP TS 23.280 [36].

7.3.4 Offline media Protection (GMK)

7.4 Key derivation for media

7.4.1 Derivation of SRTP master keys for private call

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.

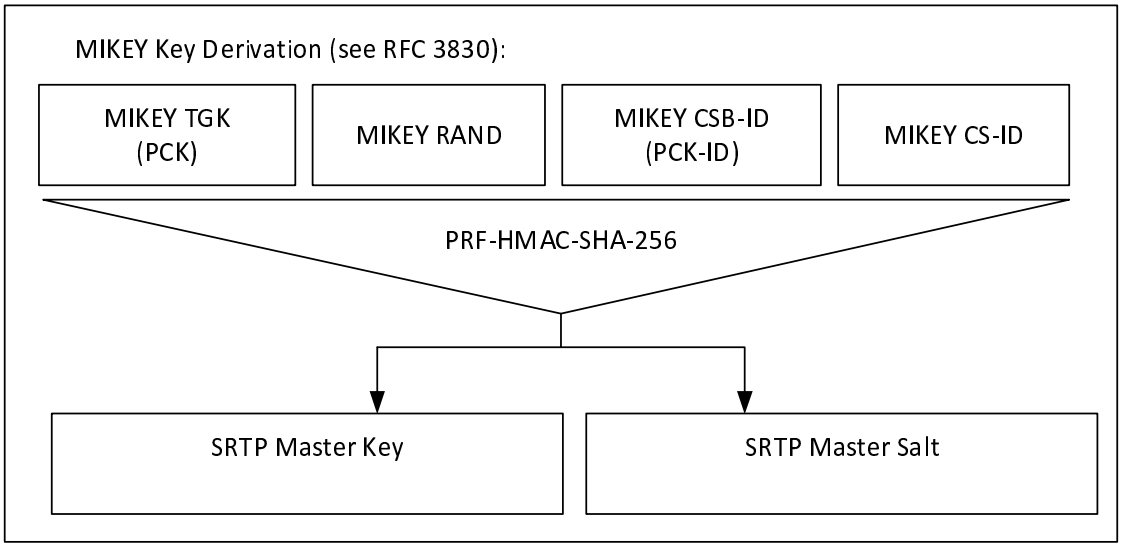


Figure 7.4.1-1: Key Derivation for media stream protection

To identify the security context from the media stream a SRTP Master Key Identifier (MKI) is required. The MKI shall be the 32-bit PCK-ID which has a purpose tag of '1'.

When the MC client is operating off network, the PCK is used to derive keys for floor and media 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 7.3.6 for key derivation procedures for floor and media control (SRTCP) when the MC client is operating on-network.

7.4.2 Derivation of SRTP master keys for group media

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.

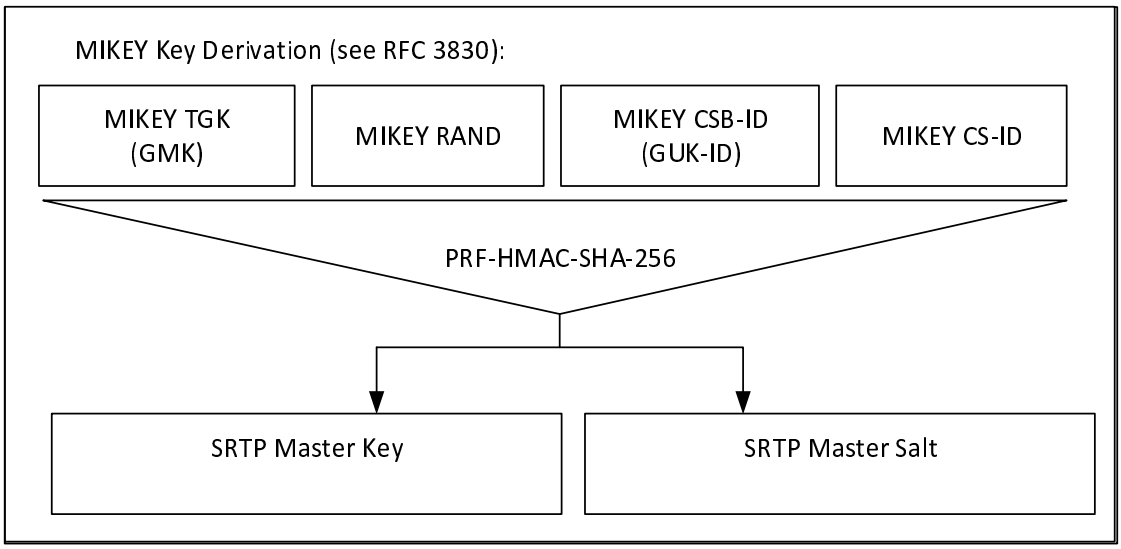


Figure 7.4.2-1: Key Derivation for media stream protection

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 MC client is operating off network, the GMK is used to derive keys for floor and media 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 7.3.3 for key derivation procedures for floor and media control (SRTCP) when the MC client is operating on-network.

7.5 Media protection profile

7.5.1 General

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

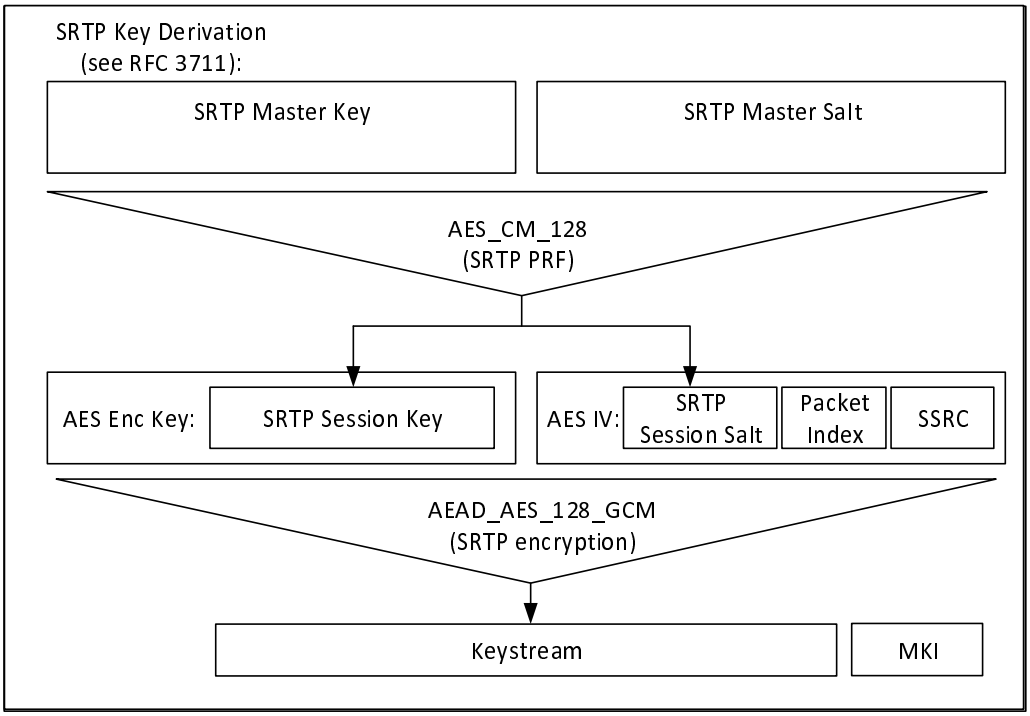


Figure 7.5.1-1: Security mechanism for media stream protection

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 MC 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.

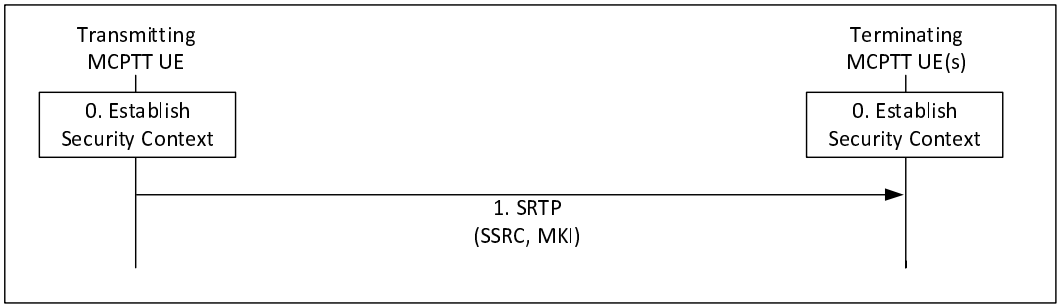


Figure 7.5.2-1: Security procedure for media stream protection

Figure 7.5.2-1 shows the security mechanism.

0) Prior to beginning this procedure the MC 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.

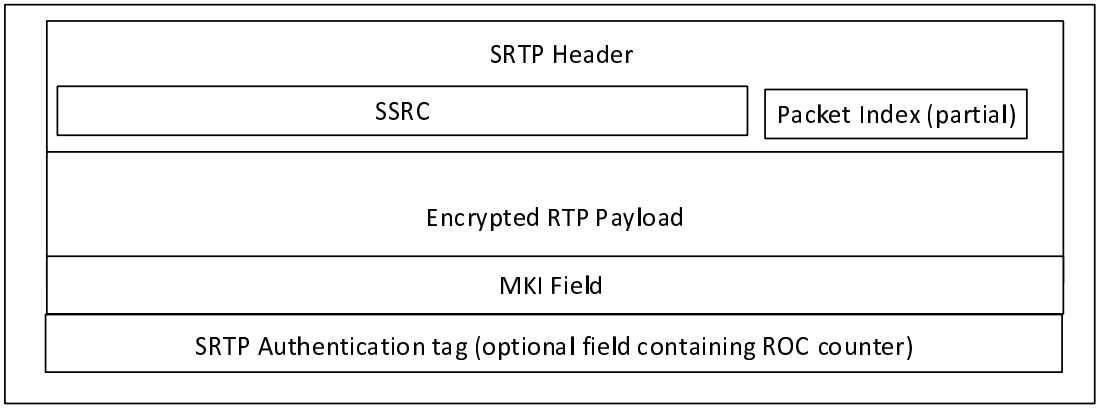


Figure 7.5.2-2: SRTP packet format showing security parameters

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

8 MCData

8.1 Overview

MCData SDS allows transmission of short data messages (SDS), either private or group, over both the signalling plane (reference point MCData-SDS-1) and media plane (reference point MCData-SDS-2).

MCData File Distribution (FD) also allows for transmission of files over the media plane.

MCData signalling for SDS and File Distribution is performed over MCData signalling payloads routed within SIP messages. Protection for these signalling messages uses the same key material as for MCPTT and MCVideo.

The MCData SDS or FD message also contains a data payload. This may be within a SIP message should the signalling plane be used, or within a MSRP message should the media plane be used. The data payload may be end-to-end confidentiality and integrity protected according to an end to end security context payload. For one-to-one

communications the PCK is used to protect the MCData data payload. For group communications, the GMK is used to protect the MCData data payload. The data payload may also be authenticated by the initiator.

Distribution of the PCK is within the signalling channel setup for the MCData private message (either SDS or FD). Distribution of the GMK is as defined in clause 5.7.

8.2 Key Management

Key management for MCData follows the same model as MCVideo and MCPTT. Where a key is used for protection of MCData or MCVideo data, the same type of key shall be used in the same circumstance for MCData. Each key used for protection of MCData payloads is known as the MCData Payload Protection Key (DPPK).

MCData signalling payloads are protected as follows:

- Unicast MCData signalling payloads between client and server are protected using the CSK (e.g. the DPPK is the CSK).
- Multicast MCData signalling payloads from server to client are protected using a MuSiK (e.g. the DPPK is a MuSiK).
- MCData signalling payloads between servers are protected using the SPK (e.g. the DPPK is the SPK).
- MCData signalling payloads between two offline clients are protected using a PCK (e.g. the DPPK is the PCK).
- MCData signalling payloads between a group of offline clients are protected using a GMK (e.g. the DPPK is the GMK).

MCData data payloads are protected as follows:

- MCData data payloads end-to-end protected between two online clients are protected using a PCK (e.g. the DPPK is the PCK).
- MCData data payloads end-to-end protected between two offline clients are protected using a PCK (e.g. the DPPK is the PCK).
- MCData data payloads end-to-end protected between a group of online clients are protected using a GMK distributed by a GMS (e.g. the DPPK is the GMK).
- MCData data payloads end-to-end protected between a group of offline clients are protected using a GMK distributed by a GMS (e.g. the DPPK is the GMK).

