ETSI TS 133 303 V15.0.0 (2018-07)

Universal Mobile Telecommunications System (UMTS);
LTE;
Proximity-based Services (ProSe);
Security aspects
(3GPP TS 33.303 version 15.0.0 Release 15)

Reference
RTS/TSGS-0333303vf00

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

Intellectual Property Rights

Essential patents

IPRs essential or potentially essential to normative deliverables 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.

Trademarks

The present document may include trademarks and/or tradenames which are asserted and/or registered by their owners. ETSI claims no ownership of these except for any which are indicated as being the property of ETSI, and conveys no right to use or reproduce any trademark and/or tradename. Mention of those trademarks in the present document does not constitute an endorsement by ETSI of products, services or organizations associated with those trademarks.

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) ;
- One-to-one ProSe direct communication for ProSe-enabled Public Safety UEs (clause 6.5);
- Prose Public Safety Discovery (clause 6.6);
- Prose UE-to-network relays (clause 6.7);

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] Void.

[11] Void.

[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] Void.

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

[25] Void.

[26] Void.

[27] Void.

[28] Void.

[29] Void.

[30] Void.

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

[32] Void.

[33] Void.

[34] Void.

[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".

[41] 3GPP TS 29.368: "Tsp interface protocol between the MTC Interworking Function (MTC-IWF) and Service Capability Server (SCS)".

[42] 3GPP TS 33.102: "3G Security; Security architecture".

[43] 3GPP TS 33.179: "Security of Mission Critical Push-To-Talk (MCPTT)".

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].

Application Level Container: See 3GPP TS 23.303 [2]

Discovery Filter: See [2]

Discovery Group ID: See [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]

ProSe Query Code: See [2]

ProSe Response Code: See [2]

ProSe Restricted Code: See [2]

Relay Service Code: See [2]

Restricted ProSe Application User ID: 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
ECCSI	Elliptic Curve-based Certificateless Signatures for Identity-based Encryption
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
PIK	ProSe Integrity Key
PFID	ProSe Function ID
PGK	ProSe Group Key
ProSe	Proximity-based Services
PSDK	Public Safety Discovery Key
PTK	ProSe Traffic Key
RPAUID	Restricted ProSe Application User ID
RSC	Relay Service Code
RTP	Real-Time Transport Protocol
RTCP	RTP Control Protocol
SAKKE	Sakai-Kasahara Key Encryption
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.

5.4 Security of the PC2 reference point

5.4.1 Requirements on PC2 reference point

When the the ProSe Application Server is controlled by a 3rd party, then the PC2 reference point shall fulfil the following requirements:

Integrity protection, replay protection, confidentiality protection and privacy protection for communication between the ProSe Function and ProSe Application Server shall be supported;

- mutual authentication between ProSe Function and ProSe Application Server shall be supported;

- integrity protection and replay protection shall be used;

- confidentiality protection should be used;

- privacy of the 3GPP user shall be provided (e.g. IMSI shall not be sent outside the 3GPP operator’s domain);

the ProSe Function in the 3GPP network shall be able to determine whether the ProSe Application Server is authorized to send request messages to the ProSe Function in the 3GPP network;

5.4.2 Security procedures for PC2 reference point

The security procedure defined for the Tsp interface in TS 29.368 [41] shall be supported and used for the PC2 reference point, where the ProSe Function takes the role as the MTC-IWF and the ProSe Application Server takes the role as the SCS.

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 applicable for open discovery, model A.
They are the following:

1. The **Service Authorization** procedure: The UE contacts the ProSe Function(s) in the HPLMN 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.

The same procedures are also used for model A and B restricted discovery (see TS 23.303 [2] for more details).

6.1.2 Security requirements

Procedures 1, 2, and 4 include traffic between the UE and ProSe Function, between other network entities and over the PC2 reference point, therefore the requirements in clause 5.3.2, 5.2.2 and 5.4.1 apply to this case.

In addition, for the overall open discovery procedure, the following security requirement applies:

The system shall support a method to mitigate the replay and impersonation attacks for ProSe open discovery.

For Restricted Discovery, the following requirements apply:

ProSe Restricted discovery shall allow a UE to discover only other UEs which it is currently authorized to discover. That is, the identities announced on the air interface shall be able to be protected from being understood by currently unauthorized UEs.

