ETSI TS 133 303 V12.7.0 (2016-08)

Universal Mobile Telecommunications System (UMTS);
LTE;
Proximity-based Services (ProSe);
Security aspects
(3GPP TS 33.303 version 12.7.0 Release 12)

Reference
RTS/TSGS-0333303vc70

Keywords
LTE,SECURITY,UMTS

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

Important notice

The present document can be downloaded from:
http://www.etsi.org/standards-search

The present document may be made available in electronic versions and/or in print. The content of any electronic and/or print versions of the present document shall not be modified without the prior written authorization of ETSI. In case of any existing or perceived difference in contents between such versions and/or in print, the only prevailing document is the print of the Portable Document Format (PDF) version kept on a specific network drive within ETSI Secretariat.

Users of the present document should be aware that the document may be subject to revision or change of status.
Information on the current status of this and other ETSI documents is available at
https://portal.etsi.org/TB/ETSIDeliverableStatus.aspx

If you find errors in the present document, please send your comment to one of the following services:
https://portal.etsi.org/People/CommiteeSupportStaff.aspx

Copyright Notification

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 aspects of the Proximity Services (ProSe) features in EPS.
Based on the common security procedures (clause 5) for

- interfaces between network entities (using NDS),
- configuration of ProSe-enabled UEs, and
- data transfer between the ProSe Function and a ProSe enabled UE (PC3 interface)

security for the following ProSe features is covered:

- Open ProSe Direct Discovery in network coverage (clause 6.1);
- One-to-many ProSe direct communication for ProSe-enabled Public Safety UEs (clause 6.2);
- EPC-level Discovery of ProSe-enabled UEs (clause 6.3);
- EPC support for WLAN Direct Discovery and Communication (clause 6.4).

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 23.303: "Proximity-based services (ProSe); Stage 2".

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

[4] 3GPP TS 33.310: "Network Domain Security (NDS); Authentication Framework (AF)".

[5] 3GPP TS 33.220: "Generic Authentication Architecture (GAA); Generic Bootstrapping Architecture (GBA)".

[6] ETSI TS 102 225: "Smart Cards; Secured packet structure for UICC based applications".

[7] ETSI TS 102 226: "Smart cards; Remote APDU structure for UICC based applications".

[8] 3GPP TS 31.115: "Secured packet structure for (Universal) Subscriber Identity Module (U)SIM Toolkit applications".

[9] 3GPP TS 31.116: "Remote APDU Structure for (U)SIM Toolkit applications ".

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

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

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

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

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

[15] NIST FIPS 186-4: "Digital Signature Standard (DSS)".

[16] BSI TR-03111: "Technical Guideline TR-03111; Elliptic Curve Cryptography".

[17] IETF RFC 5639: "Elliptic Curve Cryptography (ECC) Brainpool Standard; Curves and Curve Generation".

[18] IETF RFC 3339: "Date and Time on the Internet: Timestamps".

[19] IETF RFC 5280: "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile".

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

[21] 3GPP TS 33.401: "3GPP System Architecture Evolution (SAE); Security architecture".

[22] 3GPP TS 33.222: "Generic Authentication Architecture (GAA); Access to network application functions using Hypertext Transfer Protocol over Transport Layer Security (HTTPS)".

[23] IETF RFC 3261: "SIP: Session Initiation Protocol".

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

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

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

[27] IETF RFC 3339: "Date and Time on the Internet: Timestamps".

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

[29] IETF Draft draft-ietf-avtcore-srtp-aes-gcm-17: "AES-GCM and AES-CCM Authenticated Encryption in Secure RTP (SRTP)".

[30] Void.

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

[32] 3GPP TS 33.328: "IP Multimedia Subsystem (IMS) media plane security".

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

[34] IETF RFC 3986: "Uniform Resource Identifier (URI): Generic Syntax".

[35] IETF RFC 4563: "The Key ID Information Type for the General Extension Payload in Multimedia Internet KEYing (MIKEY)".

[36] W3C REC-xmlschema-2-20041028: "XML Schema Part 2: Datatypes".

[37] IETF RFC 2616: "Hypertext Transfer Protocol -- HTTP/1.1".

[38] 3GPP TS 33.223: "Generic Authentication Architecture (GAA); Generic Bootstrapping Architecture (GBA) Push function".

[39] 3GPP TS 23.003: "Numbering, addressing and identification".

[40] 3GPP TS 36.331: "Evolved Universal Terrestrial Radio Access (E-UTRA); Radio Resource Control (RRC); Protocol specification".

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, the terms and definitions given in 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 TR 21.905 [1].

Discovery Filter: See 3GPP TS 23.303 [2]

ProSe Application ID: See [2]

ProSe Application Code: See [2]

ProSe Application Mask: See [2]

ProSe Direct Communication: See [2]

ProSe Direct Discovery: See [2]

ProSe-enabled non-Public Safety UE: See [2]

ProSe-enabled Public Safety UE: See [2]

ProSe-enabled UE: See [2]

Validity Timer: See [2]

3.2 Abbreviations

For the purposes of the present document, the abbreviations given in 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 TR 21.905 [1].

ADF	Accounting Data Forwarding
ALUID	Application Layer User ID
AS	Application Server
BSF	Bootstrapping Server Function
CA	Certificate Authority
CTF	Charging Trigger Function
DSA	Digital Signature Algorithm
ECDSA	Elliptic Curve DSA
EPUID	EPC Level User ID
GBA	Generic Bootstrapping Architecture
GMK	Group Master Key
GPS	Global Positioning System
GSK	Group Session Key
ID	Identity
KMS	Key Management System
LCID	Logical Channel Identifier
MIC	Message Integrity Code
MIKEY	Multimedia Internet Keying
NAF	Network Application Function
NITZ	Network Identity and Time Zone
NTP	Network Time Protocol
OTA	Over The Air
PEK	ProSe Encryption Key
PFID	ProSe Function ID
PGK	ProSe Group Key
ProSe	Proximity-based Services
PTK	ProSe Traffic Key

RTP	Real-Time Transport Protocol
RTCP	RTP Control Protocol
SDP	Session Description Protocol
SEG	Security Gateway
SRTP	Secure Real-Time Transport Protocol
UID	User ID
UTC	Universal Time Coordinated

4 Overview of ProSe security

4.1 General

The overall architecture for ProSe is given in TS 23.303 [2]. ProSe includes several features that may be deployed independently of each other. For this reason, no overall security architecture is provided and each feature describes its own architecture.

Although made of several different features, those features share many procedures, for example, both ProSe Direct Discovery and ProSe Direct Communication utilize the same procedure for service authorization (see TS 23.303 [2]). Security for this common procedures are described in clause 5 of the present document, while the overall security of the ProSe features is described in clause 6 of the present document (which refers back to clause 5 as necessary).

4.2 Reference points and Functional Entities

PC8: The reference point between the UE and the ProSe Key Management Function. PC8 relies on EPC user plane for transport (i.e. an "over IP" reference point). It is used to transport security material to UEs for ProSe one-to-many communications.

5 Common security procedures

5.1 General

This clause contains a description of the security procedures that are used by more than one ProSe feature.

5.2 Network domain security

5.2.1 General

ProSe uses several interfaces between network entities, e.g. PC4a between the ProSe Function and the HSS (see TS 23.303 [2]). This subclause describes the security for those interfaces.

5.2.2 Security requirements

The ProSe network entities shall be able to authenticate the source of the received data communications.

The transmission of data between ProSe network entities shall be integrity protected.

The transmission of data between ProSe network entities shall be confidentiality protected.

The transmission of data between ProSe network entities shall be protected from replays.

5.2.3 Security procedures

For all interfaces between network elements,

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

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

NOTE: For the case of an interface between two entities in the same security domain, TS 33.210 [3] does not mandate the protection of the interface by means of IPsec.

5.3 Security of UE to ProSe Function interface

5.3.1 General

The ProSe-enabled UEs have many interactions with the ProSe Function over the PC3 in the ProSe features described in TS 23.303 [2].

5.3.2 Security requirements

Only the ProSe Function may provide configuration data impacting the ProSe-related network operations to the ProSe-enabled UE. 3rd parties shall not be allowed to provide such parameters.

The ProSe-enabled UE and the ProSe Function shall mutually authenticate each other.

The transmission of configuration data between the ProSe Function and the ProSe-enabled UE shall be integrity protected.

The transmission of configuration data between the ProSe Function and the ProSe-enabled UE shall be confidentiality protected.

The transmission of configuration data between the ProSe Function and the ProSe-enabled UE shall be protected from replays.

The configuration data shall be stored in the UE in a protected way to prevent modification.

Some configuration data may require to be stored in the UE in a protected way to prevent eavesdropping.

The transmission of UE identity should be confidentiality protected on PC3 interface.

5.3.3 Security procedures

5.3.3.1 Security procedures for configuration transfer to the UICC

After deployment of the ProSe-enabled UE the configuration parameters stored in the UICC may need to be updated to reflect the changes in the configuration applied.

In case that configuration data of ProSe-enabled UE are stored in the UICC, the UICC OTA mechanism (as specified in ETSI TS 102 225 [6] / TS 102 226 [7] and 3GPP TS 31.115 [8] / TS 31.116 [9]) shall be used to secure the transfer of the configuration data to be updated in the UICC.

5.3.3.2 Security procedures for data transfer to the UE

This subclause describes procedures for protecting data transfer between UE and ProSe Function (called the network function in the below procedures).Between the UE and network function,

for UE initiated messages, the procedures specified by clause 5.4 of TS 33.222 [22] shall be used with the following addition. The network function may optionally include an indication in the PSK-identity hint in the ServerKeyExchange message over the Ua interface to inform the UE of the FQDN of the BSF with which the UE shall run the bootstrapping procedure over the Ub interface as specified in TS 33.220[5] to provide the key material for establishment of the TLS tunnel. When performing such bootstrapping with the indicated BSF, the UE and BSF shall use the provided FQDN as the BSF Identity in all places, e.g. forming the B-TID. If there is no such indication, the UE shall perform the bootstrapping with the BSF at the address given in TS 23.003 [39].

The UE may also hold a B-TID, Ks and other associated material from bootstrapping runs with different BSFs simultaneously.

A network function that implements the NAF functionality (cf Annex F.2 or Annex F.3) shall request USSs from the BSF when requesting the Ks_(ext/int)_NAF key and the network function shall check in the USS if the USIM is authorized to be used for ProSe services. If the authorization in the network function fails then the network function shall release the PSK-TLS connection with the UE. Otherwise (cf Annex F.4) the retrieval of

authentication vectors and authorization of the ProSe UE shall be performed using the PC4a interface instead of the Zh interface. If the ProSe Function does not have a unused Authentication vector associated with the ProSe UE, the ProSe Function shall request one Authentication Vector from the HSS over the PC4a interface. ProSe Function shall always indicate to the UE, in the Ua message carrying the ServerKeyExchange of the PSK-TLS handshake, that the Ks_NAF key shall be used to bootstrap the PSK-TLS security on the PC3 interface, by setting the psk_identity_hint field to a static string "3GPP-bootstrapping".

NOTE 1: Annex F describes the possible network options for PC3 security. The UE behaviour remains the same regardless of the network option used.

NOTE 2: When the termination points of both the Ua and Ub interfaces reside in the network function then the implementation of the Zn interface is an operator decision. However, if the operator considers deploying a stand-alone BSF for use by the network function, then the operator should use the canonical BSF name from TS 23.003 and the Zn interface should be available in the network function already from the start although it would only be used internally to the network function until a stand-alone BSF was deployed. Otherwise, implementation changes to the network function will become necessary at the time of introducing the stand-alone BSF.

For network initiated messages one of the following mechanisms shall be used:

- If a PSK TLS connection has been established as a part of a pull message and is still available, the available PSK TLS session shall be used.

- Otherwise, PSK TLS with GBA push based shared key-based mutual authentication between the UE and the network function shall be used. GBA push is specified in TS 33.223 [38]. The network function (pushNAF) shall request USSs from the BSF when requesting a GPI, and the network function shall check in the USS if the USIM is authorized to be used for ProSe services. If the authorization in the network function fails then the network function shall refrain from establish PSK TLS with GBA push.

NOTE 3: If a TLS connection is released, it can only be re-established by the client, i.e. UE, even though the TLS session including security association would be alive on both sides. TLS connection, in turn, is dependent on the underlying TCP connection.

6 Security for ProSe features

6.1 ProSe direct discovery

6.1.1 Overview of ProSe direct discovery in network coverage

The Direct Discovery in network coverage feature in clause 5.2 and 5.3 of TS 23.303 [2] describes several procedures. These are the following:

1. The **Service Authorization** procedure: The UE contacts the ProSe Function(s) in theHPLMN in order to obtain authorization to use direct discovery in the various PLMNs. This step also includes the ability of the HPLMN to revoke authorization via a push message.

2. The **Discovery Request** procedure: This allows an announcing or a monitoring UE to obtain the necessary configuration information to be able to announce a code or monitor for codes in a particular PLMN. Among other things, the configuration information includes the ProSe Application Codes to be announced or Discovery Filters to be monitored for.

3. The **Discovery** procedure: The ProSe App Code is announced by the announcing UE and received by the monitoring UE.

4. The **Match Report** procedure: This allows a ProSe App Code received by the monitoring UE to be checked and confirmed by the network. As part of this procedure the network provides the UE with the ProSe Application Id Name of that code and possibly the meta-data corresponding to that ProSe Application ID Name.

6.1.2 Security requirements

Procedures 1, 2, and 4 include traffic between the UE and ProSe Function and between other network entities, therefore the requirements in clause 5.3.2 and 5.2.2 apply to this case.

In addition, for the overall discovery procedure, the following security requirement applies: The system shall support a method to mitigate the replay and impersonation attacks for ProSe open discovery.

6.1.3 Security procedures

6.1.3.1 Interface between the UE and ProSe Function

In order to protect the messages between the UE and ProSe Function over PC3, the UE shall support the procedures for the UE given in subclause 5.3.3.2 and the ProSe Function shall support the procedures for the network function given in subclause 5.3.3.2. In order to protect the messages between the UE and ProSe Function CTF (ADF) (Accounting Data Forwarding function block of the Charging Trigger Function) over PC3ch, the same communication security shall apply for PC3ch as is used for PC3.

6.1.3.2 Interfaces between network elements

This uses the procedures described in clause 5.2.3 of the present document.

6.1.3.3 Integrity protection and validation of the transmitted code for open discovery

6.1.3.3.1 Open discovery security flows

The message flows apply when both the UEs are roaming or when one or both are in their HPLMN.

Note that integrity protection via this Message Integrity Check (MIC) furthermore enables the ProSe Function to verify that the announcing UE was indeed authorized to announce this ProSe App Code at that time instance. A UTC-based counter associated with the discovery slot is used to calculate the MIC and verify the MIC. To help ensure that the announcing and monitoring UEs end up with the same value of the UTC-based counter, the announcing UE includes the

4 least significant bits of its couter value in the discovery message, which the monitoring UE uses when setting the value of the UTC-based counter that is passed to the ProSe Function.

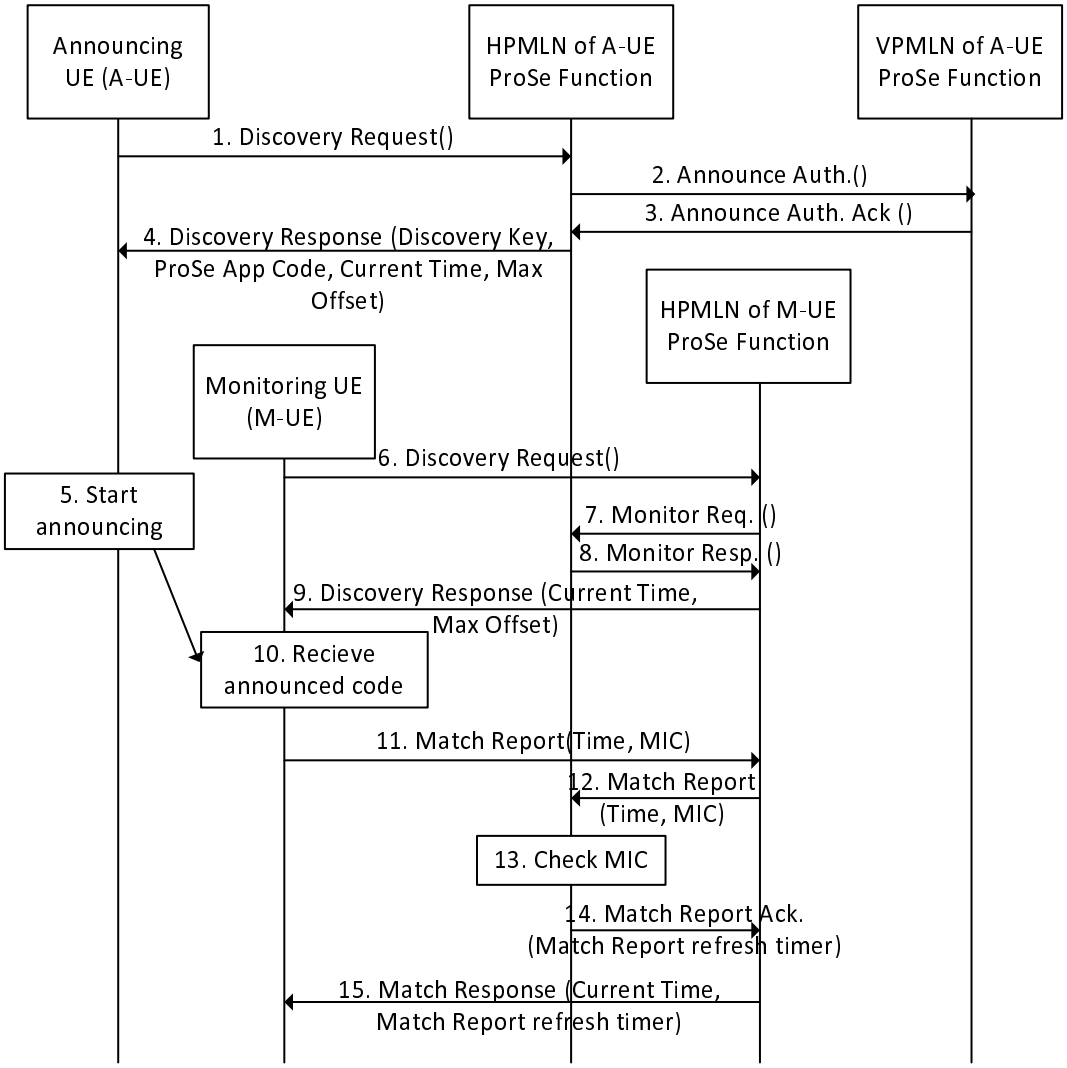

Figure: 6.1.3.3.1-1: Integrity protection of the transmitted code

1. The Announcing UE sends a Discovery Request message containing the ProSe Application ID to the ProSe Function in its HPLMN in order to be allowed to announce a code on its serving PLMN (either VPLMN or HPLMN).