NOTE: The DPPK is not a new type of key, it describes how the MC system's existing key types are used to protect MCData. Consequently, there will be multiple DPPKs in the MC System depending on the communication channel. Furthermore, while a PCK and a GMK may both be used as a DPPK to protect MCData in different channels, the PCK and the GMK shall never be the same key.

8.3 One-to-one communications

The purpose of key management is to establish a MCData Payload Protection Key (DPPK) for the one-to-one communication channel between the pair of communicating clients. In the case of a one-to-one communication, the DPPK shall be the PCK. The PCK is used for end-to-end protection of one-to-one (private) SDS or FD data payloads.

The PCK and PCK-ID are distributed within the SIP message used to initiate the session.

The PCK and PCK-ID is distributed using service-specific signalling. For all MCData services, SIP signalling is used to establish or send the MCData communication. The PCK and PCK-ID is distributed within a MIKEY payload contained within the SDP offer sent from the initiator to the receiver in the same way as for MCPTT and MCVideo. The procedures for PCK distribution are defined within clause 5.6.

This key distribution mechanism applies to the following messages defined in TS 23.282 [38]:

- MCData standalone data request
- MCData session data request

- MCData FD request

When required by the MCData service provider, protection shall be applied to the MCData data payloads using the PCK. Payload authentication may also be applied. The mechanisms used to secure these payloads are described in clause 8.5.

Once the PCK is established between the source and destination, SDS and FD exchanges between this same source and destination may continue to use the same PCK for subsequent MCData communications by simply providing the PCK-ID in every SDS message.

8.4 Group communications

The purpose of key management is to establish a MCData Payload Protection Key (DPPK) for the group communication between the group of communicating clients. In the case of group communication, the DPPK shall be the GMK. The GMK is distributed in the same way as for MCPTT and MCVideo group communications, as defined in clause 5.7.

When required by the MCData service provider, protection shall be applied to the MCData data payloads using the GMK. Payload authentication may also be applied. The mechanisms used to secure these payloads are described in clause 8.5.

8.5 MCData payload protection

8.5.1 General

The following protected (encrypted and integrity protected) payloads are defined for MCData SDS and file distribution:

- Protected SDS Signalling Payload
- Protected FD Signalling Payload
- Protected Data Payload
- Protected SDS notification message
- Protected FD notification message

The following authenticated payloads are defined for MCData SDS and file distribution:

- Authenticated Data Payload

The following authenticated and protected (encrypted and integrity protected) payloads are defined for MCData SDS and file distribution:

- Authenticated and Protected Data Payload

In this case both the procedures for protecting a payload and authenticating a payload are applied

8.5.2 Prequisites

8.5.2.1 Prequisites for protected payloads

The prequisites for encryption and integrity protection of a protected payload is that the MC client(s) or MC server(s) have a shared MCData Payload Protection Key (DPPK). This shall be the CSK, SPK, MuSiK, GMK or PCK depending on the payload that will be protected. The DPPK will also have a shared key identifier, the DPPK-ID. This shall be the CSK-ID, MuSiK-ID, SPK-ID, GMK-ID or PCK-ID respectively, based upon the type of key used.

8.5.2.2 Prequisites for authenticated payloads

The prequisites for authentication of an authenticated payload is that the MC client will have been keyed by a KMS as defined in clause 5.3.

8.5.3 Key derivation for protected payloads

Before protecting an MCData payload, the DPPK is hashed through a KDF (similar to the process used for XML protection for application signalling), to produce a MCData Payload Cipher Key (DPCK). The KDF is defined in Annex F.1.5.

8.5.4 Payload protection

8.5.4.1 Format of protected payloads

All protected payloads shall have the format defined in table 8.5.4.1-1:

Table 8.5.4.1-1: MCData Protected Payload message content

Information Element	Type/Reference	Presence	Format	Length
Message Type	Message type	M	V	1
Date and Time	Date and Time of creation of protected payload message.	M	V	5
Payload ID	The identifier for the payload.	M	V	4
Payload sequence number	The sequence number of the protected payload.	M	V	1
Algorithm	See 8.5.4.2	M	V	1
IV	Initialisation vector (or nonce) for message	M	V	16
DPPK-ID	Key identifier	M	V	4
Payload	Protected Payload (Ciphertext)	M	TLV-E	x

Where 'Payload' will be the encrypted and integrity-protected payload encoded in a binary format.

NOTE 1: Date and Time is included as plaintext to allow the MCData server to order end-to-end protected messages and assess whether end-to-end protected messages may have expired.

NOTE 2: Payload ID and Payload sequence number allow protected payloads to be split over multiple SIP messages.

8.5.4.2 Encryption of protected payloads

Protection of payloads shall support the following algorithms (cipher suites):

Table 8.5.4.2-1: DP_AES_128_GCM algorithm parameters

Parameter	Value/Reference
Algorithm ID	DP_AES_128_GCM
Cipher	AEAD_AES_128_GCM (as defined in RFC 5116 [43])
DPCK Key length	128 bits
IV length	128 bits
AEAD authentication tag length	128 bits

Table 8.5.4.2-2: DP_AES_256_GCM algorithm parameters

Parameter	Value/Reference
Algorithm ID	DP_AES_256_GCM
Cipher	AEAD_AES_256_GCM (as defined in RFC 5116 [43])
DPCK Key length	256 bits
IV length	256 bits
AEAD authentication tag length	128 bits

In using the above cipher suites as defined in RFC 5116 [43], the plaintext, P, shall be the full original plaintext payload. The associate data (AD) shall be the Message Type, Date and Time, Payload ID, Payload sequence number, Algorithm, IV, and DPPK-ID fields within the MCData Protected Payload message content defined in clause 8.5.4.1.

8.5.5 Payload authentication

Authenticated payloads shall have the format defined in table 8.5.5-1:

Table 8.5.5-1: MCData Authenticated Payload message content

Information Element	Type/Reference	Presence	Format	Length
Original Payload	Original Payload (unchanged)	M	TLV-E	x
Signature	Based on algorithm	M	TLV-E	x

The signature shall be on the entire payload excluding the value of the signature element. However, the type and length of the signature element shall be included in the signature. The signature value shall be encoded in binary format.

The ECCSI signature algorithm as defined in RFC 6507 [9] shall be supported by MC clients.

NOTE: It is assumed that the signer's identity and the signer's KMS is known prior to distribution of the payload (e.g. due to establishment of the DPPK in the signalling channel).

9 Signalling Protection

9.1 General

Signalling between entities in the MC System are defined as:

- RTCP signalling (e.g. floor control), or
- XML signalling (within SIP messages)

To allow this signalling to be protected, key distribution mechanisms are required to distribute the associated keys.

For protecting signalling between the client and the server, there are two key distribution mechanisms:

- 'CSK upload' procedure (as defined in clause 5.4).
- 'Key download' procedure (as defined in clause 5.8).

For protecting signalling between MCX Servers, there is one key distribution mechanism:

- manual SPK configuration (as defined in clause 5.5).

9.2 Key distribution for signalling protection

9.2.1 Client-Server Key (CSK)

9.2.1.1 General

A Client-Server Key is required to protect unicast RTCP signalling between the MC client and the MCX Server. The use of the CSK in this context is defined in clause 9.4.

Additionally, the MC Service provider may require that MC identities, access tokens and other sensitive information transferred between clients and MC 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. The use of CSK in this context is defined in clause 9.3.

The uses of the CSK are shown in Figure 9.2.1-1.

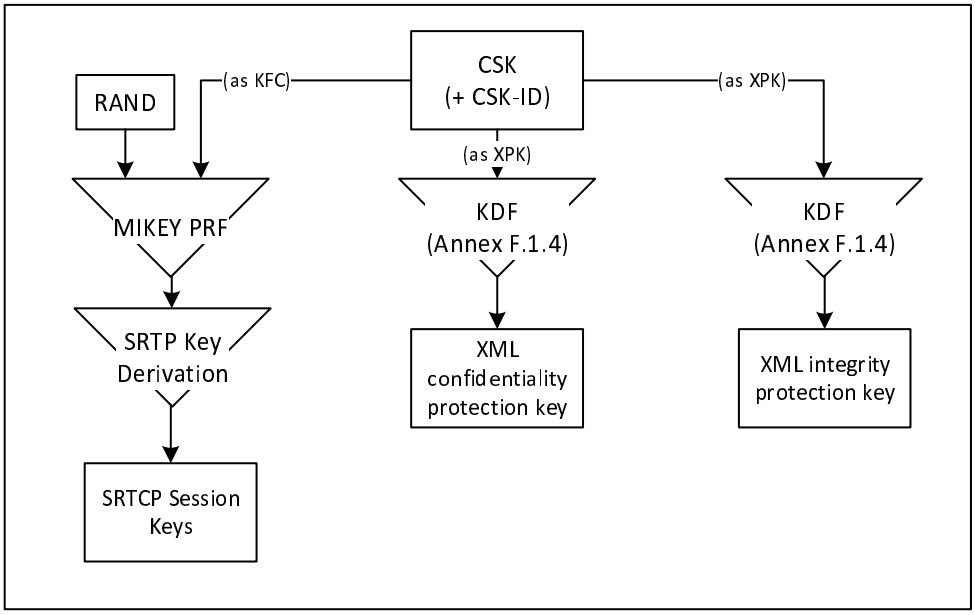


Figure 9.2.1-1: Uses of the Client-Server Key

9.2.1.2 Creation of the CSK

The 128-bit CSK is initially 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 MC user logs off, or some other indication. If during the active SIP session an update of the CSK is required, the server generates a CSK and provides it to the client using the mechanism defined in clause 5.8.

9.2.1.3 Initial 'CSK Upload' Procedure

The CSK is initially distributed via the 'CSK upload' procedure as defined in clause 5.4. The 'CSK upload' procedure creates a security association between the MC client and the MCX Server and occurs during the client's initial connection with the MC Server.

The following steps describe how the client obtains the user specific key material and securely transfers the CSK to a server within the MC 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 and encapsulates the CSK as described in clause 5.4.

2) The client includes the encapsulated CSK in its initial SIP REGISTER or in a SIP PUBLISH message that is used to perform the MC user authorization procedure, and sends the SIP message addressed to the PSI of the server.

An illustration is provided below as an example of how this message in included in the body of the SIP REGISTER message. The MIME media type "application/mikey" IETF RFC 3830 [22] is used in this example to insert a MIKEY I_MESSAGE in the SIP payload:

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 following steps describe how the MCX Server retrieves the CSK from the SIP message:

1) The server receives the SIP message and decrypts the encapsulated the CSK as described in clause 5.4.

2) Once the CSK has been extracted, MC user specific information (e.g. the access token) protected in the SIP message as defined in clause 9.3.4, may be decrypted.

9.2.1.4 CSK update via 'key download'

The MCX Server may decide to update an existing CSK at any time. This may be due to CSK revocation or expiry.

The CSK shall be updated by the MCX Server using the 'key download' procedure, defined in clause 5.8. Upon receipt of a CSK via a 'key download' procedure, the MC client shall identify the type of key as a CSK via the 4 most significant bits of the CSK-ID. The MC client shall:

- discard any previous CSKs associated with the MC Server FQDN, and
- use the new CSK for uplink signaling with the MC Server.

9.2.2 Multicast Signalling Key (MuSiK)

The Multicast Signalling Key (MuSiK) is required to protect multicast RTCP signalling from the MCX Server to the MC client. This includes MBMS subchannel control, floor control, media control and transmission control messages.

The MuSiK shall be distributed using the ‘key download’ procedure or within the MBMS bearer announcement message.

A ‘key download’ procedure is described in clause 5.8. Where a MuSiK is established at the same time as a MBMS bearer is established, MuSiK distribution is performed by attaching the MuSiK to a MBMS bearer announcement message rather than creating a separate 'key download' message. This is described in clause 5.9.

The use of the MuSiK is shown in Figure 9.2.2-1.