The possibility of tracking of UEs based on the content of their discovery messages over time should be minimized.

The possibility of replay attacks on discovery messages sent over the air interface should be minimized.

The system shall support the prevention of impersonation attacks.

The system shall support integrity protection and confidentiality protection of Restricted Discovery ProSe Codes.

NOTE 1: Any structure present in the ProSe Code before any security processing should be preserved to enable checking for matches. Preserving the structure needs to be done in a way that does not affect the security.

NOTE 2: These requirements apply to both model A and model B restricted discovery.

NOTE 3: The restricted discovery system requirements above indicate mandatory support of the various protection mechanisms. The Prose Function decides which of these mechanisms are applied to a specific discovery (i.e. it is not mandatory to apply all the protection mechanisms to each 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, except the PC2 interface which uses the procedures in clause 5.4.2.

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.

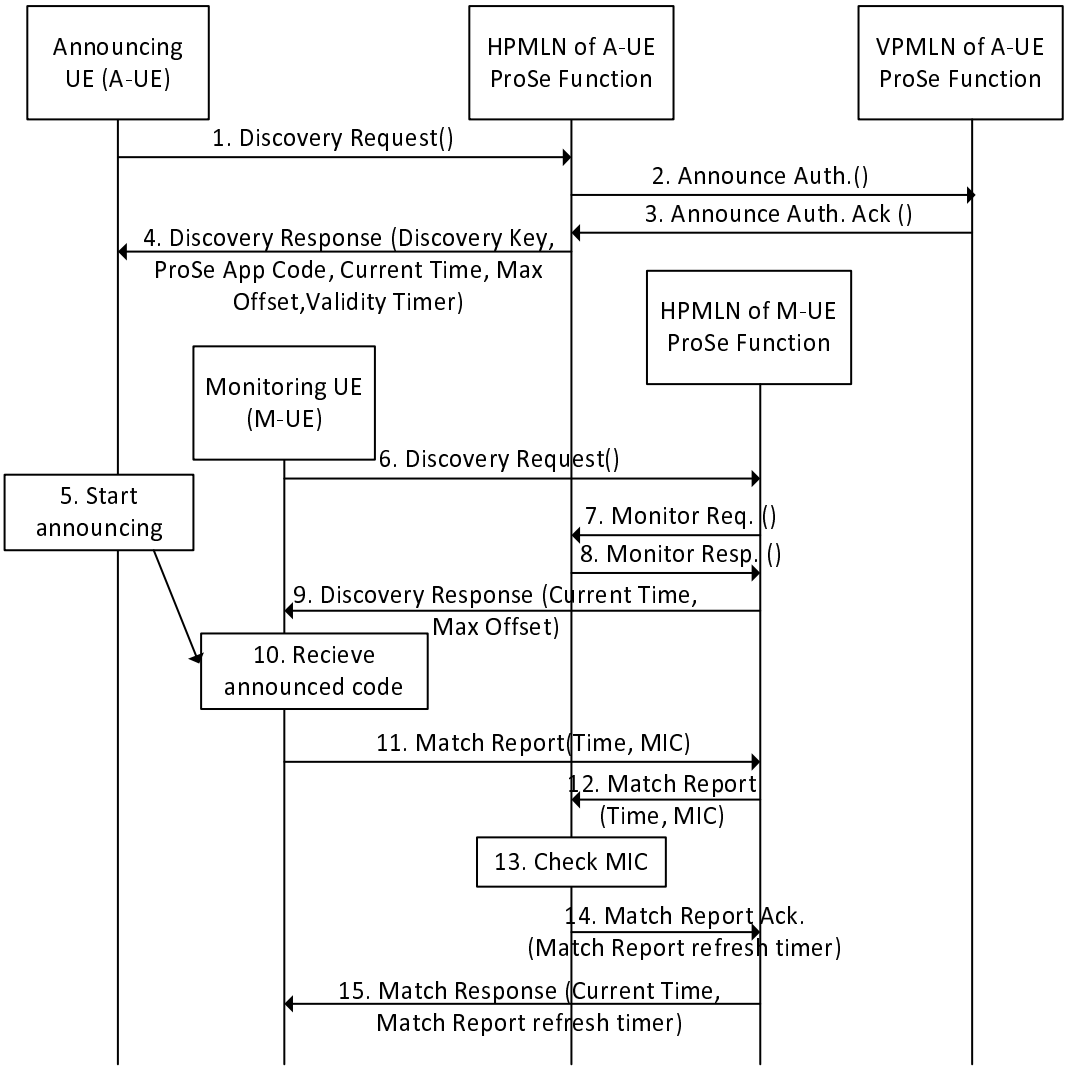

Figure: 6.1.3.3.1-1: Integrity protection of the transmitted code

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. If the announcing UE wants to send announcements while in the VPLMN, it needs to be authorised from the VPLMN ProSe Function: The ProSe Function in the HPLMN requests authorization for sending announcements from the VPLMN ProSe Function by sending Announce Auth.() message. VPLMN ProSe Function responds with an Announce Auth. Ack () message, if authorization is granted. 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 which is used for ProSe authentication (i.e. ProSe clock) to the value of CURRENT_TIME and the UE 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, via Ub interface (in GBA) (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 difference between UTC-based counter provided by the system associated with the discovery slot and the UE's ProSe clock is not greater than the MAX_OFFSET 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 difference between UTC-based counter associated with that discovery slot and UE's ProSe clock is not greater than 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..