2./3. The ProSe Functions in the HPLMN and VPLMN of the announcing UEs exchange Announce Auth. messages. There are no changes to these messages for the purpose of protecting the transmitted code for open discovery. If the Announcing UE is not roaming, these steps do not take place.

4. The ProSe Function in HPLMN of the announcing UE returns the ProSe App Code that the announcing UE can announce and a 128-bit Discovery Key associated with it. The ProSe Function stores the Discovery Key with the ProSe App Code. In addition, the ProSe Function provides the UE with a CURRENT_TIME parameter, which contains the current UTC-based time at the ProSe Function, a MAX_OFFSET parameter, and a Validity Timer (see TS 23.303 [2]). The UE sets a clock it uses for ProSe authentication (i.e. ProSe clock) to the value of CURRENT_TIME and stores the MAX_OFFSET parameter, overwriting any previous values. The announcing UE obtains a value for a UTC-based counter associated with a discovery slot based on UTC time. The counter is set to a value of UTC time in a granularity of seconds. The UE may obtain UTC time from any sources available, e.g. the RAN via SIB16, NITZ, NTP, GPS (depending on which is available).

NOTE 1: The UE may use unprotected time to obtain the UTC-based counter associated with a discovery slot. This means that the discovery message could be successfully replayed if a UE is fooled into using a time different to the current time. The MAX_OFFSET parameter is used to limit the ability of an attacker to successfully replay discovery messages or obtain correctly MICed discovery message for later use. This is achieved by using MAX_OFFSET as a maximum difference between the UTC-based counter associated with the discovery slot and the ProSE clock held by the UE.

NOTE 2: A discovery slot is the time at which an announcing UE sends the announcement.

5. The UE starts announcing, if the UTC-based counter provided by the system associated with the discovery slot is within the MAX_OFFSET of the announcing UE's ProSe clock and if the Validity Timer has not expired (see TS 23.303 [2]).For each discovery slot it uses to announce, the announcing UE calculates a 32-bit Message Integrity Check (MIC) to include with the ProSe App Code in the discovery message. Four least significant bits of UTC-based counter are transmitted along with the discovery message. The MIC is calculated as described in sub clause A.2 using the Discovery Key and the UTC-based counter associated with the discovery slot.

6. The Monitoring UE sends a Discovery Request message containing the ProSe Application ID to the ProSe Function in its HPLMN in order to get the Discovery Filters that it wants to listen for.

7/8. The ProSe Functions in the HPLMN of the monitoring UE and HPLMN of the announcing UEs exchange Monitor Req./Resp. messages. There are no changes to these messages for the purpose of protecting the transmitted code for open discovery.

9. The ProSe Function returns the Discovery Filter containing either the ProSe App Code(s), the ProSe App Mask(s) or both along with the CURRENT_TIME and the MAX_OFFSET parameters. The UE sets its ProSe clock to CURRENT_TIME and stores the MAX_OFFSET parameter, overwriting any previous values. The monitoring UE obtains a value for a UTC-based counter associated with a discovery slot based on UTC time. The counter is set to a value of UTC time in a granularity of seconds. The UE may obtain UTC time from any sources available, e.g. the RAN via SIB16, NITZ, NTP, GPS (depending on which is available).

10. The Monitoring UE listens for a discovery message that satisfies its Discovery Filter, if the UTC-based counter associated with that discovery slot is within the MAX_OFFSET of the monitoring UE's ProSe clock.

11. On hearing such a discovery message, and if the UE has either not checked the MIC for the discovered ProSe App Code previously or has checked a MIC for the ProSe App Code and the associated Match Report refresh timer (see steps 14 and 15 for details of this timer) has expired, or as required based on the procedures specified in TS 23.303 [2], the Monitoring UE sends a Match Report message to the ProSe Function in the HPLMN of the monitoring UE. The Match Report contains the UTC-based counter value with four least significant bits equal to four least significant bits received along with discovery message and nearest to the monitoring UE"s UTC-based counter associated with the discovery slot where it heard the announcement, and other discovery message parameters including the ProSe App Code and MIC.

12. The ProSe Function in the HPLMN of the monitoring UE passes the discovery message parameters including the ProSe App Code and MIC and associated counter parameter to the ProSe Function in the HPLMN of the announcing UE in the Match Report message.

13. The ProSe Function in the HPLMN of the announcing UE shall check the MIC is valid. The relevant Discovery Key is found using the ProSe App Code.

14. The ProSe Function in the HPLMN of the announcing UE shall acknowledge a successful check of the MIC to the ProSe Function in the HPLMN of the monitoring UE in the Match Report Ack message. The ProSe Function in the HPLMN of the announcing UE shall include a Match Report refresh timer in the Match Report Ack message. The Match Report refresh timer indicates how long the UE will wait before sending a new Match Report for the ProSe App Code.

15. The ProSe Function in the HPLMN of the monitoring UE returns an acknowledgement that the integrity checked passed to the Monitoring UE. The ProSe Function returns the parameter ProSe Application ID to the UE.
It also provides the CURRENT_TIME parameter, by which the UE (re)sets its ProSe clock The ProSe Function in the HPLMN of the monitoring UE may optionally modify the received Match Report refresh timer based on local policy and then shall include the Match Report refresh timer in the message to the Monitoring UE..

6.2 Security for One-to-many ProSe direct communication

6.2.1 Overview of One-to-many ProSe direct communication

The One-to-many ProSe direct communication consists of the following procedures based on TS 23.303 [2]:

1. One-to-many ProSe Direct communication transmission;
2. One-to-many ProSe Direct communication reception.

The functionality in this clause may only be supported by ProSe-enabled Public Safety UEs.

Security for one-to-many ProSe direct communication consists of bearer level security mechanism (specified in subclause 6.2.3) and media plane security mechanism (specified in subclause 6.2.4).

6.2.2 Security requirements

The requirements in clause 5.3.2 apply for the signalling between the UE and ProSe Function.

For the distribution of the keys, the following requirements apply:

- The shared keys and session keys shall be protected in integrity and confidentiality during their distribution.
- Only authorized Public Safety ProSe-enabled UEs shall receive the shared keys.
- It shall be possible for the Public Safety ProSe-enabled UE to authenticate the network entity distributing the shared keys.
- The Public Safety ProSe-enabled UE should support authentication of the group member distributing the session keys.
- It shall be possible for the Public Safety ProSe-enabled UE to store shared keys for past and future cryptoperiods.
- The mechanism for distributing session keys shall support late entry to group communications.
- Authorized Public Safety Prose-enabled UEs shall securely store the shared keys.

For the protection of the data transmission between the UEs, the following requirements apply

- The system shall support providing the Public Safety ProSe-enabled UEs with the all the necessary keying material and chosen algorithms that are used to protect the data sent between the Public Safety ProSe-enabled UE(s). This material shall be provided without requiring signalling between the Public Safety ProSe-enabled UEs.
- Confidentiality of one-to-many communications should be supported. Its use would be a configuration option related to network operations and should hence be under control of the network operator.
- Integrity protection of one-to-many communications shall be supported for media plane security mechanism.
- Security mechanisms shall scale effectively to large groups, and be compatible with rapid setup of group communications.
- Security mechanism shall support multiple logical channels between the same source/destination pair. Security mechanism shall avoid key stream repetition (COUNTs is about to be re-used with the same key), when multiple PDCP entities exist in the Public Safety ProSe-enabled UE.

6.2.3 Bearer layer security mechanism

6.2.3.1 Security keys and their lifetimes

As part of the TLS protected signalling between the UE and ProSe Key Management Function, a ProSe MIKEY Key (PMK) may be sent from the ProSe Key Management Function to the UE. This key is used to protect the MIKEY messages that are used to carry the PGKs to the UE. The usage of PMK to protect the MIKEY messages is described in clause 6.2.3.3.2.3. The UE keeps the PMK until it receives a new one from the ProSe Key Management Function.

A UE needs to have an algorithm identity and a PGK (ProSe Group Key) provisioned for each group that they belong to. From this key, a UE that wishes to broadcast data shall first generate a PTK (ProSe Traffic Key). The parameters used in this generation ensure that PTKs are unique for each UE and need to be transferred in the header of the user data packet (see below for more information). The PTK is derived when the first PDCP entity for a group is created and then PDCP entities created further for the same group shall use the PTK derived for the group.

From the PTK, a UE derives the needed ProSe Encryption Key (PEK) to be able to encrypt the data. The UE can protect the data to be sent with the relevant keys and algorithms at the bearer level (see clause 6.2.3.3 for more details).
A receiving UE would need to derive the PTK using the information in the bearer header and then the PEK used to decrypt the data.

When the PGKs are provided to the UE, they shall be provided with an Expiry Time. The Expiry Time of the PGK needs to be set such that the keys for later periods have a longer expiration period. Each PGK for each group shall be associated with a different Expiry Time value.

When protecting data that is to be sent, the UE uses the PGK that has not expired and with the earliest expiration time to derive the PTK etc, for that group. When receiving protected data the UE shall only use a PGK that has not expired or the PGK that has most recently expired. All other expired PGK(s) should be deleted.

When a PGK key is deleted in the sending UE and receiving UE, all related keys as PTK and PEK derived from the expired PGK shall be deleted as well as the related PGK Identity, PTK Identity and Counter. After releasing all the PDCP entities of a group, the PTK and PEK derived for the group is deleted.

6.2.3.2 Identities

The ProSe Key Management Function sends to the UE a PMK along with a 64 bit PMK identity. The UE uses both the PMK identity and the FQDN of the ProSe Key Management Function to identify the PMKs locally (e.g., PMK_id@FQDN). The ProSe Key Management Function shall only allocate currently (and locally) unused PMK identities.

The PGKs are specific to a particular group and hence have a Group Identity associated with that group. This Group Identity is referred to as "ProSe Layer-2 Group ID" in TS 23.303 [2] and is 24 bits long. In addition, each PGK associated with a group has 8-bit PGK Identity to identify it. This allows several PGKs for a group to be held simultaneously as each can be uniquely identified. When allocating PGK ID, the ProSe Key Management Function shall ensure that all allocated PGKs that have not expired shall be uniquely identifiable by the 5 least significant bits of the PGK ID. This means that the combination of Group Identity and PGK Identity uniquely identifies a PGK. The Group Identity is the destination Layer 2 identity of the group. An all zero PGK Identity is used to signal special cases between the UE and ProSe Key Management Function, and hence is never used to identify a PGK.

Each member of a group has a unique 24-bit Group Members Identity, identifying a UE within a group and referred to as "ProSe UE ID" in TS 23.303 [2]. This is used a part of the PTK derivation to ensure each user generates unique PTKs for protecting the data that they send. The Group Members Identity is the source Layer 2 identity when the UE sends data.

The PTK identity shall be a 16-bit counter set to a unique value in the sending UE that has not been previously used together with the same PGK and PGK identity in the UE. Every time a new PTK needs to be derived, the PTK Identity counter is incremented.

A PTK is uniquely identified by the combination of Group Identity, PGK Identity, Group Member Identity of the sending UE and a 16-bit PTK identity. The PTK Identity is used as part of the derivation of PTK to ensure that all PTKs are unique. Under a particular PGK, the PTK identities are used in order starting with 1.

A Logical Channel ID (LCID) associated with the PDCP/RLC entity is used as an input for ciphering in order to avoid key stream repetition (i.e., to avoid counter being re-used with the same PEK by one or more PDCP entities corresponding to a group).

A 16 bit counter is maintained per PDCP entity. Counter and LCID ensures key stream freshness across the transmission by multiple PDCP entities of the same group. The counter is same as the PDCP SN in regular LTE.

For each group that the UE is a member of, the ME shall store a value of PTK ID and counter in either the USIM or non-volatile memory on the ME to prevent the re-use of the same values with a LCID under a PGK in case the UE unexpectedly powers down. These stored values shall be associated with the PGK that is being used to send the data.

After power on but before sending any one-to-many data for a group, the ME shall handle the PTK ID and counter from the USIM or non-volatile memory of the ME as follows. The ME shall copy the values PTK ID and counter into volatile memory.

NOTE 1: The values stored in volatile memory represent the smallest values of PTK ID and Counter that the UE knows have not been used with currently unused LCIDs.

For USIM storage of PTK ID and counter, the ME shall also increase the PTK ID in the USIM by 3 if it is less than 2^16-4 or to 2^16 – 1 otherwise, and set the value of counter to 2^16-1 (its maximum value). If storage in non-volatile memory of the ME is used, the ME shall keep the value of PTK ID in the non-volatile memory of the ME the same, and set the counter to 2^16-1.

NOTE 2: The PTK ID on the USIM is set higher than if it was held in non-volatile memory of the ME to reduce the number of writes to the USIM. It is not set to the maximal value in both cases as this would invalidate a PGK for a possibly out of coverage UE.

When a new PDCP entity is created for sending traffic, the UE shall select a currently unused LCID. If a previously unused PGK is to be used to provide the keys for protecting this PDCP entity, then the UE acts as below. Otherwise the ME selects a PTK Identity and counter values to use with the new PDCP entity, such that no larger PTK ID has been used for this PGK and LCID and no larger counter values have been used with this PGK, PTK ID and LCID.

NOTE 3: It is enough for the ME to use the values stored in the volatile memory of the ME to ensure keystream freshness, but more sophisticated methods may allow more efficient use of PTK ID and counters.

If a previously unused PGK is to be used with the PDCP entity, then for a PGK already stored on the USIM, the ME sets the PTK identity and counter on the USIM to 3 and 2^16-1 respectively and associates them with this PGK. For a new PGK stored in non-volatile memory in the ME, the ME shall set the PTK identity and counter in the non-volatile memory of the ME to one and 2^16-1 respectively and associates them with this PGK. The ME shall set both the PTK ID and counter in volatile memory to one. The ME shall use the new PGK, a PTK ID of one and a counter of one to protect the traffic on the PDCP entity.

To encrypt the data for a PDCP entity, the ME shall calculate PTK (as described in Annex A.3) and then PEK from PTK (as described in Annex A.4). The ME then uses the PEK, LCID, PTK ID and counter to encrypt the next data packet as described in subclause 6.2.3.6.1. Immediately after encrypting the data packet, the ME shall increase the counter associated with the PDCP entity by one. If this causes the counter to wrap, then the ME shall behave as follows:

- If PTK ID < 2^16-1, then the ME shall increase the PTK ID associated with the PDCP entity by one and set the counter associated with this PDCP entity to one. Futhermore for USIM storage of PTK ID, the ME shall increase the PTK ID stored on the USIM by 3 if it is less than 2^16-4 or to 2^16-1 otherwise if the stored PTK ID in USIM would be less than the one about to be used in ME. If non-volatile memory on the ME is used to store the PTK ID, the ME shall increase the PTK ID in non-volatile memory by one.

- If PTK ID = 2^16-1 (i.e. PTK ID would wrap) and if the next PGK is previously unused (i.e. does not have the PTK ID and Counter in either the USIM or non-volatile memory of the ME associated with it), the ME shall act as though it just created a new PDCP entity with a previously unused PGK.

- Otherwise (i.e. PTK ID = 2^16-1 and the next PGK has already been used in some other PDCP entity), the ME shall use the next PGK to generate keys for this PDCP entity and set the PTK ID and counter associated with this PDCP entity to one.

In all case of counter wrap, new PTK shall be derived from the PGK taking the new PTK Identity into use. A new PEK shall be derived from the new PTK as well. The old PTK associated with this PDCP entity shall be deleted together with the corresponding old PEK derived from the old PTK key.

When closing a PDCP entity, if the PGK being used by that PDCP context is the most recently used one, the ME shall update the PTK ID and counter values stored in the volatile memory of the ME as follows:

- If the PTK ID in the PDCP entity is greater than the stored one, the ME shall update the PTK ID and counter stored in volatile memory of the ME to be the values from the PDCP entity;
- If the PTK ID in the PDCP entity is equal to the stored one and the counter values in the PDCP entity is greater than the stored one, the ME shall update the counter in the volatile memory of the ME to the value from the PDCP entity;
- Otherwise, no changes are made to the values stored in the volatile memory of the ME.

At power down, the UE first closes all its PDCP entities. Then for USIM storage of the PTK ID, the ME shall set the PTK ID and counter values in the USIM equal to those held in the volatile memory of the ME (i.e. the values that would be used to protect the next packet). Otherwise the ME shall set the PTK ID and counter values in the non-volatile memory equal to those held in the volatile memory of the ME.

If the receiving UE receives a PDCP packet on a PDCP entity with a new PTK Identity that has not been previously used with the same PGK and PGK identity in the receiving UE, then the receiving UE shall delete any old PTK for this PDCP entity and also delete the corresponding old PEK derived from the old PTK.

6.2.3.3 Security flows

6.2.3.3.1 Overview

The protection of one-to-many communication proceeds as shown in the figure below.