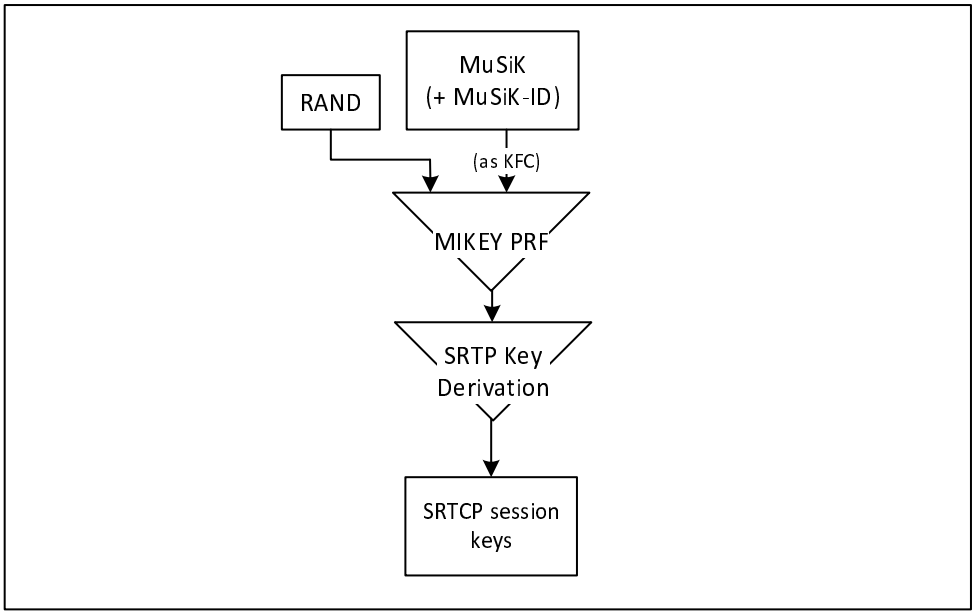


Figure 9.2.2-1: Uses of the Multicast Signalling Key (MuSiK)

The MCX Server distributes the Multicast Signalling Key (MuSiK) to a client when:

- The MCX Server requires protected signalling over the MBMS bearer to the MC client. In this case, an initial MuSiK ($MuSiK_{All}$) is distributed to the client. By default, this MuSiK is used to protect all multicast signalling.
- The MCX Server requires the transmission of group-related signalling (e.g. media control or floor control) over an MBMS bearer to the MC client, and the group configuration indicates that cryptographic segregation is required for multicast group signalling. In this case, a new MuSiK ($MuSiK_i$) is distributed to protect this group signalling.
- The MCX Server requires an existing MuSiK to be replaced. This may be due to revocation or expiry.
- A participating UE (MC client) of the multicast group roams into the MBMS bearer coverage area.

NOTE: It is expected that for the majority of MCX Groups and MBMS bearers, the participating MCX Server will use a single MuSiK. Where a MCX Group or MBMS bearer has privacy requirements, these procedures allow a new MuSiK to be distributed specifically for that purpose. Consequently, it is not expected that a new MuSiK will need to be distributed before each new bearer is established.

Upon receipt of a MuSiK the MC client shall store the MuSiK and MuSiK-ID. Should the MuSiK be rejected by the MC client, the MCX Server shall only use a unicast bearer when distributing signalling to the MC client.

Upon receipt of multicast SRTCP, the MC client shall inspect the MKI of the SRTCP packet which shall contain the MuSiK-ID. The MuSiK-ID shall be used to lookup the correct MuSiK for decrypting the SRTCP packet.

9.2.3 Signalling Protection Key (SPK)

The SPK is used to protect communications between MCX Servers. The SPK is distributed as defined in clause 5.5. The uses of the SPK for inter-server protection are shown in Figure 9.2.3-1.

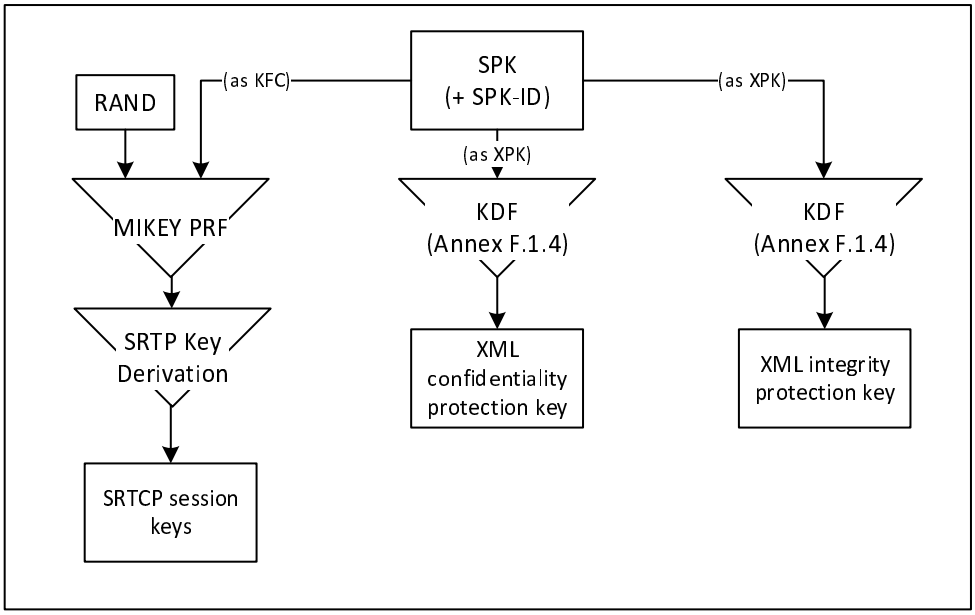


Figure 9.2.3-1: Uses of the Signalling Protection Key

9.3 Application signalling security (XML protection)

9.3.1 General

Certain values, keys and identifiers transferred in XML between a server in the MC domain and client may be treated as sensitive by public safety users and may require protection. To protect these values from all other entities outside of the MC 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 be applied to the entire XML document or to the following individual identifiers and values:

- MCX service user ID (e.g. MCPTT ID, MCData ID, MCVideo ID).
- MCX Group ID.
- User location information.
- Alerts.
- Access token.
- KMS provisioned key material.

Where confidentiality protection is applied to the entire XML document, the 'type' of message shall be clearly stated within the EncryptedData payload. The name shall reflect the names used in the message flows defined in TS 23.379 [2], TS 23.280 [36], TS 23.281 [37] and TS 23.282 [38]. This will allow the serving network to understand how their network is being used.

NOTE: Where the MCPTT Server is supporting legacy clients, these clients may not support confidentiality protection of the entire XML document. In this case, only individual identifiers and values should be confidentiality protected.

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 MC Domain the XPK shall be the 128-bit shared Client-Server Key (CSK) established as defined in clause 9.2.1. The XPK-ID shall be the CSK-ID.

For connections between servers inside and across MC Domains the XPK shall be the 128-bit manually provisioned SIP Protection Key (SPK) established as defined in clause 9.2.3. The XPK-ID shall be the SPK-ID

For connections between the KMS and the MC client, the XPK shall be the 256-bit manually provisioned TrK, described in clause 5.3.3. 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 annex F.1.4. The XPK-ID may be listed in the XML to aid decryption.

9.3.4 Confidentiality protection using XML encryption (xmlenc)

9.3.4.1 General

This clause defines an optional mechanism to allow specific XML content within the XML elements and XML URI attributes to be encrypted between the client and the server.

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

9.3.4.2 XML content encryption

XML content within XML elements is encrypted as defined by XML Encryption Syntax, Version 1.1 [27].

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 MC 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>
```

9.3.4.3 XML URI attribute encryption

XML attribute encryption shall be performed by encrypting the URI and embeddeding the encrypted ciphertext within a new URI. The appended domain name of the new URI identifies the attribute as having confidentiality protection. Encryption shall be performed using the AES-128-GCM [42], as the encryption algorithm, XPK as the key, and the use of a 96 bit randomly selected IV.

The output URI is structured to contain:

- the base64 encoded encrypted URI;
- the string ";iv=" followed by the base64 encoded 96-bit random initialisation vector (IV) which is used by the AES-128 encryption algorithm (as described in TS 33.203 subclause 6.4).
- the string ";key-id=" followed by the base64 encoded encryption key identifier (XPK-ID);
- the string ";alg=" followed by the encryption algorithm identifier (128-bit encryption algorithm "128-AES-GCM");
- the appended domain name of the new URI e.g. “@mc1-encryption.3gppnetwork.org”.

An example of the resultant sip-uri after encryption is:

sip:98yudFG45tx_89TYGedb4ujF;iv=FGD567kjhfH7d4-D;key-id=eV9kl7;alg=128-aes-gcm@mc1-encryption.3gppnetwork.org

The following is an example of unprotected XML URI content within XML attributes:

EXAMPLE:

```
Content-Type: application/pidf+xml
<?xml version="1.0" encoding="UTF-8"?>
<presence entity="sip:somebody@mcptt.org">
  <tuple id="acD4rhU87bK">
    <status>
      <affiliation group="sip:thegroup@mcptt.org" />
    </status>
  </tuple>
```

```
</presence>
```

When XML URI attribute encryption is applied, the following is an example of encrypted URIs within XML attributes:

EXAMPLE:

```
Content-Type: application/pidf+xml
<?xml version="1.0" encoding="UTF-8"?>
<presence entity="sip:c4Hrt45XG8IohRFT67vfdr3V;iv=45RtfVgHY23k8Ihy;key-id=b7UJv9;alg=128-aes-
gcm@mc1-encryption.3gppnetwork.org">
  <tuple id="acD4rhU87bK">
    <status>
      <affiliation group="sip:98yudFG45tx_89TYGedb4ujF ;iv=FGD567kjhfH7d4-D;key-
id=eV9kl7;alg=128-aes-gcm@mc1-encryption.3gppnetwork.org " />
    </status>
  </tuple>
</presence>
```

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 to a whole XML MIME body.

The XML HMAC signature mechanism is specified by W3C [28]. The HMAC-SHA256 signature method shall be supported.

When integrity protection is enabled, all XML MIME bodies transported in SIP requests and responses are integrity protected. If one or more of the XML MIME bodies are included in a SIP request or SIP response, then a MIME body is included in the SIP request or SIP response containing one or more signatures pointing to those XML MIME bodies as illustrated in the figure 9.3.5-1.

In order to integrity protect the XML MIME bodies in SIP requests and SIP responses, the MC client and MCX server shall for each MIME body, include the Content-ID header field as specified in IETF RFC 2045 [40] containing a Content-ID ("cid") Uniform Resource Locator (URL) as specified in IETF RFC 2392 [41].

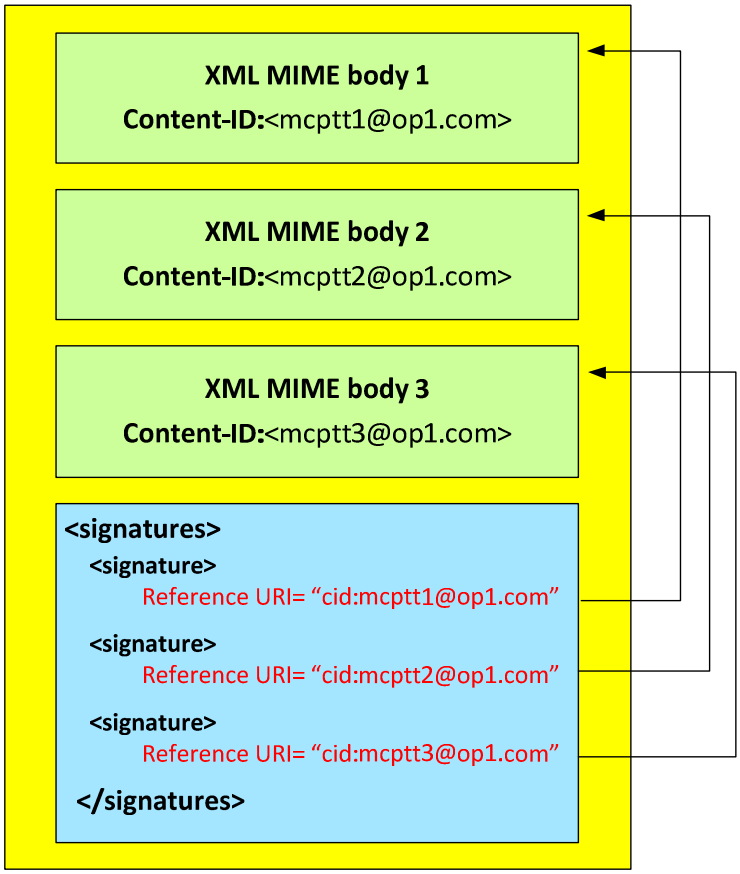


Figure 9.3.5-1: Integrity Protection of XML MIME bodies in SIP requests and SIP responses

Each MIME body that is integrity protected is assigned a unique signature contained in a <Signature> element.

The <Signature> element shall contain the following child element:

- <SignatureValue> HMAC signature of the content

The <Signature> element may contain the following child elements:

- <CanonicalizationMethod> element listing an appropriate algorithm.

- <SignatureMethod> element listing an appropriate algorithm. HMAC-SHA256 shall be supported for signatures.

- <KeyInfo><KeyName> element containing the base64 encoded XPK-ID.

- <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.

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

9.4 RTCP signalling protection (SRTCP)

9.4.1 General

RTCP encryption is required between the MC UE and MCX Server and between a pair of MCX Servers. RTCP is protected hop-by-hop, meaning that RTCP is always decrypted by the MCX server and then re-encrypted to its destination.

The following signalling uses RTCP and is protected using the procedures in this clause:

- MCPTT floor control signalling (MBCP or TBCP).
  - Unicast uplink and downlink (online), multicast downlink (online) and offline transmission.
- MCVideo transmission control (online/offline).
- Unicast uplink and downlink (online), multicast downlink (online) and offline transmission.
- MCPTT/MCVideo media signalling.
- Unicast uplink and downlink (online), multicast downlink (online) and offline transmission.
- MBMS subchannel control signalling (from MCX Server to MC UE).
  - multicast downlink (online).

All RTCP (floor control, media control and MBMS subchannel control signalling) is protected in the same way. RTCP is protected using SRTCP. The master key for SRTCP is derived from a Key For Control signalling (KFC). The KFC is shared between the transmitter and receiver(s) prior to distribution of the SRTCP packets. A 32-bit identifier for the key (KFC-ID) and a 128-bit random value (KFC-RAND) is also established.

There are a number of key distribution mechanisms for establishing the KFC based on the interface over which RTCP is being transmitted.

9.4.2 Unicast RTCP protection between client and server

In Clause 9.2.1, a Client-Server Key (CSK) is generated and shared between the MC client and MCX Server along with the CSK identifier (CSK-ID). For floor and media 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 Multicast RTCP protection between client and server

In clause 9.2.2, a Multicast Signalling Key (MuSiK) is generated and shared from the MCX Server to the MC client, along with the MuSiK identifier (MuSiK-ID). For the protection of multicast floor and media control, the KFC shall be the MuSiK and the KFC-ID shall be the MuSiK-ID. KFC-RAND shall be the MIKEY RAND value transmitted in the MIKEY message used to distribute the MuSiK.

To support legacy multicast signalling protection, the MSCCK and MKFCs may also be used for this purpose as defined in Annex H.

9.4.4 Offline floor protection

Off-network, the KFC is the PCK (for private communications) or the GMK (for group communications) as described in clause 7.3.4, and the KFC-ID is the PCK-ID or GMK-ID (respectively).

9.4.5 RTCP protection between servers

In Clause 9.2.3, a Signalling Protection Key (SPK) is shared between MCX Servers along with a SPK-ID. For floor and media control signalling transferred between MCX Servers, the KFC shall be the SPK, the KFC-ID shall be the SPK-ID and the KFC-RAND shall be the SPK-RAND.

9.4.6 Key derivation for SRTCP

As a result of the key agreement process, the entities (MCX client and server, or MCX 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 7.3.6.

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.

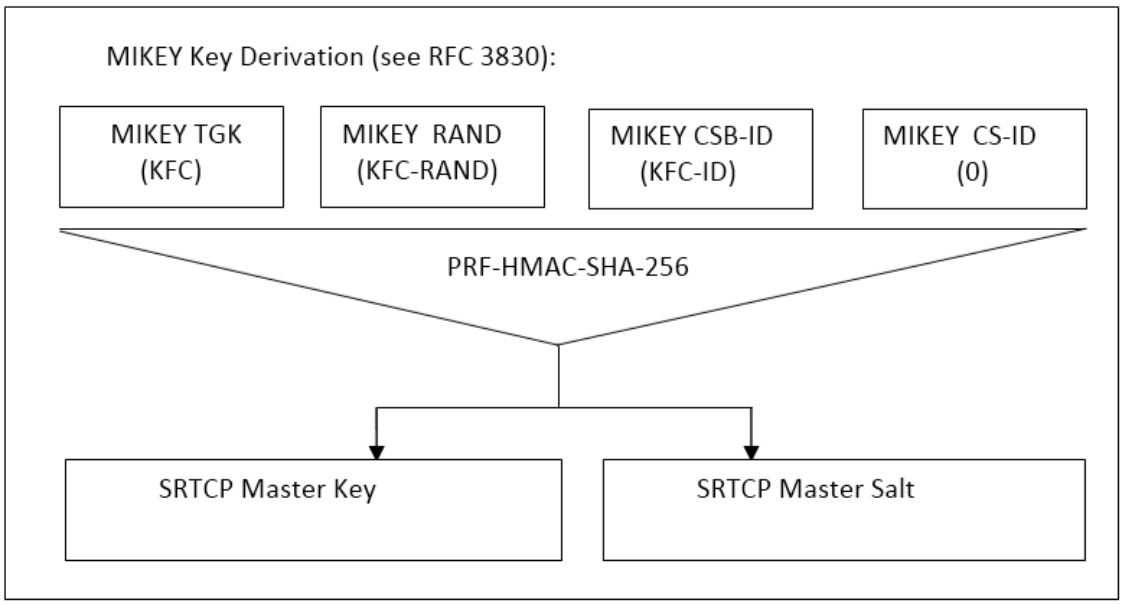


Figure 7.3.5-1: Key derivation for on-network floor and media control protection

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

9.4.7 Security procedures for transmission of RTCP content

After key establishment, RTCP protection does not require any signalling mechanism to convey information. The RTCP is protected within an SRTCP packet. The information necessary for decryption is provided within each SRTCP Packet.

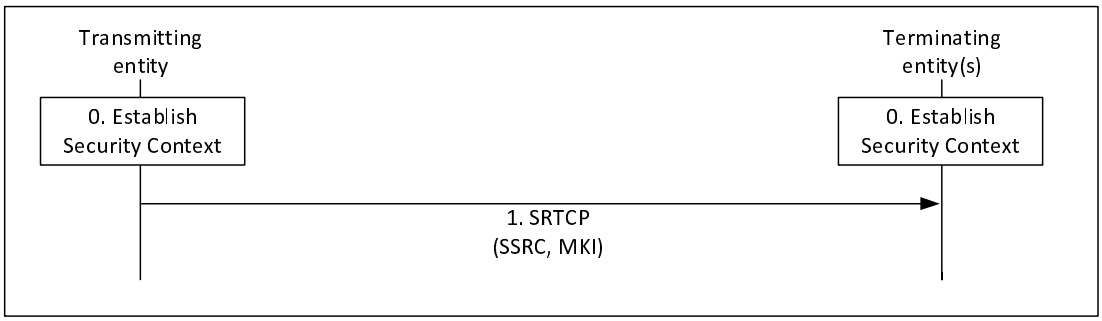


Figure 9.2.7-1: Security procedure for media stream protection

Figure 9.2.7-1 shows the security mechanism.

0) Prior to beginning this procedure the MC entities (MC UEs and/or MCX Server) involved in the communication shall have established a security context for SRTCP (Master Key, Master Salt, MKI).

1) The transmitting entity (MC UE or MCX Server) 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 entity (MC UE or MCX Server) 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 entities 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 each time it receives a SRTCP packet. The key derivation rate defined in IETF RFC 3711 [13] determines the session key generation frequency. Refer to RFC 3711 [13] for more information.

A diagram of the SRTCP packet format is within figure 9.2.7-2.

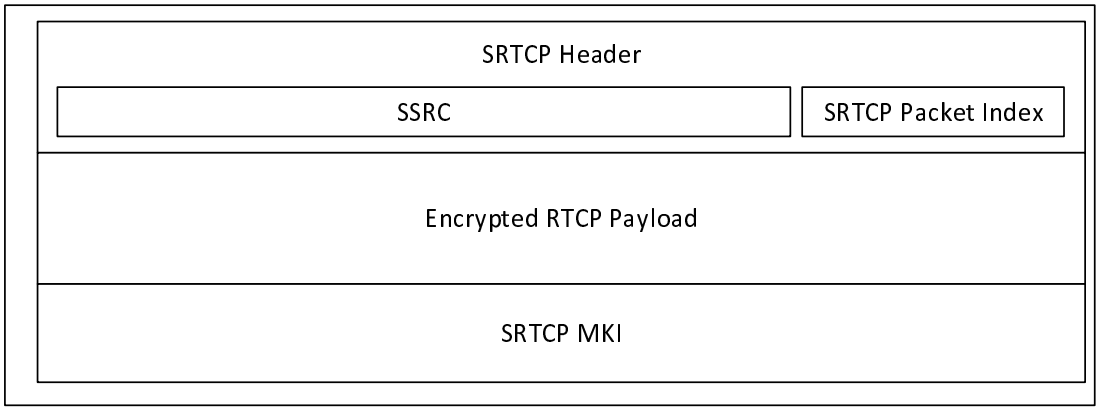


Figure 9.2.7-2: SRTCP packet format showing security parameters

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

9.4.8 RTCP protection profile

Integrity and confidentiality protection for communications using RTCP is achieved using SRTCP, as defined in IETF RFC 3711 [13]. 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 9.2.8-1.

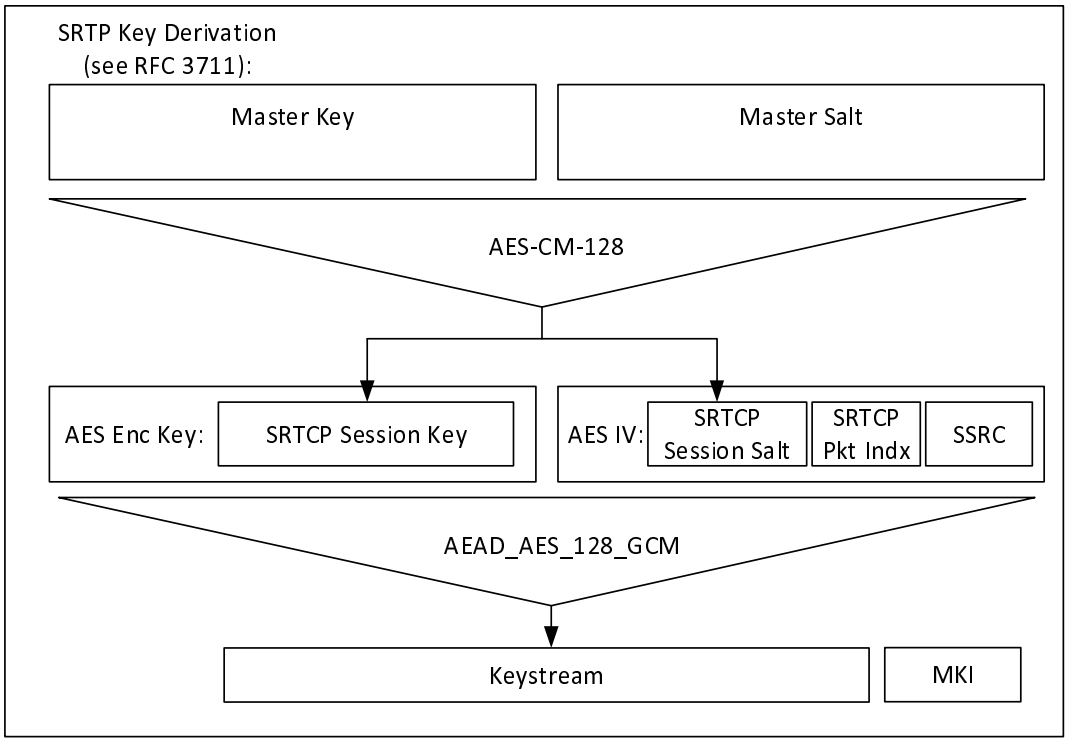


Figure 9.2.8-1: Security mechanism for floor control protection

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.

9.5 MCData signalling protection

9.5.1 Key distribution for signalling protection

Where signalling protection is required, key distribution and key use for MCData signalling is equivalent to MCPTT and MCVideo. The aim of key distribution is to establish a MCData Payload Protection Key (DPPK) per signalling channel between the communicating entities. Specifically:

- For unicast signalling transferred between the client and server, the CSK is used to protect the signalling (i.e. the DPPK is the CSK).
- For multicast signalling transferred from the server to the client, a MuSiK is used to protect the signalling (i.e. the DPPK is the MuSiK).
- For inter-server signalling, a SPK is used to protect the signalling (i.e. the DPPK is the SPK).
- MCData signalling payloads between two offline clients are protected using a PCK (e.g. the DPPK is the PCK).
- MCData signalling payloads between a group of offline clients are protected using a GMK (e.g. the DPPK is the GMK).

The procedures for CSK distribution are defined in clause 9.2.1. The procudures for MuSiK distribution are defined in clause 9.2.2. The procedures for SPK distribution are defined in clause 9.2.3.

9.5.2 Protection of MCData application signalling payloads (XML)

Protection of MCData application signalling payloads, specifically XML content within SIP messages, is defined in clause 9.3. For the protection of MCData signalling, the XPK shall be the DPPK.

9.5.3 Protection of MCData signalling payloads

Protection of MCData signalling payloads is defined in clause 8.5.

Annex A (normative): Security requirements

A.1 Introduction

Stage 1 requirements pertaining to MCX security are found in 3GPP TS 22.179 [3] and 3GPP TS 22.280 [47]. Stage 2 Architectural requirements pertaining to MCX security are found in 3GPP TS 23.179 [2], 3GPP TS 23.280 [36], 3GPP TS 23.281 [37], and 3GPP TS 23.282 [38]. The following are MCX derived security requirements:

A.2 Configuration & service access

[33.180 MCX-A.2-001] The MC UE and the network entity providing the MCX configuration data, shall mutually authenticate each other prior to MC UE configuration to use the MCX service.

[33.180 MCX-A.2-002] The MC User and the MCX Service shall mutually authenticate each other prior to providing the MC UE with the MCX Service User profile and access to user-specific services.

[33.180 MCX-A.2-003] The transmission of configuration data and user profile data between an authorized MCX server in the network and the MC UE shall be confidentiality protected, integrity protected and protected from replays.

A.3 Group key management

[33.180 MCX-A.3-001] Group key material shall be integrity and confidentiality protected for a specific MC User during distribution from the MCX service to MC UEs.

[33.180 MCX-A.3-002] Group key material shall be authenticated as coming from a valid, authorized source. The authorized source may be an MC Administrator or may be another authorized entity (e.g. an authorized MCX User or Dispatcher).

[33.180 MCX-A.3-003] 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.

[33.180 MCX-A.3-004] The creation of a new group security context (e.g. via User-Regroup operation) shall not change or compromise an existing group security context.

[33.180 MCX-A.3-005] It shall be possible for an authorized, authenticated entity to revoke and update a group security context from use.

A.4 On-network operation

[33.180 MCX-A.4-001] All users of the MCX service shall be authenticated to prevent an adversary impersonating a user for the purpose of denial of service.

[33.180 MCX-A.4-002] The MCX service should take measures to detect and mitigate DoS attacks to minimize the impact on the network and on MC users.

[33.180 MCX-A.4-003] The MC user shall be authenticated by the MCX application.

[33.180 MCX-A.4-004] A mechanism shall exist that allows the MCX application to be authenticated by the MCX user.

[33.180 MCX-A.4-005] The MC UE and MCX 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).

[33.180 MCX-A.4-006] The security solution should minimize the impact of a compromised MC UE on other MC UEs.

[33.180 MCX-A.4-007] The MCX Service shall provide a means to ensure integrity of all MCX user signalling at the application layer.

[33.180 MCX-A.4-008] The MCX Service shall protect the administrative and security management parameters from manipulation by individuals who are not explicitly authorized by the Mission Critical Organization.

[33.180 MCX-A.4-009] The MCX service shall provide a means to support confidentiality of MCX user identities from all entities outside the MCX service.

[33.180 MCX-A.4-010] The MCX service shall provide a means to support confidentiality of MCX signalling from all entities outside the MCX service.

[33.180 MCX-A.4-011] The MCX Service shall provide a means to support end-to-end confidentiality and integrity protection for all media traffic transmitted between MC UEs.

[33.180 MCX-A.4-012] The MCX Service shall provide a means to support the confidentiality and integrity protection of location information transmitted from the MC UE to the MCX application server.

A.5 Ambient listening

[33.180 MCX-A.5-001] 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.

[33.180 MCX-A.5-002] The activation of the Ambient Listening functionality shall be automatically registered by the system and will be stored as an 'event' by the system.

[33.180 MCX-A.5-003] 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.

[33.180 MCX-A.5-004] 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.6 Data communication between MCX network entities

[33.180 MCX-A.6-001] A security mechanism shall exist that allows transmission of data between 3GPP MCX 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.7 Key stream re-use

[33.180 MCX-A.7-001] The MCX system shall ensure that key streams are not reused.

A.8 Late entry to group communication

[33.180 MCX-A.8-001] An authorized MCX User shall be able to obtain the information necessary to derive the group security context for the MCX Group while an MCX Group communication is on-going. As a result, the MC User shall be able to listen to the group communication within 350ms. This requirement applies for both on-network and off-network MCX operations.

A.9 Private call confidentiality

[33.180 MCX-A.9-001] It shall be possible to establish a unique Private Call security context between any pair of authorized MCX users within the MCX system. The security context shall not be available to other MCX users, except, where necessary, authorized MCX 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 MCX UEs are operating both on-network and off-network.