6.1.3.4 Restricted discovery

6.1.3.4.1 General

The security for both models of restricted discovery is similar to that of open discovery described in subclause 6.1.3.3. Both models also use a UTC-based counter (see step 9 in clause 6.1.3.3) to provide freshness for the protection of the restricted discovery message on the PC5 interface. The parameters CURRENT_TIME and MAX_OFFSET are also provided to the UE from the ProSe Function in its HPLMN to ensure that the obtained UTC-based counter is sufficiently close to real time to protect against replays.

The major differences are that restricted discovery requires confidentiality protection of the discovery messages (e.g. to ensure a UE is not discovered by unauthorized parties or tracked due to constantly sending the same ProSe Restricted/Response Code in the clear) and that the MIC checking may be performed by the receiving UE (if allowed by the ProSe Function).

The security parameter needed by a sending UE to protect a discovery message (i.e., in model A the announcing UE and in model B the Discoverer UE sending the ProSe Query Code and the Discoveree UE sending the ProSe Response Code) are provided in the Code-Sending Security Parameters. Similarly the security parameters needed by a UE receiving a discovery message (i.e., in model A the monitoring UE and in model B the Discoverer UE receiving a ProSe Response Code and the Discoveree receiving a ProSe Query Code) are provided in the Code-Receiving Security Parameters.

6.1.3.4.2 Security flows

6.1.3.4.2.1 Model A security flows

This subclause contains the message flows for the protection of a model A restricted discovery. The flows show how a particular discovery is protected. The exact details of the actual protection to be applied to a discovery message are given in subclause 6.1.3.4.3. The message flows apply when both the UEs are roaming or when one or both are in their HPLMN. If either of the UEs is not roaming some steps in the flows are omitted as detailed in the flows.

The flows are broken down into 4 boxed stages to better relate to the procedures given in TS 23.303 [2]. The first, second and fourth stage correspond to subclause 5.3.3.4A, 5.3.3.5A and 5.3.4.2A of TS 23.303 [2] respectively. The interaction between the ProSe Function(s) and HSS is omitted for simplicity.