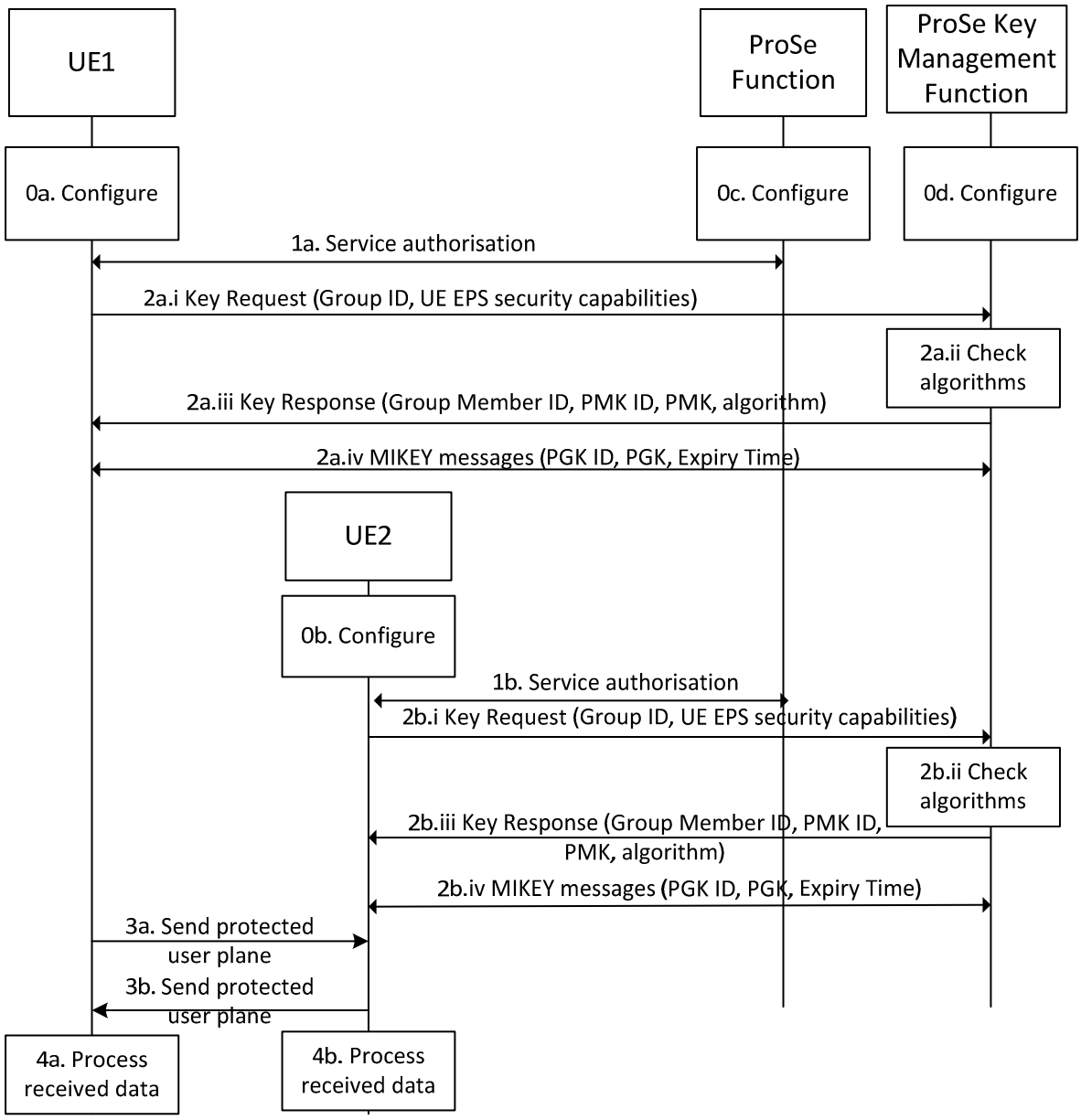

Figure 6.2.3.3-1: One-to-many security flows

0a or 0b: If needed the UE could be configured with any private keys, associated certificates or root certificate that they may need for contacting the ProSe Key Management Function to allow the keys to be kept secret from the operator. If none are provided, then the USIM credentials are used to protect that interface. The UE may also be pre-configured with the address of the ProSe Key Management Function.

NOTE 1: The ProSe Key Management Function is shown as a separate logical entity to allow the network operator to provision the radio level parameters and a 3rd party, e.g. public safety service, to have control over provisioning the keys. If such a separation is not needed then the ProSe Key Management Function may be deployed as part of the ProSe Function

0c and 0d: The ProSe Function and ProSe Key Management Function need to be configured with which subscriptions (either mobile subscriptions or identities in certificates) are member of which groups.The ProSe Key Management Function needs to pre-select an encryption algorithm for each group based on a local policy.

1a or 1b: The UE fetches the one-to-many communication parameters from the ProSe Function. As part of this procedure the UE gets its Group Identity (see TS 23.303 [2]) and is informed whether bearer layer security is needed for this group. In addition the UE may be provided with the address of the ProSe Key Management Function that it uses for obtaining keys for this group.

2a.i or 2b.i: The UE sends the Key Request message to the ProSe Key Management Function including the Group Identity of the group for which it wants to fetch keys and UE EPS security capabilities (including the set of EPS encryption algorithms the UE supports).

2a.ii or 2b.ii: The ProSe Key Management Function checks whether the group encryption algorithm is supported by the UE according to the UE EPS security capabilities, i.e. whether the group encryption algorithm is included by the set of EPS encryption algorithms the UE supports.

2a.iii or 2b.iii: The ProSe Key Management Function responds with the Key Response message. If the check of step 2a.ii or 2b.ii is successful for a particular group, this message contains the Group Member Identity and the EPS encryption algorithm identifier that the UE should use when sending or receiving protected data for this group. Otherwise, this message contains an indicator of algorithm support failure as the UE does not support the required algorithm. This message may also contain a PMK and associated PMK ID if the ProSe Key Management Function decides to use a new PMK.

2a.iv or 2b.iv: The ProSe Key Management Function sends the relevant PGKs, PGK IDs and expiry time to the UE using MIKEY.

3a or 3b: The UE calculates the PTK and PEK to protect the traffic it sends to the group. It does this by selecting the PGK as described in subclause 6.2.3.1 and uses the next unused combination of PTK Identity and Counter. It then protects the data using the algorithm given in step 2x.ii.

4a or 4b: A receiving UE gets the LC ID, Group Identity and Group Member Identity from the layer 2 header. It then uses the received bits of the PGK Identity to identify which PGK was used by the sender. The UE first checks that the PGK is valid (see subclause 6.2.3.1) and if so calculates the PTK and PEK to process the received message.

6.2.3.3.2 Messages between UE and ProSe Key Management Function

6.2.3.3.2.1 General

There are two types of messages that flow between the UE and ProSe Key Management Function. Firstly, there are the Key Request and Response messages. The UE uses these messages to request keys for particular groups, while the ProSe Key Management Function uses these messages to provide the UE with its group member identifier(s) and the security algorithms to use with the various groups. Secondly, there are the MIKEY messages, which the ProSe Key Management Function uses to send the PGKs to the UE. These messages are detailed in the following subclauses.

6.2.3.3.2.2 Key Request and Key Response messages

The purpose of these messages is for the UE to inform the ProSe Key Management Function of the groups that the UE wishes to receive keys for and the groups for which the UE no longer wishes to receive keys.The UE knows which ProSe Key Management Function to contact for each group as it is either pre-provisioned or provided by the ProSe Function. This is the FQDN of the ProSe Key Management Function.

The UE shall not release the PDN connection used to receive MIKEY messages containing PGKs until the UE has informed the ProSe Key Management Function that it no longer requires PGKs. This is to ensure that the ProSe Key Management Function is aware of the correct UE IP address for the purpose of performing PGK deliveries as specified in clause 6.2.3.3.2.3.

If the UE detects that a PDN connection, which is used for receiving PGKs is released by the network, the UE should try to send a new Key Request to inform the ProSe Key Management Function of its new IP address. This is to ensure that the ProSe Key Management Function becomes aware of the new UE IP address for the purpose of performing PGK deliveries. Any new IP address should override any existing ones of the UE at the ProSe Key Management Function.

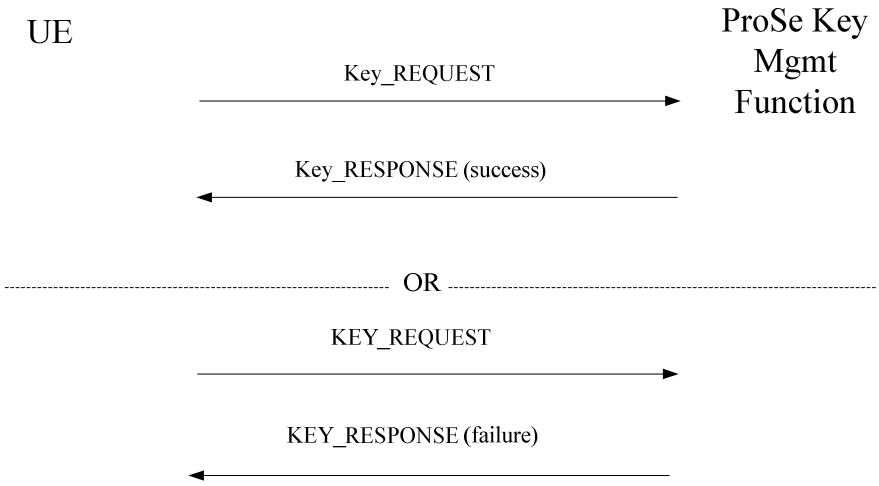

Figure 6.2.3.3.2.2-1

The protection for the Key Request and Key Response message is described in subclause 6.2.3.5.

When sending a Key Request message to request the ProSe Key Management Function to send PGKs or to change the groups for which it wants to receive keys, the UE shall include the following information;

- The indication of security algorithms that the UE supports for one-to-many commuications;
- List of Group Identities for which the UE would like to receive keys;
- For each Group Identity, the PGK IDs of any keys for that group. If the UE holds no keys for this group, then it sends an all zero PGK ID;
- List of Group Identities for which the UE would like to stop receiving keys.

The ProSe Key Management Function shall check that the UE is authorised to receive keys for the requested groups. This is done by using the UE identity that is bound to the keys that established the TLS tunnel in which the message is sent. It also checks that the UE supports the confidentiality algorithm required for each group. If the UE doesn"t then the ProSe Key Management Function responds with the appropriate error for that group. The ProSe Key Management Function shall update the stored set of the groups for which the UE will be sent keys.

The ProSe Key Management Function responds to the UE with a Key Response message that includes the following parameters:

- List of the Group Identities that were included in the Key Request message;
- For each group that keys will be supplied for, the security algorithm that should be used to protect the data and the group member identity; and
- For each of the other groups, a status code to indicate why keys will not be supplied for that group.
- An optional PMK and PMK Identity.

For the groups that the UE will get keys for, the UE shall store the received information associated with that group identity. If a PMK and PMK identity are included, the UE shall store these and delete any previously stored ones for this ProSe Key Management Function.

The ProSe Key Management Function shall initiate the PGK delivery procedures for the keys that are needed by the UE.

6.2.3.3.2.3 MIKEY messages

6.2.3.3.2.3.1 General

MIKEY is used to transport the PGKs from the ProSe Key Management Function to the UE. MIKEY shall be used with pre-shared keys as described in RFC 3830 [13]. The PMK shared by the ProSe Key Management Function and the UE shall be used as the pre-shared secret. The UDP port for MIKEY is 2269.

The ProSe Key Management Function may use the initial message to send the PGK to the UE. Alternatively the ProSe Key Management Function may use the initial MIKEY message (by setting the PGK_ ID to all zeros) to trigger a Key Request message from the UE. As part of this Key Request Message, the ProSe Key Management Function may refresh the PMK used between the UE and ProSe Key Management Function.

The response message is optional and only used if explicitly requested in an initial message containing a PGK.

The replay protection of the MIKEY messages is provided by a 32-bit counter that is associated with each PMK. It is initially set to zero and is increased by one for every message sent. A received message is discarded if the counter value is not greater than the largest successfully received message.

6.2.3.3.2.3.2 Creation of the MIKEY key delivery message

The initial MIKEY message is formed of the payloads which are filled as follows:

- **MIKEY common header**: The CSB ID field of the MIKEY common header id shall be set to a random number. If the ProSe Key Management Function requires an acknowledgement of the PGK delivery message, then, it sets the V-Bit to 1. The #CS field shall be set to zero. The CS ID map type subfield shall be set to "Empty map" as defined in RFC 4563 [35].

- **Timestamp payload**: The timestamp field shall be of type 2 and contain the value of the replay counter that is associated with this message.

- **MIKEY-RAND field**: The MIKEY-RAND field shall contain a 128-bit random number that is chosen by the ProSe Key Management Function.

- **IDi payload**: The ID Type shall be set to the value 0 to indicate an NAI with the identity formed from the Group Identity || PGK ID @ FQDN of the ProSe Key Management Function.

- **IDr payload**: The ID Type shall be set to the value 0 to indicate an NAI with the identity formed of the PMK identity of the PMK used to protect the MIKEY message @ the FQDN of the ProSe Key Management Function.

- **KEMAC Payload**: The Type Subfield shall be set to value 2. The use of the NULL algorithm in MAC field is not allowed. The use of the NULL alg in the Encr field is not allowed. The KV (Key Validity) subfield shall be set to the value 2. The lower and upper limits of the validity period of the PGK are expressed in unit of seconds and coded in binary format as the 40 least significant bits of the Coordinated Universal Time as defined in 3GPP TS 36.331 [40]. . In order to get the UE to delete a PGK, the ProSe Management Function shall set the lower and upper limits to be the same.

6.2.3.3.2.3.3 Processing the MIKEY key delivery message

When a MIKEY message arrives at the ME, the processing process as follows:

- The UE shall extract PMK @ FQDN of the ProSe Key Management Function for the IDr payload to establish which PMK was used to protect the MIKEY message.

- The Timestamp Payload is checked and compared against the stored Counter for the PMK, and the message is rejected if that counter is not larger then the current Counter.

- The integrity of the message is checked. If this fails, the message is discarded, otherwise the processing continues as follows.

- The counter associated with the PMK shall be set to the values conveyed in the Timestamp payload.

- If the PMK used to protect the message is the not the last successfully used one, then it becomes the last successfully used one and any other PMK related to this ProSe Key Management Function is deleted.

- The UE shall extract the Group ID and PGK ID from the received IDi Payload. If PGK ID is all zeros, then the UE shall stop processing the message and send a Key Request message (as described in subclause 6.3.2.2.3.3).

- The UE shall extract the PGK and Key Validity data from the KEMAC payload as described in section 5 of RFC 3830 [5]. If the lower and upper limits are the same, then the UE shall delete any key with the Group ID and PGK ID contained in the MIKEY messages. Otherwise the UE shall store the PGK along PGK ID and the validity limits.

6.2.3.3.2.3.4 MIKEY Verification message

If the ProSe Key Management Function is expecting a response (i.e. set the V-bit in the MIKEY common header to 1), then the UE shall respond with a verification message. The verification message is made up of the payloads which are filled as described:

- **MIKEY common header**: The CS ID is the same as in the MIKEY message carrying the PGK. The #CS field shall be set to zero. The CS ID map type subfield shall be set to "Empty map" as defined in RFC 4563 [35].

- **Timestamp payload**: The timestamp field shall be of type 2 and contain the value of the replay counter that is associated with this message.

- **IDr payload**: The ID Type shall be set to the value 0 to indicate an NAI with the identity formed of the PMK identity of the PMK used to protect the MIKEY message @ the FQDN of the ProSe Key Management Function.

- **Verification payload**: The use of the NULL algorithm in the MAC field is not allowed. The MAC included in the verification payload shall be calculated over both the initiator"s and the responder"s identities as well as the timestamp in addition to over the response message as defined in RFC 3830 [13].

6.2.3.4 Protection of traffic between UE and ProSe Function

In order to protect the messages between the UE and ProSe Function, the UE shall support the procedures for the UE given in subclause 5.3.3.2 and the ProSe Function shall support the procedures for the network function given in subclause 5.3.3.

6.2.3.5 Protection of traffic between UE and ProSe Key Management Function

In order to protect the UE-initiated messages between the UE and ProSe Key Management Function, the UE shall support the procedures for the UE given in subclause 5.3.3.2 and the ProSe Key Management Function shall support the procedures for the network function given in subclause 5.3.3.2.

The MIKEY messages are protected as described in subclause 6.2.3.3.2.3.

6.2.3.6 Protection of traffic between UEs

6.2.3.6.1 Protection of data

ProSe enabled Public Safety UEs shall implement EEA0, 128-EEA1 and 128-EEA2 and may implement 128-EEA3 for ciphering one-to-many traffic.

The LTE ciphering algorithms (see TS 33.401 [21]) are used with the following modifications;

- Direction is always set to 0;

- Bearer[0] to Bearer[4] are set to LCID;

- COUNT[0] to COUNT[15] are set to PTK ID;

- Counter is input into COUNT[16] to COUNT[31].

6.2.3.6.2 Key derivation data in PDCP header

In terms of signalling between the UEs to transfer the relevant security information, e.g. to indicate the correct PTK to use to calculate PEK, the header of the PDCP packet for user plane data shall contain the 5 least significant bits of the PGK Identity, PTK Identity and Counter. This is illustrated in figure 6.2.3.6.2-1.

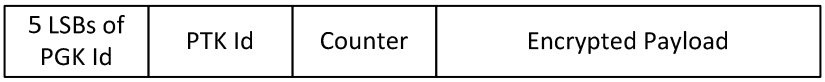

Figure 6.2.3.6.2-1: Security aspects of the PDCP packet for user plane data

NOTE: The Group Identity and Group Member Identity are carried in layers below the PDCP layer.

If the network configuration is not to use confidentiality protection, then the tramsmitting UE shall set the values of the security information (PGK Identity, PTK Identity and Counter) to zero in the header of the PDCP packet.

6.2.4 Solution description for media security of one-to-many communications

6.2.4.1 Media stream protection

The following mechanism shall be used to protect one-to-many communications which use the Real-Time Transport Protocol (RTP) or the RTP Control Protocol (RTCP), cf. RFC 3550 [10].

The integrity and confidentiality protection for one-to-many communications using RTP shall be achieved by using the Secure Real-Time Transport Protocol (SRTP), RFC 3711 [11]. The integrity and confidentiality protection for one-to-many communications using RTCP shall be achieved by using the Secure RTCP protocol (SRTCP), RFC 3711 [11]. The security profile for these types of media stream is provided in annex C.

The key management mechanism for SRTP and SRTCP is described in the rest of clause 6.2.4. As a result of this mechanism, the group members share a GSK as part of the session setup procedure. The GSK shall be used to derive the SRTP master key. The session may only last for a single transmission, or may be maintained for a period to allow many members of the group to efficiently communicate. If late-entry to the media session is required, the transmitted SDP offer, described in clause 6.2.4.5, may be periodically resent (e.g. every 5 seconds within a SIP REFER).

6.2.4.2 Overview of key management solution

The architecture for providing media security for one-to-many communications within groups is shown in figure 6.2.4.2-1.