[33.180 MCX-A.9-002] The Private Call security context shall provide a means to provide confidentiality and integrity protection of user traffic, and authenticate the MCX users involved in the Private Call.

A.10 Off-network operation

[33.180 MCX-A.10-001] The MCX service should take measures to detect and mitigate DoS attacks to minimize the impact to relays and to off-network MCX users.

[33.180 MCX-A.10-002] The MCX Service shall provide a means to support end-to-end security for all media traffic transmitted between MCPTT UEs, including where relays are used.

[33.180 MCX-A.10-003] The MCX Service shall provide a means to support the confidentiality and integrity protection of location information transmitted from the MCX UE to the MCX application server, including where relays are used.

[33.180 MCX-A.10-004] MCX off-network UEs shall be explicitly or implicitly authenticated to each other.

[33.180 MCX-A.10-005] MCX off-network UEs and MC relays shall be explicitly or implicitly authenticated to each other.

[33.180 MCX-A.10-006] The security solution should minimize the impact of a compromised MCX UE on other MCX UEs.

[33.180 MCX-A.10-007] The MCX Service shall provide a means to ensure integrity of all MCX user signalling at the MCX application layer.

[33.180 MCX-A.10-008] The MCX service shall provide a means to support confidentiality of MCX service user identities from all entities outside the MCX service.

[33.180 MCX-A.10-009] The MCX service shall provide a means to support confidentiality of MCX signalling from all entities outside the MCX service.

A.11 Privacy of MCX identities

[33.180 MCX-A.11-001] The MCX service user identities of each plane shall be used within the corresponding plane and concealed to other planes.

[33.180 MCX-A.11-002] When required by the MCX Service provider, MCX application services layer identities (such as the Mission Critical user identity, MCPTT ID, MCVideo ID, MCData ID and MCX Group IDs) and other application services sensitive information (as further described in 3GPP TS 23.179 [2], clause 8.2), 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.

[33.180 MCX-A.11-003] When protection of identities and other sensitive MCX application information is NOT required by the MCX Service provider, the MCX application services layer identities (such as the Mission Critical user identity, MCPTT ID, MCVideo ID, MCData ID and MCX Group IDs) and other application services sensitive information (as further described in 3GPP TS 23.179 [2], clause 8.2), shall remain contained within the application plane. While confidentiality protection is not required, integrity protection may be applied.

A.12 User authentication and authorization

[33.180 MCX-A.12-001] User authentication and authorization interoperability between different networks and different manufacturers' clients and servers shall satisfy the requirements for mission critical roaming and migration.

[33.180 MCX-A.12-002] User authentication and authorization shall support all deployment models listed in 3GPP TS 23.179 [2].

[33.180 MCX-A.12-003] User authentication and authorization shall support interchangeable MCPTT user authentication solutions, allowing implementations to use different means to authenticate the user, e.g. Web SSO, SIP digest, GBA, biometric identifiers, username+password.

[33.180 MCX-A.12-004] User authentication and authorization shall support scalability (number of users), providing efficient support for small MCPTT systems with few users, to large MCPTT systems with hundreds of thousands of users.

[33.180 MCX-A.12-005] User authentication and authorization shall support extensibility, providing authorization for additional mission critical services including group aware services, additional interfaces, etc.

[33.180 MCX-A.12-006] All users of the MCX Service shall be authenticated to prevent an adversary impersonating a user for the purpose of denial of service.

A.13 Inter-domain

[33.180 MCX-A.13-001] An MCX Service shall provide mechanisms to allow an MCX User to operate in a Partner MCX Service System, subject to authorization from both the Partner and the Primary MCX Service Systems of the MCX User (R-6.17.2-001 [47]).

[33.180 MCX-A.13-002] The authentication of an MCX User with an MCX Service in a Partner MCX Service System shall be based on security parameters obtained from the Primary MCX Service System of the MCX User (R-6.17.2-002 [47]).

NOTE 1: This is an application layer authentication and not 3GPP network authentication.

[33.180 MCX-A.13-003] An MCX Service shall provide mechanisms to allow an MCX User on the Primary MCX Service System to affiliate to an MCX Service Group from a Partner MCX Service System, subject to authorization from the Primary MCX Service System and the Partner MCX Service System where the MCX Service Group is defined (R-6.17.2-004 [47]).

[33.180 MCX-A.13-004] An MCX Service shall provide mechanisms to allow a roaming MCX User to affiliate to an MCX Service Group from the Partner MCX Service System, subject to authorization from the Partner MCX Service System where the MCX Service Group is defined (R-6.17.2-005 [47]).

[33.180 MCX-A.13-005] An MCX Service shall provide mechanisms to allow an MCX User that receives service from a Partner MCX Service System to affiliate to an MCX Service Group from another Partner MCX Service System, subject to authorization from the Partner MCX Service System where the MCX Service Group is defined (R-6.17.2-006 [47]).

NOTE 2: It is assumed that once affiliation from a User to a Group is successful, subsequent communication within that Group are available to the User.

[33.180 MCX-A.13-006] End to end security of an MCX Service Group communication (including in Partner MCX Service Systems) shall be based on parameters obtained from the MCX Service system where the MCX Service Group is defined (R-6.17.2-007 [47]).

[33.180 MCX-A.13-007] All Mission Critical Users shall be authenticated with their home identity management service prior to authentication or authorisation with a partner domain.

[33.180 MCX-A.13-008] A user requiring services at a partner domain shall first acquire a verifiable credential from the user’s primary identity management service.

[33.180 MCX-A.13-009] An identity management service shall authenticate a visiting user based on a verifiable credential from the user’s primary identity management service prior to authorising that user for local service(s).

[33.180 MCX-A.13-010] A visiting user shall be authorised with the local server(s) at the partner MCX System before being granted local services.

[33.180 MCX-A.13-011] The partner identity management service shall have full and overruling authorisation control of all visiting users requesting services in the partner MCX System.

[33.180 MCX-A.13-012] When using external security domains, the Home Security Domain shall apply policies which ensure that only trusted external security domains are used.

[33.180 MCX-A.13-013] Use of external security domains shall be logged to detect impersonation and misuse.

[33.180 MCX-A.13-014] MCX Services shall be able to permit/deny the use of security domains over their service.

A.14 MCData

[33.180 MCX-A.14-001] The MCData Service shall provide a means to support end-to-end confidentiality and integrity protection for messaging transmitted between MCX UEs in both media and signalling streams.

[33.180 MCX-A.14-002] The MCData Service shall provide a means to authenticate messages in both media and signalling streams.

A.15 Multimedia Broadcast/Multicast Service

[33.180 MCX-A.15-001] The security of signalling transmitted between the MCX client and MCX server shall be controlled by the MCX server. As a consequence of this requirement, the MCX Server shall not require key material from external MC Domains to enable the use of MBMS.

[33.180 MCX-A.15-002] The MCX Service shall provide means to support confidentiality and integrity protection for the MBMS subchannel control messages.

Annex B (normative): OpenID connect profile for MCX

B.1 General

The information in this annex provides a normative description of the MCX 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. MCX Connect profiles the OpenID Connect standard and includes the service IDs in the ID token and the access token, as well as the definition of MCX specific scopes for key management, MCX services, configuration management, and group management. This profile is compliant with OpenID Connect.

B.2 MCX tokens

B.2.1 ID token

B.2.1.1 General

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

B.2.1.2 Standard claims

These standard claims are defined by the OpenID Connect 1.0 specification and are REQUIRED for MCX implementation. Other claims defined by OpenID Connect are optional. The standards-based claims for an MC ID token are shown in table B.2.1.2-1.

Table B.2.1.2-1: 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 MCX user within the MCX server provider's domain.
Aud	REQUIRED. The Oauth 2.0 client_id of the MCX 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.2.1.3 MCX claims

The MCX Connect profile extends the OpenID Connect standard claims with the additional claims shown in table B.2.1.3-1.

Table B.2.1.3-1: ID token MCX claims

Parameter	Description
mcptt_id	REQUIRED for MCPTT. The MCPTT ID of the current MCPTT user of the MCPTT client.
mcvideo_id	REQUIRED for MCVideo. The MCVideo ID of the current MCVideo user of the MCVideo client.
mcdata_id	REQUIRED for MCData. The MCData ID of the current MCData user of the MCData client.

B.2.2 Access token

B.2.2.1 Introduction

The access token is opaque to MCX clients and is consumed by the MCX resource servers (i.e. KMS, MCPTT server, MCVideo server, MCData 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.2.2.2 Standard claims

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

Table B.2.2.2-1: 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 MCX authorization scopes associated with this token. The scope(s) contained here reflect the requested scope(s) from the Authentication Request (clause B.4.2.2).
client_id	REQUIRED. The identifier of the MCX client making the API request as previously registered with the IdM server.

B.2.2.3 MCX claims

The MCX Connect profile extends the standard claims defined in IETF RFC 7662 [33] with the additional claims shown in table B.2.2.3-1.

Table B.2.2.3-1: Access token MCX claims

Parameter	Description
mcptt_id	REQUIRED for MCPTT. The MCPTT ID of the current MCPTT user of the MCPTT client.
mcvideo_id	REQUIRED for MCVideo. The MCVideo ID of the current MCVideo user of the MCVideo client.
mcdata_id	REQUIRED for MCData. The MCData ID of the current MCData user of the MCData client.

B.3 MCX client registration

Before an MCX client can obtain ID tokens and access tokens (required to access MCX 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 MCX 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 MCX client.

- Registration of the client's redirect URIs.

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

B.4 Obtaining tokens

B.4.1 General

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

B.4.2 Native MCX client

B.4.2.1 General

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

MCX clients fitting the Native application profile utilize the authorization code grant type with the PKCE extension for enhanced security as shown in figure B.4.2.1-1.

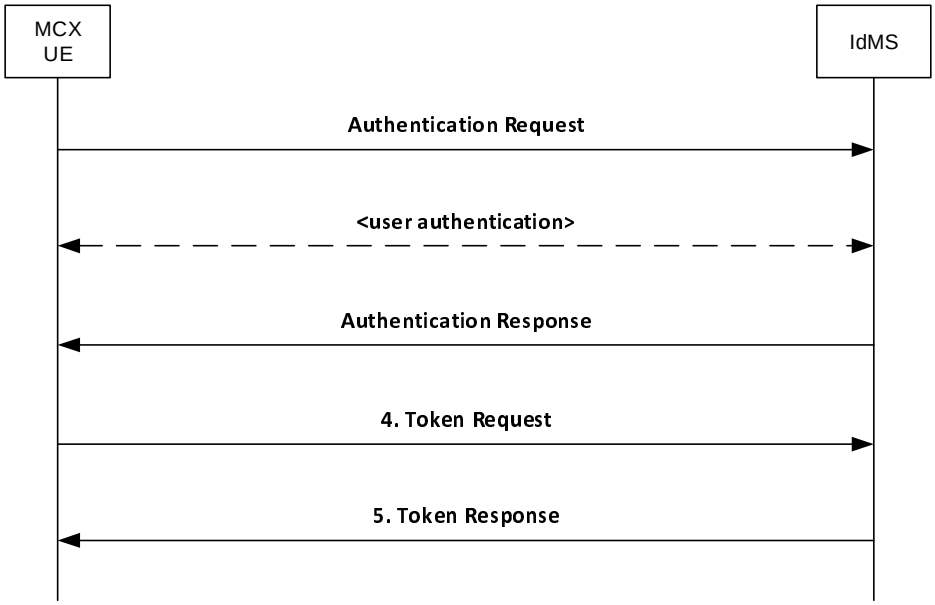


Figure B.4.2.1-1: Authorization Code flow

B.4.2.2 Authentication request

As described in OpenID Connect 1.0, the MCX 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.4.2.2-1 are required by the MCX Connect profile. Other parameters defined by the OpenID Connect specification are optional.

Table B.4.2.2-1: Authentication Request standard required parameters

Parameter	Values
response_type	REQUIRED. For native MCX clients the value shall be set to "code".
client_id	REQUIRED. The identifier of the MCX client making the API request. It shall match the value that was previously registered with the IdM server of the MCX service provider.
scope	REQUIRED. Scope values are expressed as a list of space-delimited, case-sensitive strings which indicate which MCX resource servers the client is requesting access to (e.g. MCPTT, MCVideo, MCData, 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 MCX client. This profile further defines the following additional authorization scopes: - ″3gpp:mc:ptt_service″ - ″3gpp:mc:video_service″ - ″3gpp:mc:data_service″ - ″3gpp:mc:ptt_key_management_service″ - ″3gpp:mc:video_key_management_service″ - ″3gpp:mc:data_key_management_service″ - ″3gpp:mc:ptt_config_management_service″ - ″3gpp:mc:video_config_management_service″ - ″3gpp:mc:data_config_management_service″ - ″3gpp:mc:ptt_group_management_service″ - ″3gpp:mc:video_group_management_service″ - ″3gpp:mc:data_group_management_service″ Others may be added in the future as new MCX resource servers are introduced by 3GPP (see note).
redirect_uri	REQUIRED. The URI of the MCX client to which the IdM server will redirect the MCX client's user agent in order to return the authorization code to the MCX 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 MCX client to maintain state between the authorization request and authorization response. The IdM server includes this value in its authorization response back to the MCX 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 MCX Connect might look like:

EXAMPLE:

```
GET/as/authorization.oauth2?response_type=code&client_id=mcptt_client&scope=openid 3gpp:mc:ptt_servi
ce&redirect_uri=http://3gpp.mcptt/cb&state=abc123&acr_values=3gpp:acr:password&code_challange=0x1234
56789abcdef&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 MCX client, the IdM server performs user authentication. Note that user authentication is completely opaque to the MCX client (which never sees any of it, as it is run directly between the IdM server and the user-agent on the UE).

B.4.2.3 Authentication response

The authorization endpoint running on the IdM server issues an authorization code and delivers it to the MCX client. The authorization code is used by the MCX 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.4.2.3-1.

Table B.4.2.3-1: Authentication Response standard required parameters

Parameter	Values
code	REQUIRED. The authorization code generated by the authorization endpoint and returned to the MCX 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 MCX Connect might look like.

EXAMPLE:

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

B.4.2.4 Token request

In order to exchange the authorization code for an ID token, access token and refresh token, the MCX 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.4.2.4-1.

Table B.4.2.4-1: 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 MCX 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 provisioned via a 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 MCX 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.4.2.5 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 MCX 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 MCX client may now validate the user with the ID token and configure itself for the user (e.g. by extracting the MC service ID from the ID Token). The MCX client then uses the access token to make authorized requests to the MCX resource servers (MCPTT server, MCVideo server, MCData server, KMS, etc.) on behalf of the end user.

B.5 Refreshing an access token

B.5.1 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 MCX clients and web-based MCX clients). Refresh tokens are not given to clients utilizing the implicit grant type (browser-based MCX clients). Figure B.5.1-1 shows how Native MCX clients can use the refresh token as a grant type to obtain new access tokens.

Figure B.5.1-1: Requesting a new access token

B.5.2 Access token request

To obtain an access token from the IdM server using a refresh token, the MCX client makes an access token request to the token endpoint of the IdM server. The MCX 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.5.2-1.

Table B.5.2-1: 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 MCX 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 MCX client as part of the original authorization request.

An example of a token request for MCX 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 MCX client was provided with client credentials by the 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.5.3 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 MCX client, and optionally another refresh token.

An example of a successful response for MCX 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.6 MCX client registration with partner IdM service

MCX client registration with a partner IdM service shall be as described in clause B.3.

B.7 Obtaining an access token from a partner domain

B.7.1 Overview

When an MCX user requires user service authorisation for services owned and managed within a partner domain, the MCX client shall use the OAuth 2.0 token exchange extension grant type mechanism to obtain a security token for authentication with the partner IdM service. The OAuth 2.0 token exchange procedure defines a method for obtaining the security token from the primary IdMS which contains information about the user that is verifiable by the partner IdMS.

The MCX client then provides this security token to the partner IdM service in exchange for an access token that is specific to the services in the partner domain. The MCX UE then uses the access token for user service authorisation to those services within the partner domain.

The security token and acess token(s) are specific to a IdMS and partner domain and therefore the OAuth 2.0 token exchange procedure shall be repeated with each additional domain to obtain user service authorisation to partner services within those domains.

Figure B.7.1-1 shows the OAuth 2.0 token exchange procedure used to obtain a security token and acess token(s). The messages are described in the following sub-clauses.

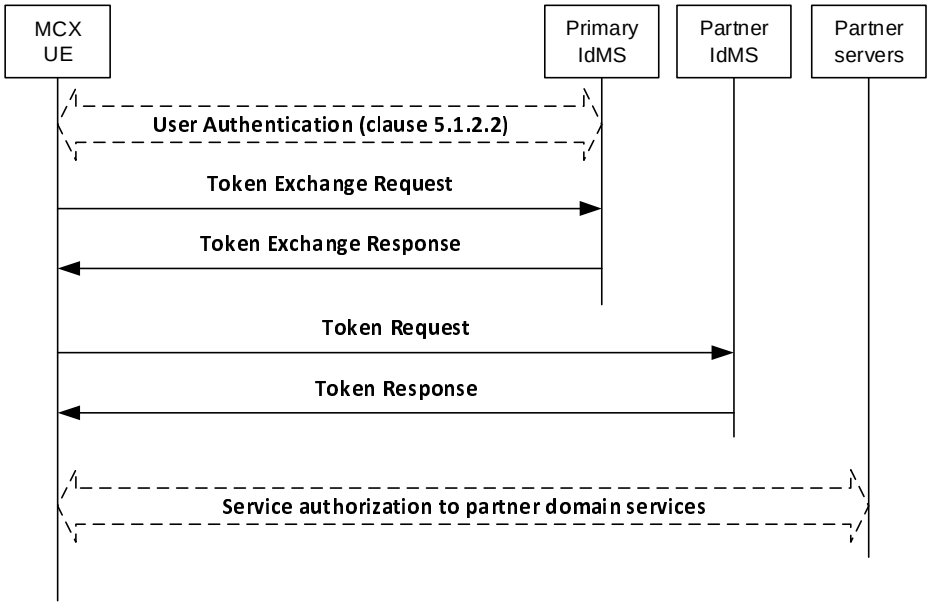


Figure B.7.1-1: Token exchange flow

B.7.2 Token Exchange Request

In order to obtain a security token, the MCX client makes a request to the primary authorization server's token endpoint by sending the following parameters using the "application/x-www-form-urlencoded" content-type and a character encoding of UTF-8 in the HTTP request entity-body. The standard parameters shown in table B.7.2-1 are required by the MCX Connect profile.

Table B.7.2-1: Token Exchange Request standard required parameters

Parameter	Values
grant_type	REQUIRED. The value shall be set to “urn:ietf:params:oauth:grant-type:token-exchange” indicating that a token exchange is being performed.
subject_token	REQUIRED. A token that represents the identity of the party on behalf of whom the request is being made. This shall be the access token previously obtained in the token response message (clause B.4.2.5) during authorisation (clause B.4).
subject_token_type	REQUIRED. An identifier that indicates the type of the security token in the subject_token parameter. The value shall be set to “urn:ietf:params:oauth:token-type:jwt” indicating the access token is a JSON Web Token.

An example of a successful token exchange request might look like:

EXAMPLE:

```
POST /as/token.oauth2 HTTP/1.1
Host: IdM.server.com:9031
Authorization: Basic cnA33hpsb25nABClY3VyZS1yYW5kb20tc2VjdnV0
Content-Type: application/x-www-form-urlencoded
grant_type=urn%3Aietf%3Aparams%3Aoauth%3Agrant-type%3Atoken-exchange
&subject_token=baaR3jcJyb4BWCxGsndq23ScbdFMogUC5Pb233jKLTC
&subject_token_type=
urn%3Aietf%3Aparams%3Aoauth%3Atoken-type%3Aaccess_token
```

B.7.3 Token Exchange Response

Upon successfully receiving and validating the token exchange request message from the MCX client, the IdM server shall return a token exchange response containing a security token specific to the partner IdMS.

The token exchange response standard parameters are shown in table B.7.3-1.

Table B.7.3-1: Token exchange response standard required parameters

Parameter	Values
access_token	REQUIRED. This is the security token specific to the partner IdMS.
issued_token_type	REQUIRED. This field shall be “urn:ietf:params:oauth:token-type:jwt”
token_type	REQUIRED. This field shall be “bearer”
expires_in	RECOMMENDED. The lifetime in seconds of the security token.