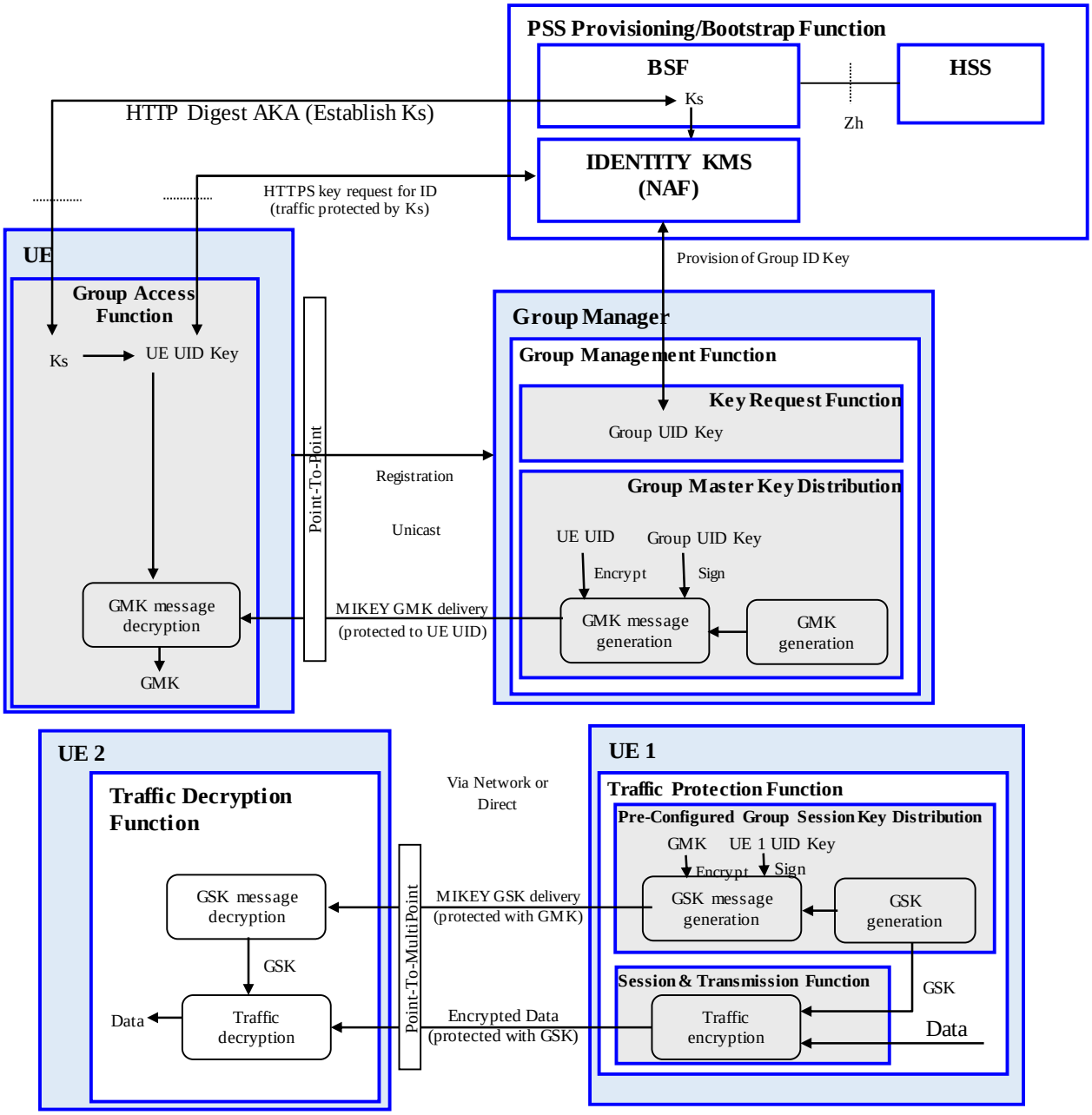

Figure 6.2.4.2-1: Security architecture for media security of one-to-many group communications

Figure 6.2.4.2-1 describes the security architecture to distribute keying material for one-to-many group communications. The architecture provides a media-plane solution to distribute group keys and secure group communications.

At the top of the diagram, a Public Safety UE is provisioned by a KMS with key material associated with its identity. If required, GBA is used to bootstrap the security of the connection between the UE and the KMS.
The KMS also provisions the Group Manager with keying material for the identity of groups which it manages. This process is described in clause 6.2.4.3.

The Group Manager distributes Group Master Keys (GMKs) to UEs within the group. The GMK is encrypted to the identity associated with the receiving UE and signed with the identity of the group. This process is described in clause 6.2.4.4.

Once a Group Master Key has been distributed within the group, UEs are able to setup group communications.
The initiating UE generates, encrypts and transmits a Group Session Key (GSK) to group members.
This transmission is encrypted using the GMK and may be authenticated, allowing the origin of the transmission to be verified. The distribution process may also be performed over a direct transmission. This process is described in clause 6.2.4.5.

The Group Session Key is used as the SRTP master key to provide media security of the one-to-many communication, as described in clause 6.2.4.1.

6.2.4.3 UE provisioning by KMS

6.2.4.3.1 General

Prior to group security configuration, each Public Safety UE shall be provisioned with keys corresponding to identity associated with the Public Safety UE, along with domain specific information such as KMS's public certificate.

To complete this procedure, the UE shall request to be provisioned by the KMS, and as part of this process, the KMS shall securely verify the identity of the UE. The KMS shall respond with domain information and key material appropriate to the request and according to local policies.

6.2.4.3.2 Pre-provisioning

Prior to provisioning, the Public Safety UE shall be provided with the address (e.g. URL) of the KMS.

Public-safety UEs may also be pre-provisioned with additional security parameters where required. For some user groups the UE may be entirely pre-provisioned prior to deployment.

6.2.4.3.3 UE to KMS connection security

To be provisioned, the Public Safety UE shall establish a secure connection to the KMS.

In order to protect the messages between the UE and KMS, the UE shall support the procedures for the UE given in subclause 5.3.3.2 and the KMS shall support the procedures for the network function given in subclause 5.3.3.2.

6.2.4.3.4 Provisioning request

This procedure registers a Public Safety UE within a specific domain. The Public Safety UE shall send a provisioning request to the KMS. Upon successful request, the KMS shall return indication of success and provisioning material.

It is assumed that this request is made and responded to over an established secure connection as described in clause 6.2.4.3.3. Figure 6.2.4.3.4-1 describes the procedure.

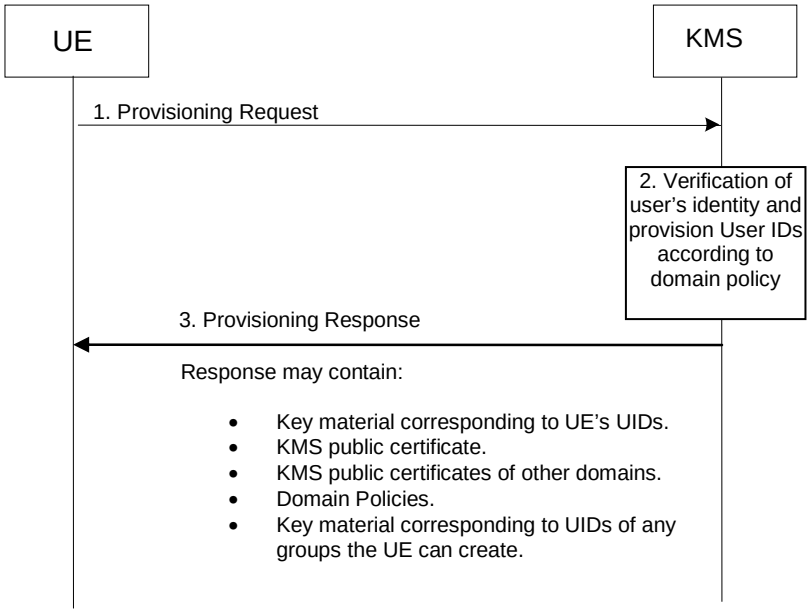

Figure 6.2.4.3.4-1: KMS provisioning process

The provisioning shown in Figure 6.2.4.3.4-1 is as follows:

1) The Public Safety UE shall send a provisioning request to the KMS.

2) The KMS shall check that the request is valid and shall ensure the Public Safety UE's identity is verified and apply the domain policy on assigning user identities to the UE.

3) After successful processing, the KMS returns a provisioning response to the Public Safety UE. The KMS populates the response with all information required to provision the UE. The contents of response may be signed by the KMS. The response includes key material corresponding to the user identity associated with UE. It may also include information relating to KMS domains, cross-domain communications and groups.

Details of provisioning requests may be found in Annex B.

6.2.4.4 Provisioning of the group master key

6.2.4.4.1 General

To create the group's security association, a Group Master Key (GMK) is distributed to Public Safety UEs in a group notification message. The Group Manager generates and distributes the group notification message via a Local Group Server and a Serving Signalling Server.

The role of the Group Manager is to manage the security context for a (set of) specific group(s). The Local Group Server manages group communications with local group Public Safety UEs. The Serving Signalling server manages the routing of messages to Public Safety UEs. For one-to-many group communications, the Serving Signalling Server, Local Group Server and Group Manager should be viewed as functions provided by the ProSe Function.

Security associations are setup using group notification messages generated by the Group Manager. Group notification messages notify Public Safety UEs of the existence of the group and also provide the GMK encrypted for the Public Safety UE.

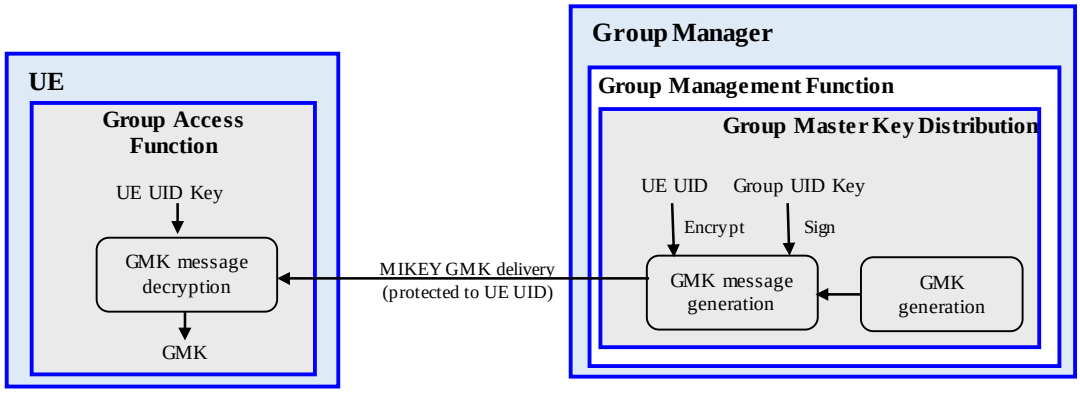

Figure 6.2.4.4.1-1: Overview of Group security configuration

The security processes are summarized in Figure 6.2.4.4.1-1. The GMK is encrypted to the user identity associated to the Public Safety UE and signed using the (key associated with the) identity of the group, an identity which the group manager is authorized to use by the KMS. Key distribution is confidentiality protected, authenticated and integrity protected via this mechanism.

It is expected that there will be one GMK per group UID. In the event of compromise of the GMK, the group owner will use a new Group UID with a new GMK.

6.2.4.4.2 Security procedures for GMK provisioning

This procedure distributes a Group Master Key (GMK) from the group manager to the Public Safety UEs within the group.

Figure 6.2.4.4.2-1 shows the security procedures for creating a security association for a group.

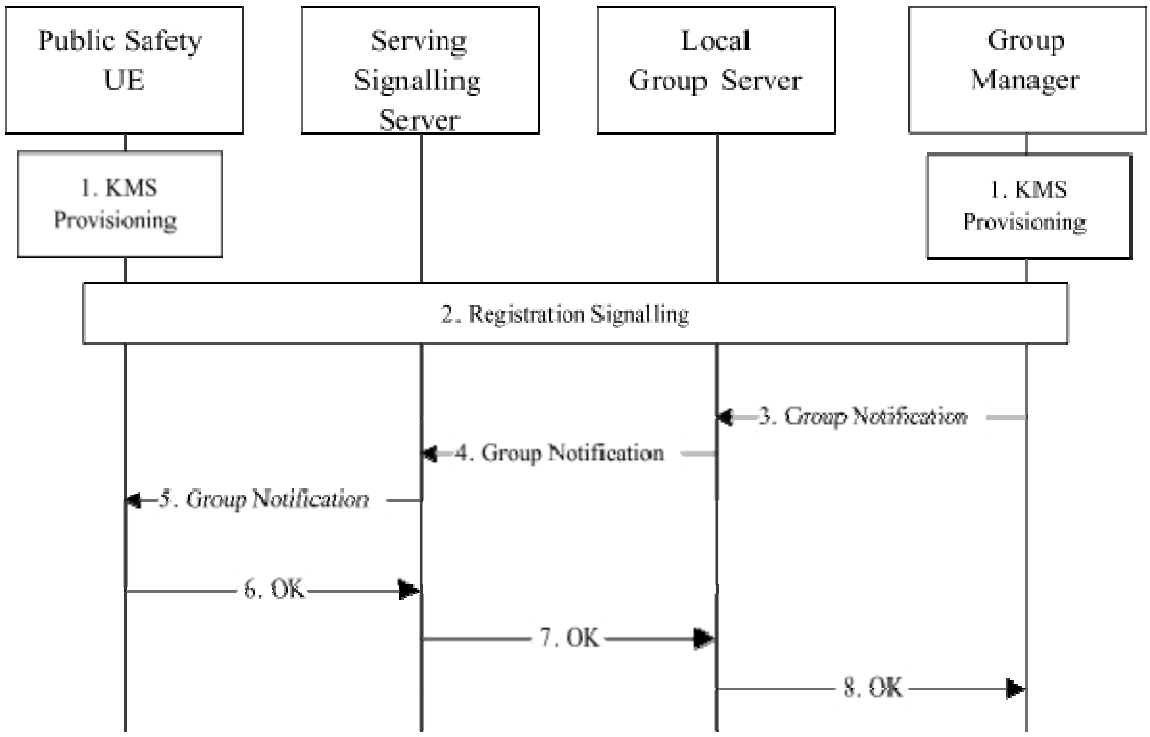

Figure 6.2.4.4.2-1: Security configuration for Groups

A description of the procedures depicted in Figure 6.2.4.4.2-1 follows:

1) Prior to beginning this procedure the Public Safety UEs shall be provisioned by a KMS as described in clause 6.2.4.3. The Group manager shall also be securely provisioned by the KMS to use the identities of the group(s) that it manages.

2) The Public Safety UE shall register with a serving signalling server to use communication services. As part of this mechanism the Public Safety UE shall register with a group server and group manager.

NOTE 1: As a consequence of registration, the group server obtains address information for Public Safety UEs.

3) The group manager shall send a notification message to Public Safety UEs within the group.
The message shall be routed to the local group manager. The notification message shall contain a MIKEY-SAKKE I_MESSAGE as specified in RFC 6509 [12] and shall encapsulate a GMK for the group.
The I_MESSAGE shall be encrypted to the user identity associated to the Public Safety UE and shall be signed using the (key associated with the) group identity. The message should also provide the GMK key id and period of use for the GMK.

NOTE 2: Only a Group Manager authorised by a trusted KMS knows the private key associated with the group identity and is able to sign a message. All group members are able to verify the signature and hence can confirm the group manager is authorised to manage this group.

NOTE 3: If there is a requirement to multicast the group notification message, the group manager should send the group notification message to a URI-List, and should include a different I_MESSAGE for each UE.

4) The local group server shall forward the group notification message to the user's serving signalling server.

5) The serving signalling server shall forward the message to the Public Safety UE.
The UE shall verify the group identity and signature of the I_MESSAGE are acceptable. If so, the UE shall extract the GMK from the I_MESSAGE. The UE shall use the last received GMK as the current group key based on the timestamp in the I_MESSAGE.

6) Upon successful receipt and processing, the Public Safety UE shall confirm receipt of the group notification message to its serving signalling server. The confirmation contains no security information.

7) The serving signalling server shall forward the confirmation to the local group server.

8) The local group server shall forward the confirmation to the Group manager.

If rekeying or revocation of the GMK is required, the Group manager may repeat the above procedure using a new Group identity.

Details of the format of the MIKEY message is defined in annex D.

6.2.4.5 GSK distribution for group media security

6.2.4.5.1 General

Group communications within a session are protected using a Group Session Key (GSK). Hence, prior to beginning a communication, a session key is generated by the initiating UE and shared with the group members. For groups, the Group Master Key (GMK) is used to protect the distribution of the GSK. The GSK may also be signed using the identity associated with initiator's UE.

This process is summarized in Figure 6.2.4.5.1-1.

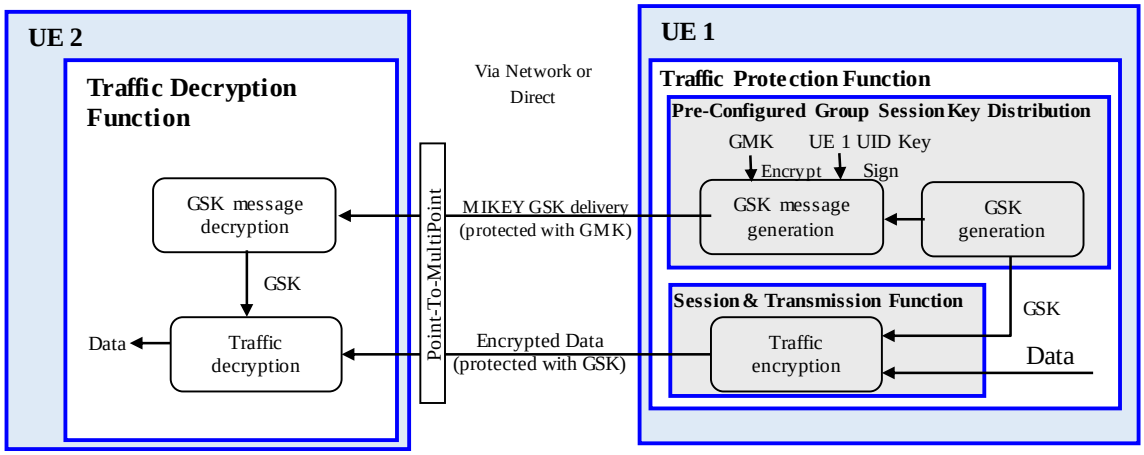

Figure 6.2.4.5.1-1: Overview of Session Key Distribution for Groups

The MIKEY GSK delivery message may include a MKI associated with the GSK to allow identification of GSK-encrypted traffic.

6.2.4.5.2 Security procedures for GSK distribution (network independent)

The SDP Offer is broadcast by the initiating UE and no response is expected. The SDP Offer contains the Group Session Key (GSK) protected by the Group Master Key (GMK). Following the broadcast of the SDP Offer, the media is broadcast over the one-to-many direct link, protected under the GSK.

Figure 6.2.4.5.2-1 describes the procedure. As there are no responses or acknowledgements, this procedure shall proceed on a "best-endeavour' basis.

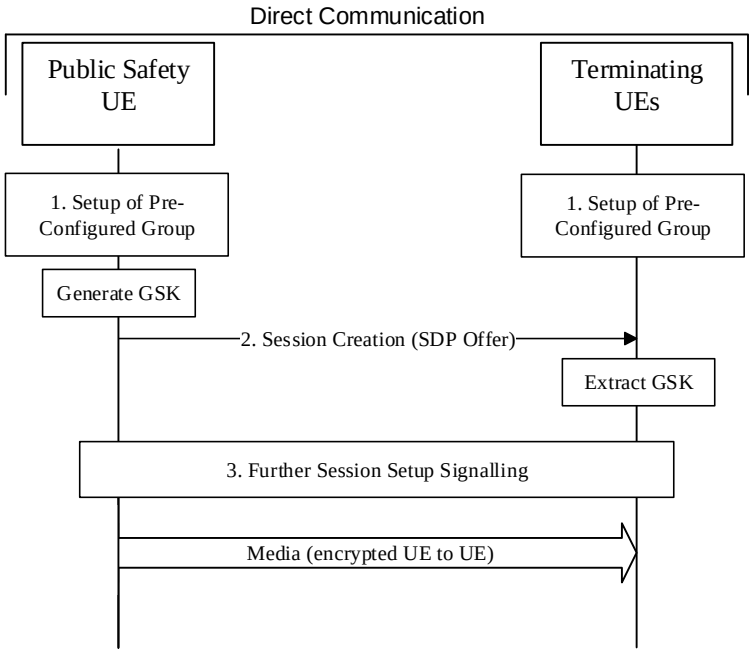

Figure 6.2.4.5.2-1: Security procedures for Session Key Distribution for Groups (network independent)

The procedure in Figure 6.2.4.5.2-1 is described step-by-step below:

1) Prior to beginning this procedure the security association for the group shall be created, as described in clause 6.2.4.4.

2) The initiating Public Safety UE shall generate a GSK and shall broadcast a 'media session creation message' containing an SDP Offer. Within the SDP Offer, the initiating UE shall include a MIKEY Pre-Shared-Key message, as specified in RFC 3830 [13]. The MIKEY PSK message shall encapsulate the session key with the Group Master Key (GMK) and shall be denoted by a key identifier.
The MIKEY message may be signed using the (key associated with the) identity of the initiating UE by attaching a ECCSI SIGN payload, as defined in RFC 6509 [12] and RFC 6507 [14].

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

3) The receiving UE(s) may check the signature on the MIKEY PSK message if it exists. If checked, the message shall be rejected if the signature is incorrect or the signing identity unacceptable. The receiving UE shall use the key identifier to find the GMK used by the initiating UE and use this to extract the Group Session Key. Further messages may be sent by the initiating UE to setup the one-to-many broadcast media session, but these contain no security information.

As a result of this procedure, the initiating UE has shared a Group Session Key (GSK) for protecting the media stream. The message format of the MIKEY PSK message is defined in annex D.

6.2.4.5.3 Security procedures for GSK distribution (network connected)

NOTE 1: Clause 6.2.4.5.3 is intended to show an example of how the SDP offer could be sent via network while the encrypted data is sent directly between UEs. The clause is not proposing an architecture for standardization.

From a security perspective, this procedure follows exactly the same security mechanism as for network connected group communications described in clause 6.2.4.5.2, though the routing differs. In this case, these procedures assume the existence of a Group Controller and signalling is routed via the network. The Group Controller is aware of which UEs are members of the group and is also responsible for routing signalling and media traffic to the members of the group. The Group Controller need not be a member of the group or have access to the media that it routes. If this entity requires access to the group communication to fulfil its function, it should be treated as a member of the group and be provisioned with the Group Master Key by the Group manager.

For one-to-many group communications, the Group Controller, Serving Signalling Server and Local Group Server should be viewed as a functions provided by the ProSe Function.

Figure 6.2.4.5.3-1 demonstrates the signalling flow for session key distribution.

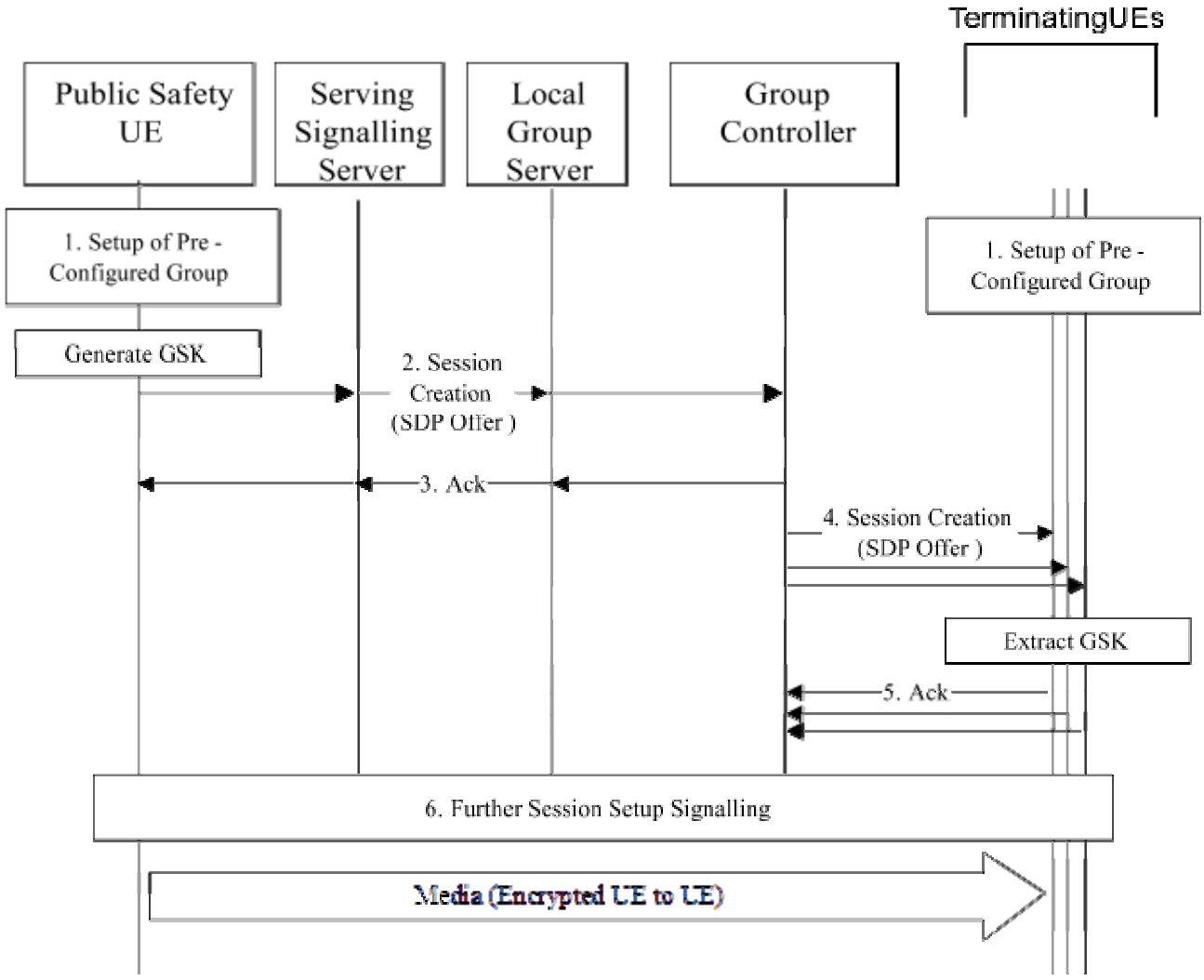

Figure 6.2.4.5.3-1: Security procedures for GSK distribution for Groups (network connected)

The procedure in Figure 6.2.4.5.3-1 is described step-by-step below:

1) Prior to beginning this procedure a security association for a group shall be created, as described in clause 6.2.4.4.

2) The initiating UE shall generate a GSK and send a session creation notification (e.g. SIP INVITE) to the group controller, via the serving signalling server and local group server. Within the SDP Offer of this message, the initiating UE shall include a MIKEY Pre-Shared-Key message, as specified in RFC 3830 [13]. The MIKEY-PSK message shall encapsulate the session key with the Group Master Key (GMK) which shall be denoted by a key identifier. The MIKEY message may be signed using the (key associated with the) identity of the initiating UE by attaching a ECCSI SIGN payload, as defined in RFC 6509 [12] and RFC 6507 [14].

NOTE 2: This message may be pre-generated to increase the efficiency of the communication.

3) The group controller shall notify the initiating UE that it received the message (e.g. SIP TRY). This message does not contain any security information.

4) The group controller shall send a session creation notification (e.g. SIP INVITE) to each Public Safety UE within the group, via each UE's local group server and signalling server. The SDP Offer of the message provided by the initiating UE shall be duplicated within each message, including the MIKEY-PSK content. The message shall be routed via each Public Safety UE's local group server and serving signalling server.

5) The receiving UE(s) may check the signature on the MIKEY PSK message if it exists. If checked, the message shall be rejected if the signature is incorrect or the signing identity unacceptable. The receiving UE shall use the key identifier to find the GMK used by the initiating UE and use this to extract the Group Session Key. The receiving UE(s) shall notify the group controller that the session creation message was received. This message does not contain any security information.

6) Further messages may be sent to setup the group session. These messages contain no security information.

As a result of this procedure, the initiating UE has shared a Group Session Key (GSK) for protecting the media stream.

The message format of the MIKEY PSK message is defined in annex D.

6.3 EPC-level discovery of ProSe-enabled UEs

6.3.1 Security for proximity request authentication and authorization

6.3.1.1 General

The ProSe Function of two UEs may belong to different PLMNs and, according to the current version of TS 23.303 [2], the application server is the entity which discloses the EPUID of a particular target UE to a ProSe Function so that it can request to the Prose Function managing that target UE the sending of its location information for a particular period of time.

The risk is that it would take just one Proximity Request from the UE for the ProSe Function to actually keep a record of [ALUID_B, EPUID_B, PFID_B] mapping and have the capability to later send a proximity request (step 4 of Proximity Request procedure in clause 5.5.5 of TS 23.303 [2]) although the UE didn't actually send a proximity request at all (step 4 of Proximity Request procedure in clause 5.5.5 of TS 23.303 [2]). This could lead to massive surveillance of users by other PLMNs that may be very difficult to detect by the PLMN which serves particular ProSe UEs.

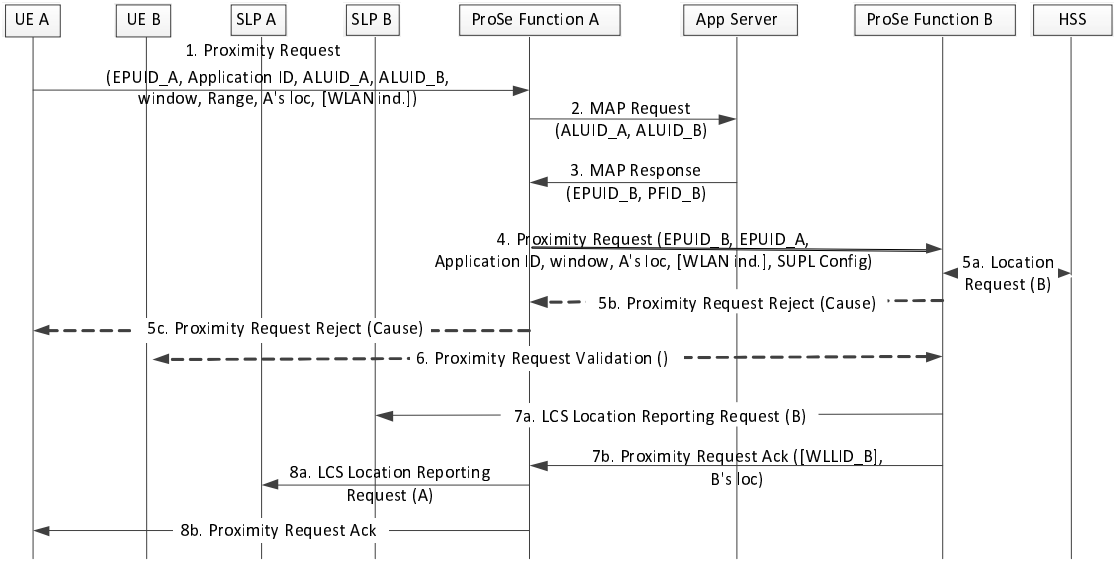

Figure 6.3.1.1-1: extract from 3GPP TS 23.303 [2], Proximity request procedure (clause 5.5.5)

6.3.1.2 Application Server-signed proximity request

UE A doesn't sign the proximity request sent to its ProSe Function A, but trusts the Application Server to control the authorization of the proximity request sent on its behalf.
The authorization criteria can be based on detection mechanisms of very high volume of incoming proximity requests from a ProSe Function that doesn't match with the frequency usage of the ProSe Application by the users, or it can be based on a presence detection mechanism over the PC1 interface.
ProSe Function A requests an authorization to the Application Server for each proximity request it shall transmit over the PC2 interface. The Application Server returns parameters which specify which operations are authorized (e.g. authorized to send only one request, authorized to send X requests until particular date, etc…).
ProSe Function B is assured of the authenticity of the proximity request received from ProSe Function A by verifying the signature with a verification key from the Application Server.
The token verification key is fetched over the PC2 interface between the ProSe Function B and the Application Server.

The procedure below further defines the Proximity Request procedure in clause 5.5.5 of TS 23.303 [2] to support authenticity of the request.

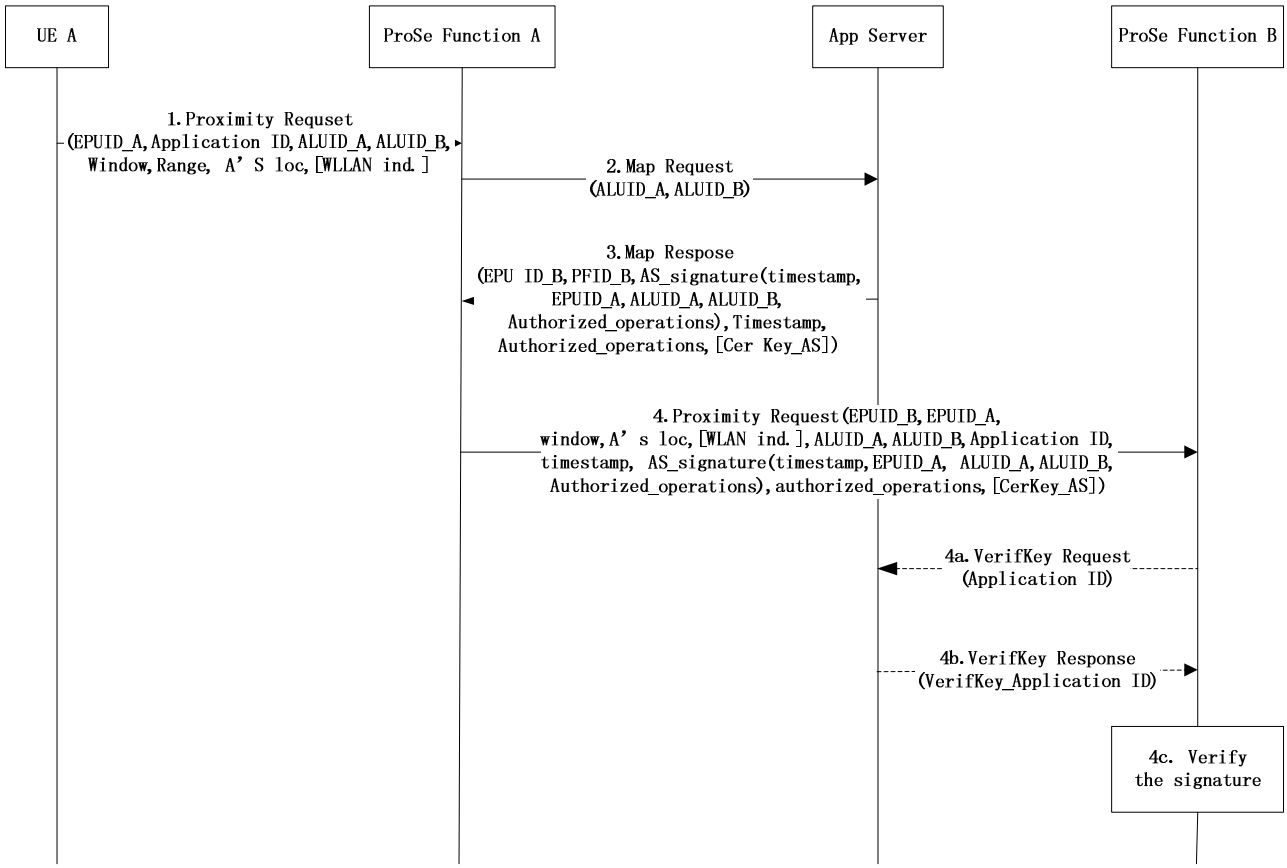

Figure 6.3.1.2-1: Application Server-signed Proximity Request

1. Same as Step 1 of procedure in clause 5.5.5 of TS 23.303 [2]

2. Same as Step 2 of procedure in clause 5.5.5 of TS 23.303 [2]

3. The Application Server returns as part of the Map Response the following additional data: the authorized operations (e.g. authorized to send only one request, authorized to send X requests until particular date, etc…), a timestamp, the signature AS_signature of the cryptographic hash of the concatenation of the EPUID_A, the ALUID_A, the ALUID_B, the authorized operations and the timestamp value, and optionally the associated certificate CertKey_AS of Application Server's verification key.

4. ProSe Function A sends as part of the Proximity Request to ProSe Function B the following additional data: the AS_signature, ALUID_A, ALUID_B, the timestamp, the authorized operations, the Application ID and optionally the CertKey_AS's certificate.

4a . If the CertKey_AS's certificate wasn't part of the Proximity request, or that either the CertKey_AS's certificate or verification key wasn't stored in internal memory, then ProSe Function B sends a Verification Key fetching requests to Application Server's verification key (identifiable with the Application ID).

4b . The Application Server returns the verification key.

4c . If the verification of signature from the Application Server is successful then the procedure continues the procedure from step 5 in clause 5.5.5 of TS 23.303 [2].

6.3.1.3 Proximity request digital signature algorithms and key strength

The cryptographic length of the signing asymmetric keys shall have at least a key strength equivalent to a 128-bits symmetric key. The following digital signature algorithms may be used:

RSA as specified in FIPS 186-4 [15]: the minimum key length shall be 3072 bits. The minimal size for the hash function shall at least be SHA-256 which shall be used as specified by NIST [20].

DSA as specified in FIPS 186-4 [15]: the minimum key length shall be 256 bits.

ECDSA as specified in BSI TR-03111 [16]: the minimum key length shall be 256 bits, the elliptic curve domain parameters shall be selected among those available in RFC 5639 [17]. The corresponding hash function shall be chosen depending on the previously selected elliptic curve domain parameters (cf. clause 5 in RFC 5639 [17]).

6.3.1.4 Proximity request hash input format

The input to the hash function shall be encoded as specified in Annex B.1 of TS 33.220 [5] and shall consist of the concatenation of proximity request parameters and their respective lengths:

FC = 0x00

P0 = EPUID_A

L0 = Length of P0 value

P1 = ALUID_A

L1 = Length of P1 value

P2 = ALUID_B

L2 = Length of P2 value

P3 = Timestamp. It shall use the date-time format as defined in clause 5.6 of RFC 3339 [18] and shall be encoded according to Annex B.2.1.2 of TS 33.220 [5]

L3 = Length of P3 value

P4 = Authorized operations

L4 = Length of P4 value

NOTE: The key derivation function defined in Annex B.1 of TS 33.220 [5] is not used, therefore the FC value should only be considered as a dummy value.