An example of a successful token exchange response might look like:

EXAMPLE:

```
HTTP/1.1 200 OK
   Content-Type: application/json
   Cache-Control: no-cache, no-store
  {
   "access_token":"eyJhbGciOiJFUzI1NiIsImtpZCI6IjllciJ9.eyJhdWQiOiJo
   dHRwczovL2JhY2tlbmQuZXhhbXBsZS5jb20iLCJpc3MiOiJodHRwczovL2FzLmV
   4YW1wbGUuY29tIiwiZXhwIjoxNDQxOTE3NTkzLCJpYXQiOjE0NDE5MTc1MzMsIn
   N1YiI6ImJjQGV4YW1wbGUuY29tIiwic2NwIjpbImFwaSJdfQ.MXgnpvPMo0nhce
   PwnQbunD2gw_pDyCFA-Saobl6gyLAdyPbaALFuAOyFc4XTWaPEnHV_LGmXklSTp
   z0yC7hlSQ",
   "issued_token_type":"urn:ietf:params:oauth:token-type:jwt",
   "token_type":"Bearer",
   "expires_in":600
  }
```

B.7.4 Token Request

In order to exchange the security token for an access token and (optional) refresh token, the MCX client makes a request to the partner 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 authentication of the security token is REQUIRED in order to exchange the security token for an access token. The security token shall be transported in the HTTP Authorization Header. The token request standard parameters are shown in table B.7.4-1.

Table B.7.4-1: Token Request standard required parameters

Parameter	Values
grant_type	REQUIRED. This value shall be set to "urn:ietf:params:oauth:grant-type:jwt-bearer" as per rfc 7523 [46].
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 provisioned via a development portal.
scope	REQUIRED. Scope values are expressed as a list of space-delimited, case-sensitive strings which indicate which MCX resource servers the client is requesting access to at the partner system (e.g. MCPTT group services, MCVideo group services, MCData group services, etc.). If authorized, the requested scope values will be bound to the access token returned to the client in the token exchange response message. The scope shall include one or more of the following: - ″3gpp:mc:ptt_service″ - ″3gpp:mc:video_service″ - ″3gpp:mc:data_service″ - ″3gpp:mc:ptt_key_management_service″ - ″3gpp:mc:video_key_management_service″ - ″3gpp:mc:data_key_management_service″ - ″3gpp:mc:ptt_config_management_service″ - ″3gpp:mc:video_config_management_service″ - ″3gpp:mc:data_config_management_service″ - ″3gpp:mc:ptt_group_management_service″ - ″3gpp:mc:video_group_management_service″ - ″3gpp:mc:data_group_management_service″ Others may be added in the future as new MCX resource servers are introduced by 3GPP.

Examples of a successful token request might look like:

EXAMPLE 1:

```
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=urn%3Aietf%3Aparams%3Aoauth%3Agrant-type%3Ajwt-bearer&
client_id=myNativeApp&scope=openid 3gpp:mc:ptt_group_management_service&
```

EXAMPLE 2:

```
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=urn%3Aietf%3Aparams%3Aoauth%3Agrant-type%3Ajwt-
bearer&client_id=myNativeApp&scope=openid 3gpp:mc:ptt_service,3gpp:mc:ptt_key_management_service
,3gpp:mc:ptt_config_management_service,3gpp:mc:ptt_group_management_service&
```