6.3.1.5 Verification key format

The following shall be supported by the Application Server and the ProSe Function:

The verification key shall be formatted like the "Subject Public Key Info" element of a X.509 certificate.
The "Subject Public Key Info" is specified in clause 4.1.2.7 of RFC 5280 [19].
RFC 5639 [17] shall also be supported in order to include ECDSA with Brainpool elliptic curve domain parameters.

6.3.1.6 Profile for Application Server certificate

The following may be supported by the Application Server and the ProSe Function:

Certificates used for authentication of the Proximity Request shall meet the certificate profiles given in TS 33.310 [4] as follows:

- clause 6.1.3, for SEG certificates, shall apply to ALUID-associated certificates, and
- clause 6.1.4, for SEG CA certificates, shall apply to any CA certificates used in a chain to validate the certificates, with the following additions and exceptions:
  - Mandatory critical key usage: only digitalSignature shall be set.

6.3.2 Protection of traffic between UE and ProSe Function

In order to protect the messages between the UE and ProSe Function, the UE shall support the procedures for the UE given in subclause 5.3.3.2 and the ProSe Function shall support the procedures for the network function given in subclause 5.3.3.2.

6.4 Security for EPC support WLAN direct discovery and communication

For EPC support for WLAN Direct Discovery, the security mechanisms as defined for EPC-level ProSe Discovery (clause 6.3) apply.

For EPC support for WLAN Direct communication, based on the UEs WLAN security capabilities the ProSe Function may provide only such WLAN security parameters to the ProSe-enabled UEs that ensure confidentiality and integrity of the ProSe-assisted WLAN direct communication path to a level comparable with that provided by the existing 3GPP system.

NOTE: Release 12 does not provide stage 3 specification.

Annex A (normative): Key derivation functions

A.1 KDF interface and input parameter construction

A.1.1 General

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

A.1.2 FC value allocations

The FC number space used is controlled by TS 33.220 [5].
FC values allocated for this specification are in range of 0x49 – 0x4F.

A.2 Calculation of the MIC value

When calculating a MIC using the Discovery Key, the following parameters shall be used to form the input S to the KDF that is specified in Annex B of TS 33.220 [5]:

- FC = 0x49.
- P0 = Message Type (see TS 24.334).
- L0 = length of above (i.e. 0x00 0x01).
- P1 = ProSe Application Code.
- L1 = length of above (i.e. 0x00 0x17).
- P2 = UTC-based counter associated with the discovery slot.
- L2 = length of above (i.e. 0x00 0x04).

The MIC is set to the 32 least significant bits of the output of the KDF.

The Discovery Key, Time parameter and discovery message follow the encoding also specified in Annex B of TS 33.220 [5].

A.3 Calculation of PTK

When calculating a PTK from PGK, the following parameters shall be used to form the input S to the KDF that is specified in Annex B of TS 33.220 [5]:

- FC = 0x4A.
- P0 = Group Member Identity (i.e. the Layer 2 source address of the sending UE).
- L0 = length of Group Member Identity (i.e. 0x00 0x03).
- P1 = PTK Identity.
- L1 = length of PTK Identity (i.e. 0x00 0x02).

- P2 = Group Identity.
- L2 = length of Group Identity (i.e. 0x00 0x03).

The input key shall be the 256-bit PGK.

A.4 Calculation of keys from PTK

When calculating a PEK from PTK, the following parameters shall be used to form the input S to the KDF that is specified in Annex B of TS 33.220 [5]:

- FC = 0x4B
- P0 = 0x00
- L0 = length of P0 (i.e. 0x00 0x01)
- P1 = algorithm identity
- L1 = length of algorithm identity (i.e. 0x00 0x01)

NOTE: If more than one key is needed from PTK, then parameter P0 could be turned into an algorithm type distinguisher as in TS 33.401[21].

The algorithm identity shall be set as described in TS 33.401 [21].

The input key shall be the 256-bit PTK.

For an algorithm key of length n bits, where n is less or equal to 256, the n least significant bits of the 256 bits of the KDF output shall be used as the algorithm key.

Annex B (Normative): KMS provisioning messages to support media security

B.1 General aspects

This annex specifies the key management procedures between the KMS and the UE that allows the UE to be provisioned to use media security. It describes the requests and responses for the following provisioning messages:

- KMS Initialise
- KMS KeyProvision
- KMS CertCache

All KMS communications are made via HTTP(S). The UE 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 UE are secure and that the KMS has authenticated the identity of the UE.

B.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: "http://example.org/keymanagement/identity/v1/init" is the resource path to initialise a user within the domain "example.org".

To make a "KMS Initialise" request the UE shall make a HTTP POST request to the subdirectory "init" i.e. Request-URI takes the form of "/keymanagement/identity/v1/init".

To make a "KMS KeyProvision" request the UE shall make a HTTP POST request to the subdirectory "keyprov" i.e. Request-URI takes the form of "/keymanagement/identity/v1/keyprov".

Optionally, the Request-URI of the POST request may contain a specific user or group URI which the UE 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 RFC 5905 [28]. For example, if the user required keys specifically for 23rd Feb 2014 at 08:39:14.000 UTC, the request would be: "/keymanagement/identity/v1/keyprov/user%40example.org/D6B4323200000000".

To make a "KMS CertCache" request the UE 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 "/keymanagement/identity/v1/certcache/12345".

B.3 KMS responses

B.3.0 General

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

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

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

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

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

B.3.1 KMS Certificates

B.3.1.1 Description

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

It is assumed that the UE is managed by a single KMS, the UE"s Root KMS. This Root KMS is the only KMS which provisions the UE. The KMS certificate of the Root KMS is known as the Root KMS certificate. This certificate is required to encrypt to the UE, and verify signatures of UE (as well as others within the domain).

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

B.3.1.2 Fields

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

Table B.3.1.2.1: Contents of a KMS Certificate

Name	Description
Version	(Attribute) The version number of the certificate type (1.0.0)
Role	(Attribute) This shall indicate whether the certificate is a "Root" or "External" certificate.
CertUri	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	Date from which the Certificate may be used.
ValidTo	Date at which the Certificate expires.
Revoked	A Boolean value defining whether a Certificate has been revoked.
UserIDFormat	A string denoting how MIKEY-SAKKE UserIDs should be constructed. If the routine in Section B.3.1.4 is used, the tag should have the attribute "Conversion" set to "Hash".
PubEncKey	The SAKKE Public Key, "Z", as defined in [24]. This is an OCTET STRING encoding of an elliptic curve point.
PubAuthKey	The ECCSI Public Key, "KPAK" as defined in [14]. This is an OCTET STRING encoding of an elliptic curve point.
KmsDomainList	List of domains with which the certificate may be used.

Note that a KMS may serve multiple domains. However, each domain may have at most one KMS.

B.3.1.3 User IDs

To secure communications with a specific user, the initiator must compose the User Identifier (UID) to which the message will be encrypted. RFC 6509 [12] defines a UID format, however this is limited in terms of flexibility.

Instead, the KMS defines the UID format for its users within the KMS Certificate. This allows flexibility on the choice of UID depending on the requirements of the specific user group. The only constraint is that UID must be well-defined and derived by the initiator of the communication unambiguously.

For example, if communications occur within an IMS framework, the UID is likely to contain the user"s IMS Public identity, IMPU (e.g. user@example.org), or in other contexts, perhaps the MSISDN. For most domains, the UID used for communications will contain a reference to the current year, and also the current month or week within the year. This defines the length of time a particular UID is used, and also the key period for the key associated with the User ID.

B.3.1.4 Constructing the UserID from a KMS certificate.

B.3.1.4.0 General

The string "UserIDFormat" within the certificate contains the rules for generating the UserID based upon the user"s URI, the KMS URI, the current year, the current month, current week and the current day of the year. It is assumed that the user"s URI must be a SIP URI as defined in RFC 3261 [23].

If the user"s URI is a tel URI, the URI is first converted to a SIP URI. There are a number of ways that this conversion may be performed. Either the SIP infrastructure may perform the conversion, and refer the client to the correct URI, or the client may convert the tel URI as stated in Section 19.1.6 of RFC 3261 [23], using a default SIP domain.

B.3.1.4.1 The "Hash" UserID Conversion

The string "UserIDFormat" must contain at least one occurrence of the substrings "#uri" or "#user" and must contain the substring "#year".

The UserID is constructed as follows:

1) Initially, the UserID is set to be the UserIDFormat string.

2) Where UserID contains the substring "#uri", this substring is replaced with the NAI-part (user@host) of user"s URI as defined in Section 19.1.1 of RFC 3261 [23.

3) Where UserID contains the substring "#user", this substring is replaced with the "user" component of user"s URI as defined in Section 19.1.1 of RFC 3261 [23].

4) Where UserID contains the substring "#host", this substring is replaced with the "host" component of user"s URI as defined in Section 19.1.1 of RFC 3261 [23].

5) Where UserID contains the substring "#parameters", this substring is replaced with the "uri-parameters" component of user"s URI as defined in Section 19.1.1 of RFC 3261 [23].

6) Where UserID contains the substring "#kms", this substring is replaced with KMS URI.

7) Where UserID contains the substring "#year", this substring is replaced with the current "date-fullyear" as specified in Section 5.6 of RFC 3339 [27].

8) Where UserID contains the substring "#month", this substring is replaced with the current "date-month" as specified in Section 5.6 of RFC 3339 [27].

9) Where UserID contains the substring "#week", this substring is replaced with the current "date-week" as specified in Section 5.6 of RFC 3339 [27].

10) Where UserID contains the substring "#yday", this substring is replaced with the current "date-yday" as specified in Section 5.6 of RFC 3339 [27].

11) The resulting string is interpreted as an ASCII string and used as the User Identifier (UserID) as discussed in Section 3.2 of RFC 6509 [12].

The UserIDFormat shall not contain the "#" symbol, except to denote one of the above substrings.

B.3.1.4.2 Example UserID Construction

If a user with SIP URI: "sip:alice@example.org" is the receiver, the KMS Certificate containing the domain "example.org" is used. Assume the KMS Certificate uses the UserIDFormat:

#uri?P-Year=#year&P-Month=#month

The UserID for user with SIP URI: "sip:alice@example.org" during September 2013 is constructed as:

user@example.org?P-Year=2013&P-Month=09

As another example, of a process for converting tel URIs, assume the user has tel URI: "tel:+358-555-1234567;postd=pp22", and is in default domain "example.org". Then the UserID may be constructed by first converting the tel URI to a SIP URI:

sip:+3585551234567@example.org

then constructing the UserID as follows (for Sept 2013):

+3585551234567@example.org?P-Year=2013&P-Month=09

B.3.1.5 Signing using ECDSA and the Root KMS Certificate

The KMS may use its Root KMS Certificate as a public certificate to sign information.

Signatures are performed using ECDSA-SHA256. The domain parameters shall be the P256 elliptic curve and base-point as defined in FIPS186-4 [15]. The KMS private key shall be "KSAK" as defined in ECCSI [14], the public key shall be the "PubAuthKey" as provided with the KMS Certificate ("KPAK" as defined in ECCSI [14]).

Where the Certificate is implicitly known, the KMS may sign information by referencing the CertUri rather than including the entire Certificate.

B.3.2 User Key Provision

B.3.2.1 Description

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

B.3.2.2 Fields

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

Table B.3.2.2.1: Contents of a KMS Key Set

Name	Description
Version	(Attribute) The version number of the key provision XML (1.0.0)
KmsUri	(Optional) 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	(Optional) URI of the user for which the key set is issued.
UserID	UserID 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.
Revoked	(Optional) A Boolean value defining whether the key set has been revoked.
UserDecryptKey	The SAKKE "Receiver Secret Key" as defined in [24]. This is an OCTET STRING encoding of an elliptic curve point as defined in Section 2.2 of [25].
UserSigningKeySSK	The ECCSI private Key, "SSK" as defined in [14]. This is an OCTET STRING encoding of an integer as described in Section 6 of [26].
UserPubTokenPVT	The ECCSI public validation token, "PVT" as defined in [14]. This is an OCTET STRING encoding of an elliptic curve point as defined in Section 2.2 of [25].

B.3.3 Example KMS Response XML

B.3.3.1 Example KMSInit XML

```
<?xml version="1.0" encoding="UTF-8"?>
<KmsResponse xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance" xmlns:ds =
"http://www.w3.org/2000/09/xmldsig#" xmlns = "urn:3GPP:ns:PublicSafety:KeyManagement:2014"
xsi:schemaLocation = "urn:3GPP:ns:PublicSafety:KeyManagement:2014 IdentityKmsResponse.xsd" Id =
"xmldoc" 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">
      <KmsCertificate Version = "1.0.0" 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>2100-01-26T23:59:59</ValidTo>
        <Revoked>false</Revoked>
        <UserIdFormat Conversion="Hash">#uri?P-Year=#year&amp;amp;P-Month=#month</UserIdFormat>
        <PubEncKey>(OCTET STRING)</PubEncKey>
        <PubAuthKey>(OCTET STRING)</PubAuthKey>
        <KmsDomainList>
          <KmsDomain>sec1.example.org</KmsDomain>
          <KmsDomain>sec2.example.org</KmsDomain>
        </KmsDomainList>
      </KmsCertificate>
    </KmsInit>
  </KmsMessage>
</KmsResponse>
```

B.3.3.1 Example KMSKeyProv XML

```
<?xml version="1.0" encoding="UTF-8"?>
<KmsResponse xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance" xmlns:ds =
"http://www.w3.org/2000/09/xmldsig#" xmlns = "urn:3GPP:ns:PublicSafety:KeyManagement:2014"
xsi:schemaLocation = "urn:3GPP:ns:PublicSafety:KeyManagement:2014 IdentityKmsResponse.xsd" Id =
"xmldoc" 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">
      <KmsKeySet Version = "1.0.0">
        <KmsUri>kms.example.org</KmsUri>
        <CertUri>cert1.kms.example.org</CertUri>
```

```
        <Issuer>www.example.org</Issuer>
        <UserUri>user@example.org</UserUri>
        <UserID>user@example.org?P-Year=2014&amp;P-Month=02</UserID>
        <ValidFrom>2014-01-31T00:00:00</ValidFrom>
        <ValidTo>2014-03-01T23:59:59</ValidTo>
        <Revoked>false</Revoked>
        <UserDecryptKey>(OCTET STRING)</UserDecryptKey>
        <UserSigningKeySSK>(OCTET STRING)</UserSigningKeySSK>
        <UserPubTokenPVT>(OCTET STRING)</UserPubTokenPVT>
      </KmsKeySet>
      <KmsKeySet Version = "1.0.0">
        <KmsUri>kms.example.org</KmsUri>
        <CertUri>cert1.kms.example.org</CertUri>
        <Issuer>www.example.org</Issuer>
        <UserUri>user.psuedonym@example.org</UserUri>
        <UserID>user.psuedonym@example.org?P-Year=2014&amp;P-Month=02</UserID>
        <ValidFrom>2014-01-31T00:00:00</ValidFrom>
        <ValidTo>2014-03-01T23:59:59</ValidTo>
        <Revoked>false</Revoked>
        <UserDecryptKey>(OCTET STRING)</UserDecryptKey>
        <UserSigningKeySSK>(OCTET STRING)</UserSigningKeySSK>
        <UserPubTokenPVT>(OCTET STRING)</UserPubTokenPVT>
      </KmsKeySet>
    </KmsKeyProv>
  </KmsMessage>
</KmsResponse>
```

B.3.3.1 Example KMSCertCache XML

```
<?xml version="1.0" encoding="UTF-8"?>
<KmsResponse xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance" xmlns:ds =
"http://www.w3.org/2000/09/xmldsig#" xmlns = "urn:3GPP:ns:PublicSafety:KeyManagement:2014"
xsi:schemaLocation = "urn:3GPP:ns:PublicSafety:KeyManagement:2014IdentityKmsResponse.xsd" Id =
"xmldoc" 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>
        <KmsCertificate Version = "1.0.0" 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 Conversion="Hash">#uri?P-Year=#year&amp;P-YDay=#yday</UserIdFormat>
          <PubEncKey>(OCTET STRING)</PubEncKey>
          <PubAuthKey>(OCTET STRING)</PubAuthKey>
          <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 = "#xmlcache">
              <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>
        <KmsCertificate Version = "1.0.0" 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 Conversion="Hash">#uri?P-Year=#year&amp;P-YDay=#yday</UserIdFormat>
          <PubEncKey>(OCTET STRING)</PubEncKey>
          <PubAuthKey>(OCTET STRING)</PubAuthKey>
          <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 = "#xmlcache">
              <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>
```

B.3.4 KMS Response 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 = "urn:3GPP:ns:PublicSafety:KeyManagement:2014"
targetNamespace = "urn:3GPP:ns:PublicSafety:KeyManagement:2014" 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 = "xsd:string" 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"/>
    <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"/>
    <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"/>
    <xsd:attribute name = "CacheNum" type = "xsd:integer"/>
    <xsd:anyAttribute namespace = "##other" processContents = "lax"/>
  </xsd:complexType>

  <xsd:complexType name = "SignedKmsCertificateType">
    <xsd:sequence>
      <xsd:element name = "KmsCertificate" type = "KmsCertificateType"/>
      <xsd:element ref = "ds:Signature" minOccurs = "0"/>
    </xsd:sequence>
    <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 = "CertUri" maxOccurs = "1"/>
      <xsd:element type = "xsd:anyURI" name = "KmsUri" maxOccurs = "1"/>
      <xsd:element type = "xsd:string" name = "Issuer" maxOccurs = "1" minOccurs = "0"/>
      <xsd:element type = "xsd:dateTime" name = "ValidFrom" maxOccurs = "1"/>
      <xsd:element type = "xsd:dateTime" name = "ValidTo" maxOccurs = "1"/>
      <xsd:element type = "xsd:boolean" name = "Revoked" maxOccurs = "1"/>
      <xsd:element type = "UserIdFormatType" name = "UserIdFormat" maxOccurs = "1"/>
      <xsd:element type = "xsd:hexBinary" name = "PubEncKey" maxOccurs = "1"/>
      <xsd:element type = "xsd:hexBinary" name = "PubAuthKey" maxOccurs = "1"/>
      <xsd:element name = "KmsDomainList" maxOccurs = "1">
        <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"/>
    <xsd:attribute name = "Role" type = "RoleType"/>
    <xsd:anyAttribute namespace = "##other" processContents = "lax"/>
  </xsd:complexType>

  <xsd:complexType name = "UserIdFormatType">
      <xsd:simpleContent>
         <xsd:extension base = "xsd:string">
           <xsd:attribute name = "Conversion" type = "xsd:string"/>
           <xsd:anyAttribute namespace = "##other" processContents = "lax"/>
        </xsd:extension>
     </xsd:simpleContent>
```

```
  </xsd:complexType>

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

  <xsd:complexType name = "KmsKeySetType">
    <xsd:sequence>
      <xsd:element type = "xsd:anyURI" name = "KmsUri" maxOccurs = "1" minOccurs = "0"/>
      <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" minOccurs = "0"/>
      <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:boolean" name = "Revoked" maxOccurs = "1" minOccurs = "0"/>
      <xsd:element type = "xsd:hexBinary" name = "UserDecryptKey" maxOccurs = "1" minOccurs = "0"/>
      <xsd:element type = "xsd:hexBinary" name = "UserSigningKeySSK" maxOccurs = "1" minOccurs =
"0"/>
      <xsd:element type = "xsd:hexBinary" name = "UserPubTokenPVT" maxOccurs = "1" minOccurs = "0"/>
      <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:schema>
```

Annex C (Normative): SRTP/SRTCP Profile

Each Crypto-Session (CS) in the SRTP Crypto-Session Bundle (CSB) is provisioned with a unique 16-octet SRTP Master Key and 12 octet SRTP Master Salt (both derived from the TGK). The Master Key and Salt are not used directly to protect SRTP traffic. Rather, SRTP Session Encryption Keys and Session Salt are derived for this purpose. The AES-CM Pseudo Random Function as described in RFC 3711 [11], section 4.3.3, shall be used.

SRTP/SRTCP peers shall support IETF Draft draft-ietf-avtcore-srtp-aes-gcm-17 [29]. Both SRTP and SRTCP payloads shall be encrypted using the authenticated encryption algorithm AEAD_AES_128_GCM as defined in IETF Draft draft-ietf-avtcore-srtp-aes-gcm-17 [29] . The key derivation rate shall be 0..

SRTP packets may include the Master Key Identifier tag as defined in RFC 3711 [11]. If included, the MKI tag shall contain the identifier of the GSK.

The SRTP transforms and parameters in the SRTP protection profile defined in Annex O of TS 33.328 [32] may also be used to protect the media stream.

Annex D (Normative): MIKEY Message Formats for Media Security

D.1 General aspects

D.1.0 Introduction

MIKEY is used to transport GMKs from Group Owners to UEs and GSKs from the initiating UE to the receiving UE.

GMK transmission shall use MIKEY-SAKKE messages as defined in RFC 6509 [12]. GSK transmission shall use pre-shared keys as defined in RFC 3830 [13]. GSKs may be identified using key identifiers (MKI) which are included in the KEMAC payload.

The GMK is encrypted to the UserIDs associated to the receiving UE and signed using the UserID associated to the Group. Details of this process are defined in RFC 6508 [33] and RFC 6507 [14]. The structure of the UserIDs used is defined by the KMS of the UE.

GSKs are transmitted by MIKEY in a key transport mode as defined in RFC 3830 [13]. The pre-shared key used for transmission is the GMK. The encryption and authentication keys used to transport the GSK are derived from the GMK.

The GMK and GSK shall be 16 octets in length. The size of the authentication key to be used to verify the MAC field of a MIKEY message shall be 20 octets in length.

D.1.1 MIKEY common fields

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

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

URIs within the payloads shall conform to the normalisation procedures defined in Section 6 of RFC 3986 [34].

D.2 MIKEY message structure for GMK Distribution

The MIKEY-SAKKE message shall include the Common Header payload, timestamp payload, IDRi payload, IDRr payload, SAKKE payload and ECCSI payload.

The message may also include a RAND payload and Security Properties payload. The structure of the MIKEY message carrying a GMK key shall be according to Figure D.2-1.

In the Common Header payload, the CSB ID field of MIKEY common header is not used for identification purposes and shall be set to zero. The CS# shall also be zero. The CS ID map type subfield shall be set to "Empty map" as defined in RFC 4563 [35].

The MIKEY-RAND payload is not used to derive key material and need not be included. The Security Properties payload is also not required but may also be included.

The IDRi payload shall contain the Group URI corresponding to the GMK. The IDRr payload shall contain the user URI associated to the UE. These shall be IDR payloads as defined in Section 6.6 of RFC 6043 [33]. IDRi contains the URI of the Group (i.e. the identifier for the group managed by the group owner). As it is distributing keys, it has role "KMS", Value 3. IDRr contains the URI of UE and has role "Initiator", Value 1. The ID Type field of IDRi and IDRr payloads shall be set to value 1 (=URI).

The SAKKE payload shall encapsulate the GMK to the user URI. Only one GMK key shall be transported in the SAKKE payload. The same GMK shall be encapsulated to each member of the group.

The entire MIKEY message shall be signed by including an SIGN payload providing authentication of the group owner. The signature shall be of type 2 (ECCSI). The signature shall use the (key associated with) the group URI included in IDRi.

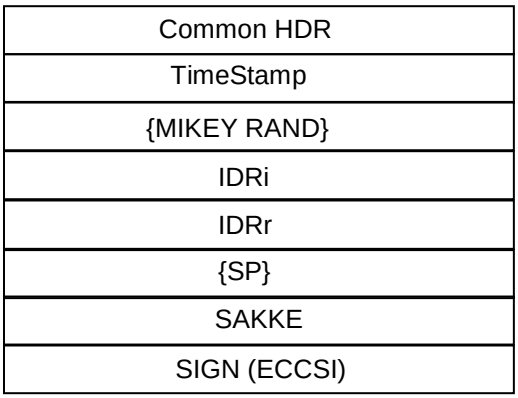

Figure D.2-1: The MIKEY GMK distribution message

D.3 MIKEY message structure for GSK distribution

GSKs are distributed using the MIKEY pre-shared key mechanism described in RFC 3830 [13]. The pre-shared key shall be the GMK.

The MIKEY message shall contain a Common Header, Timestamp payload, RAND payload, IDRi payload, IDRr payload and KEMAC payload. The Security Properties payload, a Key ID extension payload and a SIGN (ECCSI) payload may also be included. The structure of the MIKEY message carrying a GSK key shall be according to Figure D.3-1.

The Common Header payload shall include a CSB ID field, CS# and CS map type of "SRTP-ID" as defined in RFC 3830 [13].

The RAND payload shall be included as it is used to derive MIKEY internal key-encryption keys and the SRTP/SRTCP master keys/salts.

The IDRi payload shall contain the URI associated with the initiating UE. The IDRr payload shall contain the group URI. These shall be IDR payloads as defined in Section 6.6 of RFC 6043 [33]. IDRi payload has role "Initiator", Value 1. IDRr contains the URI associated with the group and has role "Receiver", Value 2. The ID Type field of IDRi and IDRr payloads shall be set to value 1 (=URI).

Security Policy (SP) payload may be included to provide information for the security protocol such as algorithms to use, key lengths, initial values for algorithms etc. See Section 6.2.4 for details on SRTP security.

The KEMAC payload shall contain the encrypted GSK. Only one GSK key can be transported in the KEMAC payload. The KEMAC payload shall use encryption algorithm AES-CM-128 as specified in RFC 3830 [13], and MAC algorithm HMAC-SHA-256-256 as specified in RFC 6043 [33]. The encryption and MAC keys shall be derived from the GMK using the key derivation function defined in section 4.1.4 of RFC 3830 [13] and using the PRF-HMAC-SHA-256 Pseudo-Random Function as described in RFC 6043 [33], Section 6.1.

Within the encrypted key-data sub-payload of the KEMAC payload, the Type subfield shall be set to TGK (value 0). The key validity data may be included. This may include an MKI or validity period.

The entire MIKEY message may be signed by including an SIGN payload providing authentication of the caller. The signature shall be of type 2 (ECCSI). The signature shall use the (key associated with) the initiating UE URI included in IDRi.

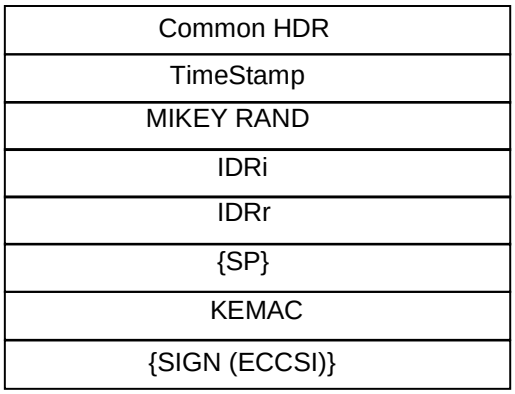

Figure D.3-1: The logical structure of the MIKEY message used to deliver GSK

D.4 Exporting the GSK to SRTP

To key SRTP, the GSK is used as the shared TEK Generation Key (TGK) described in [13]. The MIKEY Pseudo-Random Function (MIKEY-PRF) is used to derive a SRTP Master Key and Master Salt pair for each SRTP Crypto Session (CS) from the GSK (equivalently TGK). The PRF-HMAC-SHA-256 Pseudo-Random Function as described in [33], section 6.1, shall be used to perform the derivation.

The same process shall apply to key SRTCP.

Annex E (Normative): Key Request and Response messages

E.1 Introduction

This annex defines the Key Request and Key Response procedures between the UE and the ProSe Key Management Function for ProSe one-to-many communications.

E.2 Transport protocol for messages between UE and ProSe Key Management Function

The UE and ProSe Key Management Function shall use HTTP 1.1 as specified in IETF RFC 2616 [37] as the transport protocol for the messages between the UE and ProSe Key Management Function. The ProSe messages described here shall be included in the body of either an HTTP request message or an HTTP response message. The following rules apply:

- The UE initiates the transactions with an HTTP request message;
- The ProSe Key Management Function responds to the requests with an HTTP response message; and
- HTTP POST methods are used for the procedures.

E.3 XML Schema

Implementations in compliance with the present document shall implement the XML schema defined below for messages used in ProSe key management procedures.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns="urn:3GPP:ns:ProSe:KeyManagement:2014"
           elementFormDefault="qualified"
           targetNamespace="urn:3GPP:ns:ProSe:KeyManagement:2014">
        <xs:annotation>
            <xs:documentation>
                Info for ProSe Key Management Messages Syntax
            </xs:documentation>
        </xs:annotation>


  <!-- Complex types defined for parameters with complicate structure -->

  <xs:complexType name="GroupKey-Request">
    <xs:sequence>
      <xs:element name="GroupId" type="xs:integer"/>
      <xs:element name="PGKId" type="xs:integer" maxOccurs="unbounded"/>
      <xs:element name="anyExt" type="anyExtType" minOccurs="0"/>
      <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:anyAttribute namespace="##any" processContents="lax"/>
  </xs:complexType>

  <xs:complexType name="GroupKey-Reject">
    <xs:sequence>
      <xs:element name="GroupId" type="xs:integer"/>
      <xs:element name="error-code" type="xs:integer"/>
      <xs:any namespace="##any" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:anyAttribute namespace="##any" processContents="lax"/>
  </xs:complexType>


  <xs:complexType name="GroupKey-Response">
```

```
    <xs:sequence>
         <xs:element name="GroupId" type="xs:integer"/>
      <xs:element name="GroupMemberId" type="xs:integer"/>
      <xs:element name="AlgorithmInfo" type="xs:hexBinary" />
      <xs:any namespace="##any" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:anyAttribute namespace="##any" processContents="lax"/>
  </xs:complexType>

  <xs:complexType name="PMK-info">
    <xs:sequence>
      <xs:element name="PMK-ID" type="xs:hexBinary" />
      <xs:element name="PMK" type="xs:hexBinary"/>
      <xs:any namespace="##any" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:anyAttribute namespace="##any" processContents="lax"/>
  </xs:complexType>



  <!-- Complex types defined for transaction-level -->

  <xs:complexType name="KeyReq-info">
    <xs:sequence>
      <xs:element name="transaction-ID" type="xs:integer"/>
      <xs:element name="AlgorithmAvailable" type="xs:hexBinary" minOccurs="0" maxOccurs="1"/>
      <xs:element name="GroupKeyReq" type="GroupKey-Request" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="GroupKeyStop" type="xs:integer" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="anyExt" type="anyExtType" minOccurs="0"/>
      <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:anyAttribute namespace="##any" processContents="lax"/>
  </xs:complexType>

  <xs:complexType name="KeyRsp-info">
    <xs:sequence>
      <xs:element name="transaction-ID" type="xs:integer"/>
      <xs:element name="GroupNotSupported" type="GroupKey-Reject" minOccurs="0"
maxOccurs="unbounded"/>
      <xs:element name="GroupResponse" type="GroupKey-Response" minOccurs="0"
maxOccurs="unbounded"/>
      <xs:element name="Key-info" type="PMK-info" minOccurs="0"/>

      <xs:element name="anyExt" type="anyExtType" minOccurs="0"/>

      <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:anyAttribute namespace="##any" processContents="lax"/>
  </xs:complexType>

  <!--  extension allowed -->
  <xs:complexType name="KeyManagementMsgExtType">
    <xs:sequence>
      <xs:any namespace="##any" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>

  <!--  XML attribute for any future extensions  -->
  <xs:complexType name="anyExtType">
    <xs:sequence>
      <xs:any namespace="##any" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>

<!--  Top level Key Management Message definition  -->
  <xs:element name="prose-key-management-message">
    <xs:complexType>
      <xs:choice>
        <xs:element name="KEY_REQUEST" type="KeyReq-info"/>
        <xs:element name="KEY_RESPONSE" type="KeyRsp-info"/>
        <xs:element name="message-ext" type="KeyManagementMsgExtType"/>
        <xs:any namespace="##other" processContents="lax"/>
      </xs:choice>
    </xs:complexType>
  </xs:element>

</xs:schema>
```

An entity receiving the XML body ignores any unknown XML element and any unknown XML attribute.

E.4 Semantics

E.4.1 General

The < prose-key-management-message > element is the root element of this XML document and it can be one of the following elements:

- <KEY_REQUEST>;
- <KEY_RESPONSE>;
- <message-ext> element containing other ProSe key management message defined in future releases; or
- an element from other namespaces defined in future releases.

E.4.2 Semantics of <KEY_REQUEST>

The <KEY_REQUEST> element consists of:

1) <transaction-ID> element which contains the parameter defined in subclause E.5.2.2.1:
2) zero or one <AlgorithmAvailable> element contains the parameter defined in subclause E.5.2.2.2.
3) zero or more <GroupKeyRequest> element, each of which consists of:
    a) a <GroupId> element containing the parameter defined in subclause E.5.2.2.3;
    b) one or more <PGKId> element containing the parameter defined in subclause E.5.2.2.4;
    c) zero or one <anyExt> element containing elements defined in future releases;
    d) zero, one or more elements from other namespaces defined in future releases; and
    e) zero, one or more attributes defined in future releases;
4) zero or more <GroupKeyStop> element containing the parameter defined in subclause E.5.2.2.5;
5) zero or one <anyExt> element containing elements defined in future releases;
6) zero, one or more elements from other namespaces defined in future releases; and
7) zero, one or more attributes defined in future releases.

E.4.3 Semantics of <KEY_RESPONSE>

The <KEY_RESPONSE> element consists of:

1) a <transaction-ID> element which contains the parameter defined in subclause E.5.2.2.1: and
2) zero or more <GroupNotSupported> element, each of which consists of :
    a) a <GroupId> element containing the parameter defined in subclause E.5.2.2.3;
    b) a <Error-Code> element containing the parameter defined in subclause E.5.2.2.5;
    c) zero, one or more elements defined in future releases; and
    d) zero, one or more attributes defined in future releases;

3) zero or more <GroupResponse> element, each of which consists of:

   a) a <GroupId> element containing the parameter defined in subclause E.5.2.2.3;

   b) a <GroupMemberID> element containing the parameter defined in subclause E.5.2.2.6;

   c) a <AlgorithmInfo> element containing the parameter defined in subclause E.5.2.2.7;

   d) zero, one or more elements defined in future releases; and

   e) zero, one or more attributes defined in future releases;

4) zero or one <Key-info> element, each of which consists of:

   a) a <PMK-ID> element containing the parameter defined in subclause E.5.2.2.8;

   b) a <PMK> element containing the parameter defined in subclause E.5.2.2.9;

   c) zero, one or more elements defined in future releases; and

   d) zero, one or more attributes defined in future releases; 5)zero or one <anyExt> element containing elements defined in future releases;

6) zero, one or more elements from other namespaces defined in future releases; and

7) zero, one or more attributes defined in future releases.

E.5 General message format and information elements coding

E.5.1 Overview

This clause contains general message format and information elements coding for the messages used in the procedures described in the present document.

E.5.2 ProSe direct discovery message formats

E.5.2.1 Data types format in XML schema

To exchange structured information over the transport protocol, XML text format/notation is introduced.

The corresponding XML data types for the data types used in Key Management messages are provided in table E.5.5.1-1.

Table E.2.5.1-1: Primitive or derived types for ProSe Parameter Type

ProSe Parameter Type	Type in XML Schema
Integer	decimal
String	string
Boolean	boolean
Binary	hexBinary
Date and Time	dateTime

For complex data types described in subclause E.5.2.2, an XML "complexType" can be used.

Message construction shall be compliant with W3C REC-xmlschema-2-20041028: "XML Schema Part 2: Datatypes" [36]

E.5.2.2 Parameters in ProSe key management messages

E.5.2.2.1 Transaction ID

This parameter is used to uniquely identify a ProSe Key management transaction. The UE shall set this parameter to a new number for each outgoing new discovery request. The transaction ID is an integer in the 0-255 range.

E.5.2.2.2 Supported Algorithm

This parameter is used to indicate which encryption algorithm the UE supports for one-to-many commuications. It is a 1 octet long binary parameter encoded as shown in table E.5.2.2.2-1:

Table E.5.2.2.2-1: UE encryption algorithm capability information element

EPS encryption algorithms supported (octet 1)
EPS encryption algorithm EEA0 supported (octet 1, bit 8)
0 EPS encryption algorithm EEA0 not supported
1 EPS encryption algorithm EEA0 supported
EPS encryption algorithm 128-EEA1 supported (octet 1, bit 7)
0 EPS encryption algorithm 128-EEA1 not supported
1 EPS encryption algorithm 128-EEA1 supported
EPS encryption algorithm 128-EEA2 supported (octet 1, bit 6)
0 EPS encryption algorithm 128-EEA2 not supported
1 EPS encryption algorithm 128-EEA2 supported
EPS encryption algorithm 128-EEA3 supported (octet 1, bit 5)
0 EPS encryption algorithm 128-EEA3 not supported
1 EPS encryption algorithm 128-EEA3 supported
EPS encryption algorithm EEA4 supported (octet 1, bit 4)
0 EPS encryption algorithm EEA4 not supported
1 EPS encryption algorithm EEA4 supported
EPS encryption algorithm EEA5 supported (octet 1, bit 3)
0 EPS encryption algorithm EEA5 not supported
1 EPS encryption algorithm EEA5 supported
EPS encryption algorithm EEA6 supported (octet 1, bit 2)
0 EPS encryption algorithm EEA6 not supported
1 EPS encryption algorithm EEA6 supported
EPS encryption algorithm EEA7 supported (octet 1, bit 1)
0 EPS encryption algorithm EEA7 not supported
1 EPS encryption algorithm EEA7 supported

E.5.2.2.3 Group ID

This parameter is used to indicate the Group that the UE is requesting keys for. It is an integer in the 0-167777215 range.