B.7.5 Token Response

If the token request is valid and authorized, the partner IdM server returns an access token to the MCX client specific to the user for the partner services and optionally a refresh token; 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.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
}
```

The MCX client then uses the access token to make authorized requests to the partner MCX resource servers (MCPTT group management service, MCVideo group management service, MCData group management service, etc) on behalf of the end user.

B.8 Security tokens

Security tokens obtained from the primary IdMS and used for authentication with a partner IdMS shall conform to the id token requirements and format described in clause B.2.1.

B.9 Access tokens for partner services

Access tokens obtained from a partner IdMS and used for user service authorisation to services within the partner domain shall conform to the access token requirements and format described in clause B.2.2.

B.20 Using the token to access MCX resource servers

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

The access token is opaque to the MCX 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 MCX resource servers. If the access token is presented to an MCX resource server and the scope is invalid or the token is expired or revoked, the MCX resource server should return an error message indicating such to the MCX client.

B.21 Token validation

B.21.1 ID token validation

The MCX client shall validate the ID token as per section 3.1.3.7 of the OpenID Connect 1.0 specification [21].

B.21.2 Access token validation

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

B.21.3 Security token validation

The IdM server shall validate the security token as per section 3.1.3.7 of the OpenID Connect 1.0 specification [21].

Annex C (informative): OpenID connect detailed flow

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

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

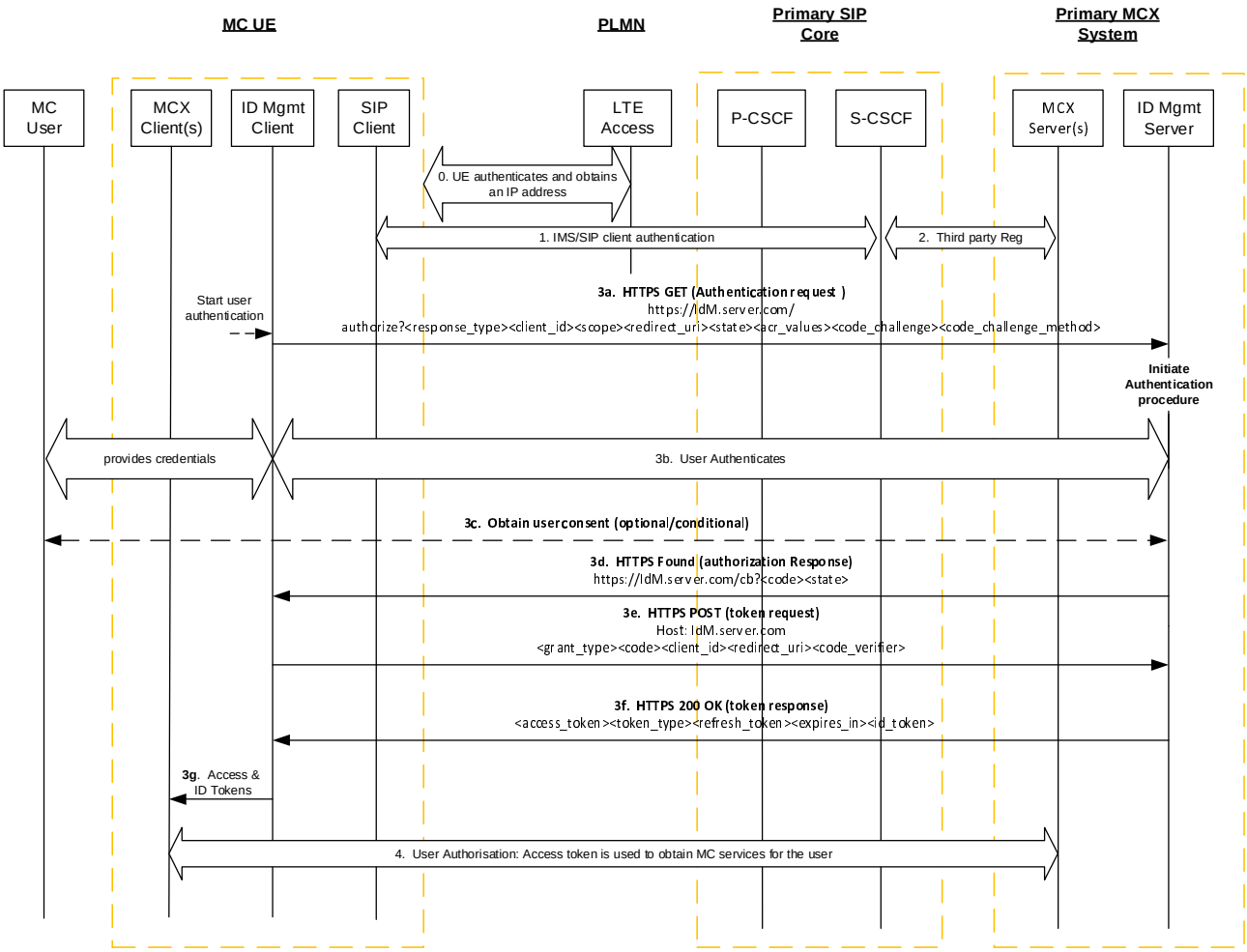


Figure C.1-1: OpenID Connect MC User Authentication and Registration

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/SIP Client authenticates with the primary IMS/SIP core. For IMS authentication, 3GPP TS 33.203 [9] applies.

Step 2: The SIP core sends a SIP 3rd Party Registration to the MCX application Server(s), notifying them of the MC 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 MC network. The client includes the code_challenge value in this request.

Step 3b: The MC User Identity and associated credentials are provided to the IdM server. The credentials are successfully authenticated (and optionally authorized) by the IdM Server.

Step 3c: The IdM Server may optionally request user consent for granting the MCX client access to the MCX service in the MCX 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 MC user and MCX service(s)) 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 made available to the MCX client(s).

Step 4: The MC UE performs user service authorization.

C.2 Detailed flow for inter-domain MC user service authorization using OpenID Connect token exchange

Figure C.2-1 shows the detailed message flow for inter-domain MCX user authentication and service authorisation using the OpenID Connect token exchange method as described in Annex B.

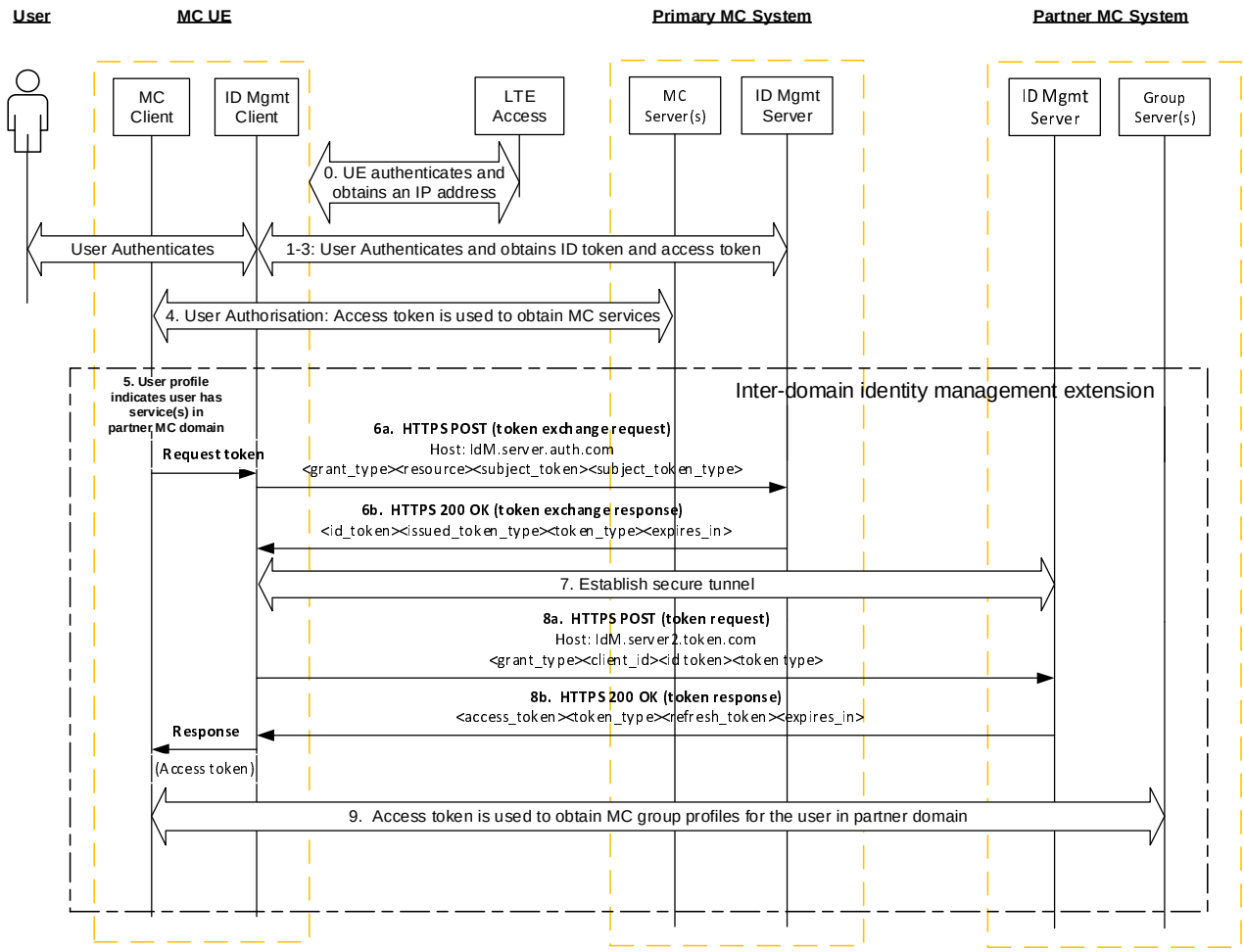


Figure C.2-1: Inter-domain user authentication and service authorisation

Steps 0-3: These steps are the same as described in steps 0-3 of Figure C.1-1, which provide the initial network access, network security, HTTPS tunnel to IdM server, user authentication, IMS authentication, and SIP registration.

Step 4: This step represents the culmination of steps C-1 through C-5 in Figure 5.1.3.1-1, which authorises the user for services in the primary domain. As part of this step the UE obtains the user's profile, which specifies both the local (primary domain) and the non-local (partner domain) group services.

Step 5: From the user's profile, the UE identifies group service(s) home to a partner domain. The user profile includes metadata of the group service(s) and information about the partner IdMS (i.e. the token endpoint host address and the "aud" parameter for use in the token exchange request).

Step 6a: Based on the OAuth token exchange procedure, the UE IdM Client performs a HTTP POST (token exchange) request to the user's primary IdM Server token endpoint. This request consists of the access token obtained in step 3 and information about the partner IdMS (i.e. the "aud" parameter obtained from the user profile group metadata).

Step 6b: The primary IdM Server token endpoint verifies the access token and returns a security token specific to the partner IdM Server.

Step 7: The UE establishes a secure HTTP tunnel with the partner IdM token endpoint using HTTPS.

Editor's Note: It is FFS how the TLS tunnel between the visiting user and the partner systems IdM server is authenticated.

Step 8a: The UE IdM Client performs a HTTP POST token request to the partner IdM token endpoint to exchange the security token for an access token. This message is defined in [19].

Step 8b: The partner IdM Server token endpoint verifies the security token and issues an access token specific to the user and the user's local MC group service(s).

NOTE 1: Additional access tokens may be requested as needed by repeating steps 8a and 8b.

Step 9: For each group service, the GM client in the UE follows the "Retrieve group configurations at the group management client" flow as shown in clause 10.1.5.2 of TS 23.280 [36], presenting an access token in the Get group configuration request over HTTP. If the access token is valid, the GMS authorises the user for the specific group management service. Completion of this step results in the GMS sending the user's group policy information and group key information to the GM client. This step is repeated for each additional group service that is home to this partner domain.

NOTE 2: Steps 5–9 are repeated for user service authorization to services in each additional partner domain.

Annex D (Normative): KMS provisioning messages

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 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.3.1 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), and may respond with a new TrK. The data is returned within a "KMSInit" tag.

In response to a "KMS KeyProvision" request, the KMS shall provision appropriate user Key Sets within a "KMSKeyProv" tag, and may also respond with a new TrK.

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.3.2 KMS certificates

D.3.2.1 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 at any one time, 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 user is managed by a single KMS. The root certificate for this KMS is required to encrypt messages to the user, and verify signatures from the user.

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

D.3.2.2 Fields

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

Table D.3.2.2-1: 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.3.2.3 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 Mission Critical Services as MC Service IDs are not Tel URIs.

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

D.3.3 User Key Provision

D.3.3.1 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.3.3.2 Fields

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

Table D.3.3.2-1: 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.3.4 Example KMS response XML

D.3.4.1 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 key management client have shared a bootstrap TrK, e.g. 'tk.11.user@example.org'.

In this example, the KMS provides the 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.3.4.2 Example KMSKeyProv XML

In this example, the 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.3.4.3 Example KMSCertCache XML

In this example, a number of 'external' KMS certificates are provided to the 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.3.5 KMS response XML schema

D.3.5.1 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.3.5.2 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.1.1 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 MC UE and Private Call Keys (PCKs) between MC UEs.

The GMK is encrypted to the UID generated from the receiving user's MC Service user 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 MC Service user ID and current time period. It is signed using the UID generated from the initiating user's MC Service user 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.1.2 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 MCX 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 MC Service user IDs (i.e. MCPTT ID, MCVideo ID, MCData ID) 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 MC Service user 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 MC Service user ID associated with the initiating user. The IDRr payload shall contain the MC Service user ID associated to the receiving user. The message shall also include IDRkmsi and IDRkmsr that contains the URI of the KMS used by the initiating user and terminating user respectively.

NOTE: In some deployments MC Service user IDs within these payloads may treated as private. In this case, the initiating and terminating MC 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 MC Service user 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 MC Service user 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 MC Service user ID associated with the initiating user. The IDRr payload shall contain the MDSI of the MCX Domain. The message shall also include IDRkmsi and IDRkmsr that contains the URI of the KMS used by the initiating user and MCX Server respectively.

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

The SAKKE payload shall encapsulate the CSK to the UID generated from the MDSI of the MCX 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 MC Service user ID of the initiating user.

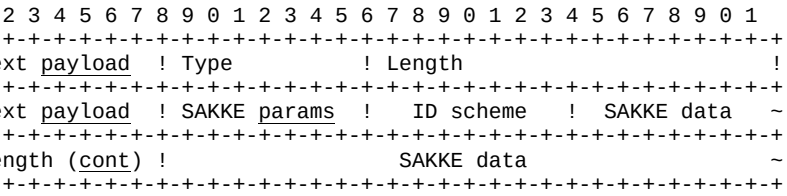
E.5 MIKEY general extension payload to support 'SAKKE-to-self'

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 MCX service via more than one MC UE, the other MC 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 MC 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;
- MC 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.

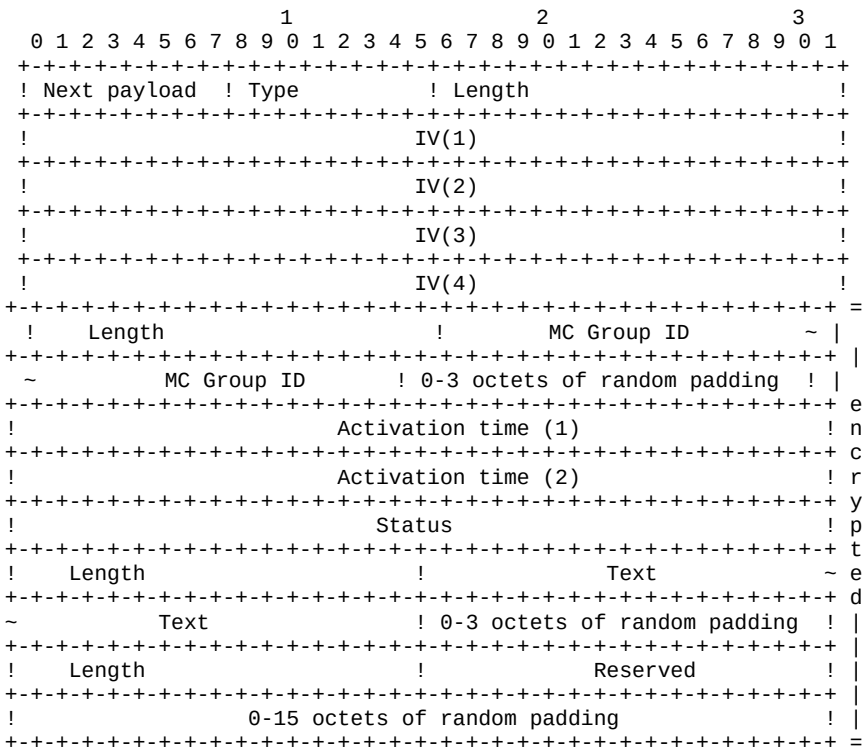


Figure E.6.1-1: Layout of general extension payload for associated parameters of GMK

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 MC group ID

The 'MC 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 'MC 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 'MC 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 'MC group ID' sub-element, 0 -3 octets of random padding shall be added so that the total length of the ('Length' sub-element + 'MC 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 'MC 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 MC Service user IDs in transit. To protect these identifiers in MIKEY-SAKKE messages the following approach may be taken.

The sensitive MC Service user ID in the IDRr or IDRi field is replaced with the UID generated from the MC Service user 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 MC Service user ID, it shall ensure that the SIP message containing the I_MESSAGE contains the initiator's MC Service user 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 MC Service user 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 MC Service user ID. On obtaining the initiator's MC Service user 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 [17] 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 [17].

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 = MC Service user ID.
- L0 = length of above (i.e. 0x00 0x17).

The GMK and MC Service user 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 [17].

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 = MC Service user 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 MC Service user ID and XPK-ID follow the encoding also specified in annex B of 3GPP TS 33.220 [17].

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.1.5 Calculation of keys for MCData payload protection

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 DPPK:

- FC = 0xaa, (for MCData Payload Protection),
- P0 = DPPK-ID.
- L0 = length of above, expressed in number of bytes (i.e. 0x00 0x17).

The DPPK-ID follow the encoding also specified in annex B of 3GPP TS 33.220 [17].

Where the DPPK is 128-bits, the DPCK 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 DPPK is 256-bits, the output DPCK 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 MC Service user IDs may not be Tel-URIs, this UID format cannot be used within MC applications 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.2 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, MCVideo ID or MCData 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 1: The key derivation function defined in clause B.2 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 2: 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 3: 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 (normative): Key identifiers

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 MC client and MC domain.
- 3: the SPK shall be used to protect application signalling (XML and SRTCP) between servers in MC domain(s).
- 4: The MKFC may be used as defined in Annex H.
- 5: The MSCCK may be used as defined in Annex H.
- 6: The MuSiK shall be used to protect multicast signalling between the MCX Server and the MC Client.
- 7-15: not defined

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

Annex H (normative): Support for legacy multicast keys (MKFC and MSCCK)

H.1 General

TS 33.179 [7] specified a different key distribution mechanism for the distribution of group multicast keys (MKFC). To allow MCPTT clients to operate with legacy MCPTT servers (as defined by the functionality in TS 33.179 [7]), MCPTT clients shall support the MKFC key distribution mechanisms defined in clause H.2 with the following constraints:

- The MCPTT client shall reject MKFCs received from other MC systems (based upon the GMS identity).
- The MCPTT client shall discard previously received MKFCs upon attaching to a new MC system.

MCPTT Servers shall not support MKFC distribution. Should a MCPTT Server be provided with a MKFC from a legacy GMS, the MCPTT Server shall reject the MKFC. For this reason, the MCPTT Server shall only support transmission of signalling over a unicast bearer to a legacy MCPTT client (as defined by the functionality TS 33.179 [7]), This shall be detected by the MCPTT Server on the rejection of the MuSiK.

MCPTT Servers and MCPTT clients shall support distribution of the MSCCK. The mechanism for the distribution of the MSCCK is defined in clause H.3.

NOTE: The MSCCK and MSCCK-ID are equivalent to the MuSiK and MuSiK-ID, with the sole exception that the MSCCK may only be used to protect MBMS subchannel control messages, and hence the MSCCK-ID has a different key identifier.

MSCCK and MKFC are used as defined in clause H.4.

H.2 MKFC Receipt

MKFCs are distributed using the same procedures as for GMK distribution. The client receives an MKFC from the MCPTT server using the procedures in clause 7.3, with the exception that the MKFC and MKFC-ID is distributed in the place of the GMK and GMK-ID, and the user salt is zero (meaning that the GUK-ID is the MKFC-ID).

MKFCs are either distributed in their own group key distribution message (separate from the GMK distribution message), or in the same distribution message as the GMK. Distributing the MKFC in the same message as the GMK is achieved by embedding two MIKEY payloads in one distribution message.

H.3 MSCCK Distribution

MSCCK and MSCCK-ID are distributed within MBMS bearer announcement messages. The procedures are identical to those for distribution of the MuSiK, as defined in clause 5.9, with the exception that the MSCCK and MSCCK-ID are distributed instead of the MuSiK and MuSiK-ID.

H.4 Use of multicast signalling keys (MKFC and MSCCK)

For the protection of multicast floor and media control received from legacy MCPTT servers, 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 KFC. The KFC is used as defined in clause 9.4.6 and 9.4.7.

For the protection of MBMS subchannel control messages from a legacy MCPTT server, or to a legacy MCPTT client, the KFC shall be the MSCCK and the KFC-ID shall be the MSCCK-ID. KFC-RAND shall be the MIKEY RAND value transmitted in the MIKEY message used to distribute the KFC. The KFC is used as defined in clause 9.4.6 and 9.4.7.

Annex I (informative): Change history

Change history							
Date	**Meeting**	**TDoc**	**CR**	**Rev**	**Cat**	**Subject/Comment**	**New version**
2017-06	SA#76	SP-170419				Presented for approval	2.0.0
2017-06	SA#76					Upggrade to change control version	14.0.0

History

Document history		
V14.0.0	July 2017	Publication