E.5.2.2.4 PGK ID

This parameter is used to indicate the PGK IDs for a particular group. It is an integer in the 0-255 range.

E.5.2.2.5 Error Code

This parameter is used to indicate the particular reason why the UE will not be receiving keys for a requested group. It is an integer in the 0-255 range encoded as follows:

0 Reserved

1 UE does not support the required security algorithms

2 The ProSe Key Management Function does not supply keys for this group

3 UE is not authorised to receive keys for this group

4 UE requested to stop receiving PGKs for this group

5-255 Unused

E.5.2.2.6 Group Member ID

This parameter is used as the identity of the UE within a Group. It is an integer in the 0-167777215 range.

E.5.2.2.7 Algorithm Info

The purpose of the Algorithm info is to indicate the confidentilaity algorithms to be used for ciphering protection of the particular group traffic. It is a binary parameter of length 1 octet that is encoded as shown in table E.5.2.2.7-1.

Table E.5.2.2.7-1: Selected security algorithm information element

Type of ciphering algorithm (octet 1, bit 5 to 7)

Bits 7	6	5	
0	0	0	EPS encryption algorithm EEA0 (null ciphering algorithm)
0	0	1	EPS encryption algorithm 128-EEA1
0	1	0	EPS encryption algorithm 128-EEA2
0	1	1	EPS encryption algorithm 128-EEA3
1	0	0	EPS encryption algorithm EEA4
1	0	1	EPS encryption algorithm EEA5
1	1	0	EPS encryption algorithm EEA6
1	1	1	EPS encryption algorithm EEA7

Bits 1- 4 and bit 8 of octet 1 are spare and shall be coded as zero.

E.5.2.2.8 PMK ID

This parameter is used to identify a PMK. It is an 8 octet long binary parameter.

E.5.2.2.9 PMK

This parameter is a key that is used to protect MIKEY messages. It is a 32 octet long binary parameter.

Annex F (Informative): Network options for PC3 security

F.1 General

The present annex describes the network options for securing PC3 and PC8 interfaces.

F.2 Prose Function using standalone BSF

The network option for Prose Function using standalone BSF is described below:

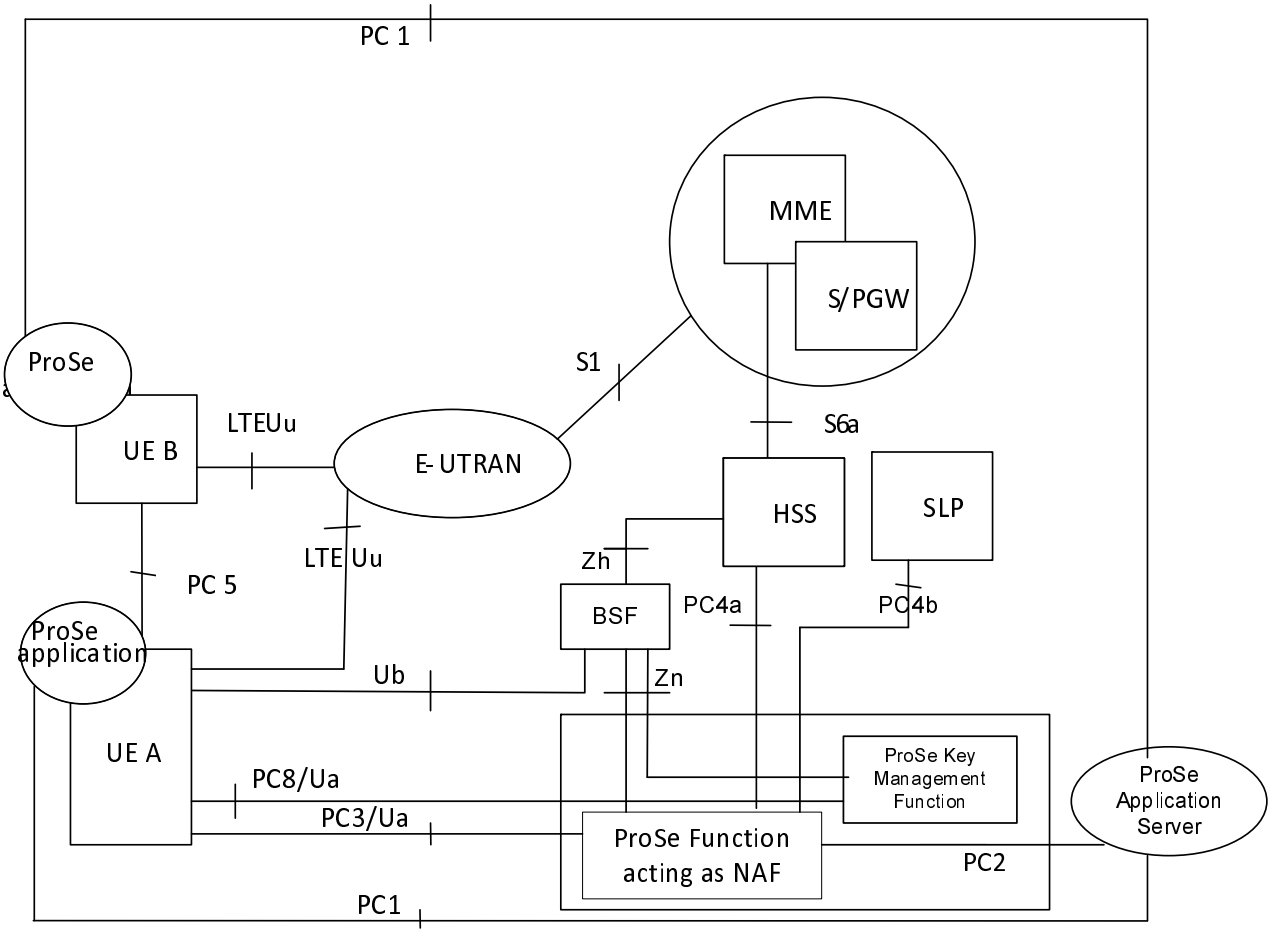

Figure F.2-1: Prose Function using standalone BSF.

This architecture follows the GBA architecture, where the ProSe Function and the ProSe Key Management Function are acting asa NAF.

F.3 BSF - Prose Function/NAF colocation

The network option for Prose Function using colocated BSF is described below:

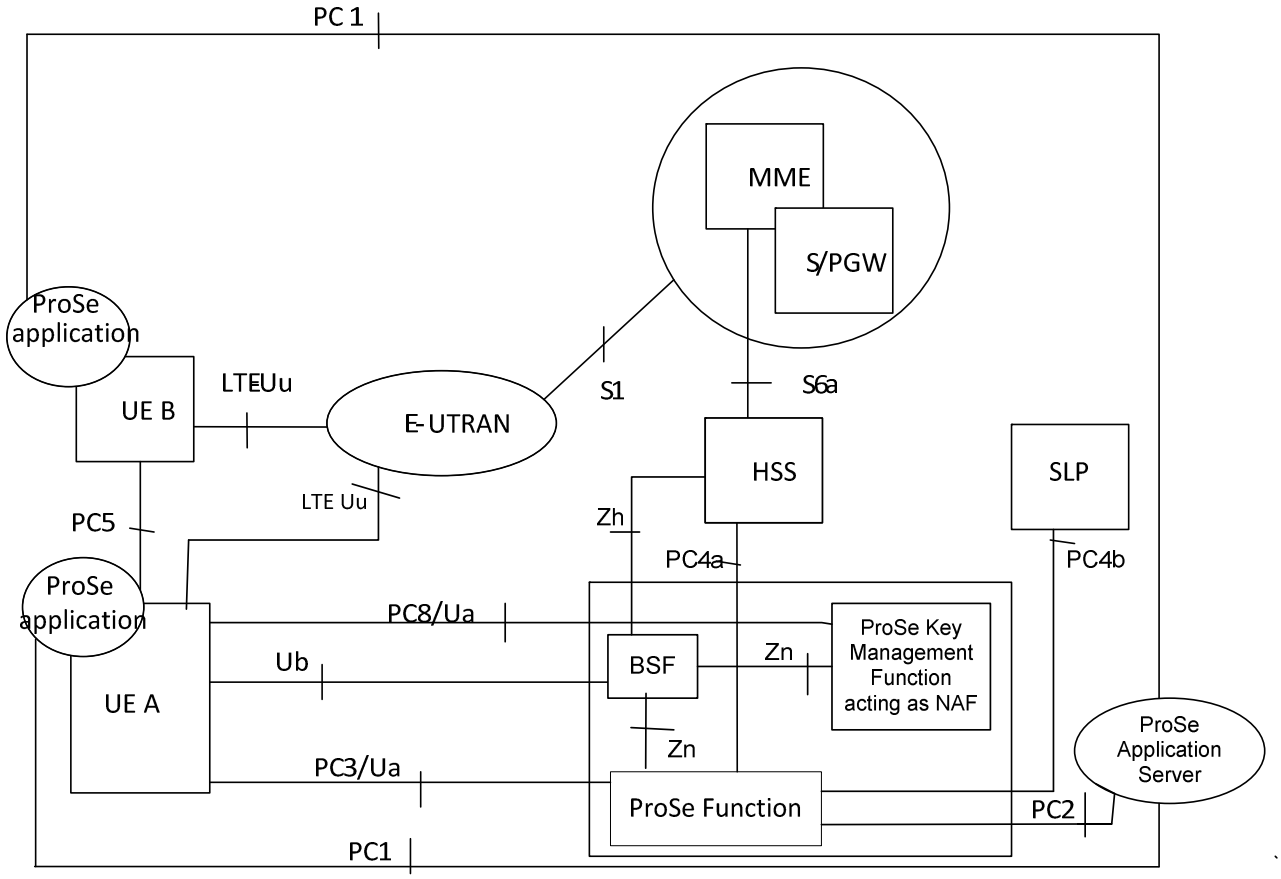

Figure F.3-1: BSF – Prose Function colocation

In this architecture the BSF is colocated with the ProSe Function and the ProSe Key Management Function acting as a NAF. Zn interface is present between BSF and ProSe Function and also between BSF and ProSe Key Management Function.

F.4 Prose Function with bootstrapping entity

The network option for Prose Function with bootstrapping entity is described below:

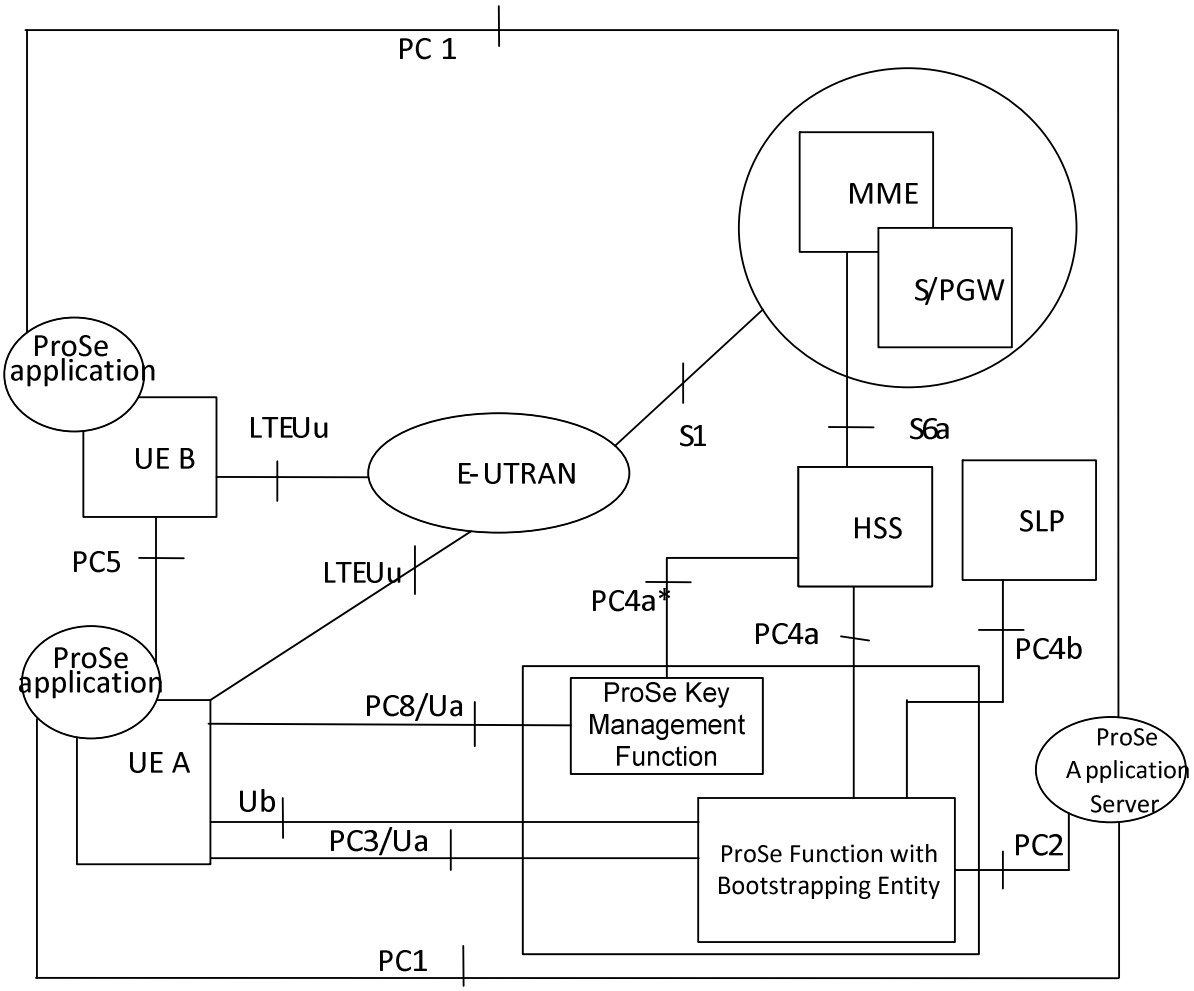

Figure F.4-1: Prose Function with bootstrapping entity

In this architecture the ProSe Function and the ProSe Key Management Function are enhanced in the way that the subset of BSF and NAF functionality towards the UE is reused, specifically the protocols defined for Ub and Ua reference points. Interfaces Zn and Zh are not used. Functionality of retrieving the authentication vector in order to support the mutually authenticated key exchange is the responsibility of the PC4a interface between the ProSe Function and the HSS, or the PC4a* interface between ProSe Key Management Function and HSS.

Annex G (informative): Change history

Change history							
Date	**TSG #**	**TSG Doc.**	**CR**	**Rev**	**Subject/Comment**	**Old**	**New**
2014-09	SP-65	SP-140591	001	1	Correction of key on Ua interface	12.0.0	12.1.0
			002	1	Addition of abbreviations		
			003	2	Modification of Open Discovery Security Flows		
			004	1	ProSe PTK Calculation		
			005	1	ProSe PTK ID		
			006	1	ProSe Group Key Clarification		
			007	-	ProSe PEK clarification		
			008	1	Protection of UE identity on PC3 interface		
			009	1	Fix the procedure of Application Server-signed Proximity Request		
			011	1	Discovery Filter using ProSe Application Mask		
			012	2	EPS encryption algorithm in ProSe Direct Communication		
			014	1	Deletion of parameters related to PGK key		
			016	1	Correct the conditions for sending of Match Report in ProSe Direct Discovery		
			017	2	Alignment of identifier definitions for ProSe one-to-many communications		
			018	1	Sending only the LSB of PGK Id in the PDCP header		
			022	1	Defining for the time parameter used in ProSe discovery		
			024	1	Support for multiple PDCP entities for a Group		
			027	1	KMS Provisioning Message Formats for media security		
			028	1	SRTP/SRTCP Profile for ProSe Media Security		
			029	1	MIKEY message formats for media security		
			030	1	Clarification on link between Group UIDs and GMKs		
2014-12	SP-66	SP-140828	019	2	Adding the details of the PGK delivery	12.1.0	12.2.0
			020	3	Adding details of the PC3 message security		
			023	2	Clarifying PTK handling for one-to-many communications		
			035	1	Considerations on security for EPC supported WLAN direct discovery and communication		
			037	1	Algorithms for ProSe communication		
			038	1	Clarification of service authorization		
			039	1	PGK Usage clarification		
			040	-	Add Missing Acronym		
			041	1	Including PTK ID into the encryption algorithm input		
			042	1	Removing editors notes from one-to-many security requirements		
			043	1	Alignment of Group Identity and Group Member Identity with RAN specifications		
			045	1	Correction and Clarification on ProSe UE behaviour		
			048	1	Calculation of the MIC value		
			049	-	Algorithm distinguisher		
			050	-	Correction of missing figure		
2015-03	SP-67	SP-150150	052	1	Discovery Slot Clarification	12.2.0	12.3.0
			053	-	Clarifying PTK Identity description in TS 33.303		
			054	1	Clarification of one-to-many requirements		
			055	1	Security information in PDCP Header		
			056	1	Adding a MIC Check timer to direct discovery procedures		
2015-06	SP-68	SP-150300	057	1	ProSe NAF Key Indication	12.3.0	12.4.0
			060	1	Annex F correction to show ProSe Key Management Function KMS and PC8		
			062	1	Correction of the hash input parameters for proximity request		
			063	1	Corretion on PTK ID handing		
			064	1	PGK expiration		
			065	-	Clarifying MIC Calculation is based on parameters, not the discovery message itself		
			067	-	Clarification on PC3ch (charging interface) security		
			068	-	Correction of standalone BSF term		
2015-09	SP-69	SP-150472	073	2	Matching scope section with content of spec - ProSe Rel-12	12.4.0	12.5.0
			075	1	Correction to terminology - ProSe Rel-12		
			079	-	Correction of the MIKEY message details for one-to-many comms		
			082	1	Correction of the XML for Key Requests and Response		
2015-12	SP-70	SP-150726	096	1	Annex A update of MIC computation for discovery messages	12.5.0	12.6.0
			105	2	Usage of AEAD_AES_128_GCM_12 in ProSe		
			107	-	Alignment with 23.303		

Change history							
Date	**Meeting**	**TDoc**	**CR**	**Rev**	**Cat**	**Subject/Comment**	**New version**
2016-06	SA#72	SP-160440	0131	-	F	Correcting a mis-numbered clause	12.7.0

History

Document history		
V12.1.0	October 2014	Publication
V12.2.0	January 2015	Publication
V12.3.0	April 2015	Publication
V12.4.0	July 2015	Publication
V12.5.0	October 2015	Publication
V12.6.0	January 2016	Publication
V12.7.0	August 2016	Publication