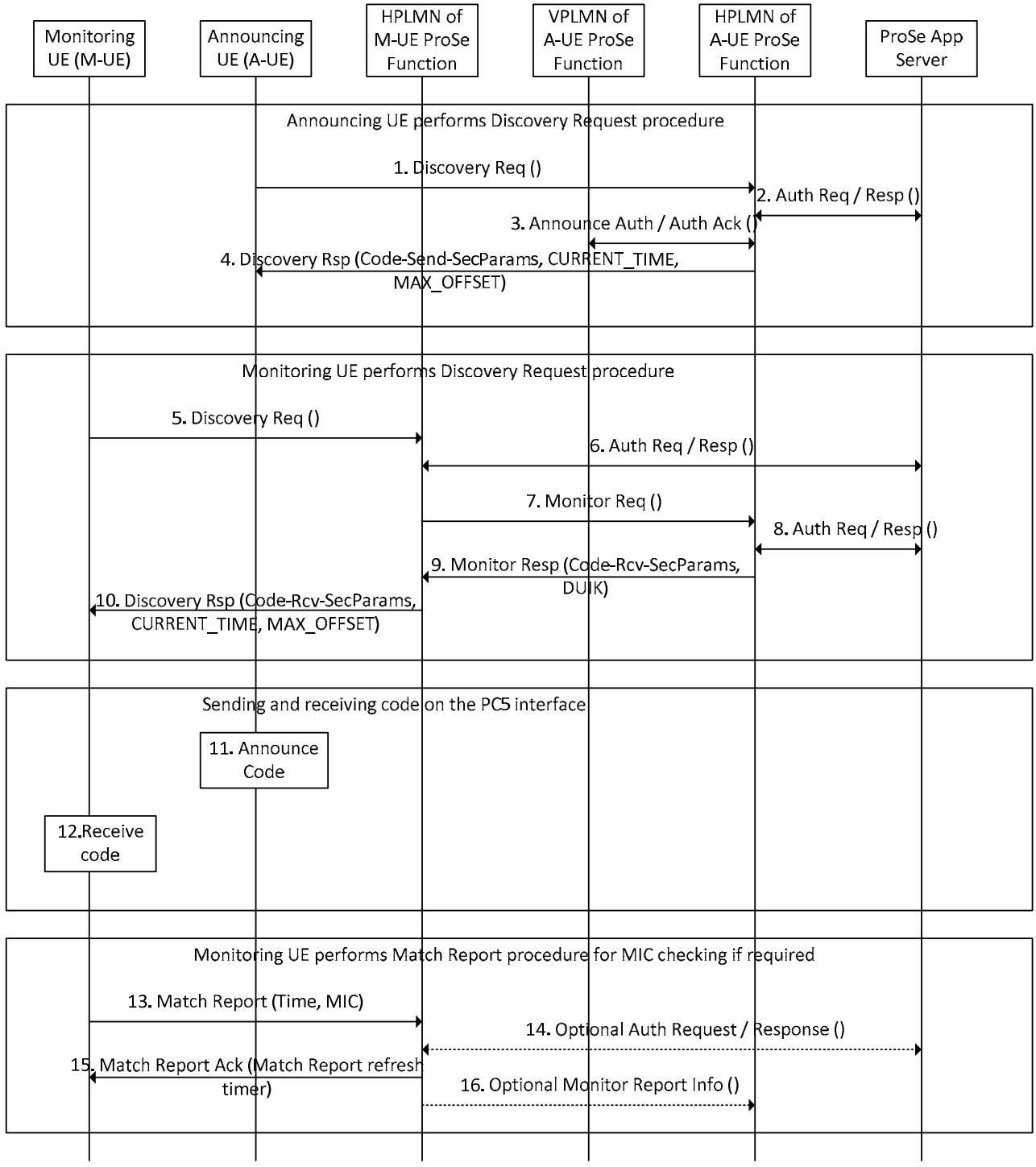

Figure 6.1.3.4.2.1-1: Flows for securing model A restricted discovery

Steps 1-4 refer to an Announcing UE.

1. Announcing UE sends a Discovery Request message containing the RPAUID to the ProSe Function in its HPLMN in order to get the ProSe Code to announce and to get the associated security material. The command indicates that this is for announce (Model A) operation, i.e. for an Announcing UE.

2. The ProSe Function may check for the announce authorization with the ProSe Application Server depending on ProSe Function configuration.

3. The ProSe Functions in the HPLMN and VPLMN of the Announcing UE exchange Announce Auth. messages. If the Announcing UE is not roaming, these steps do not take place.

4. The ProSe Function in the HPLMN of the Announcing UE returns the ProSe Code and the corresponding Code-Sending Security Parameters, along with the CURRENT_TIME and MAX_OFFSET parameters. The Code-Sending Security Parameters provide the necessary information for the Announcing UE to protect the transmission of the ProSe Code (see subclause 6.1.3.4.3.2) and are stored with the ProSe Code. The Announcing UE takes the same actions with CURRENT_TIME and MAX_OFFSET as described for the Announcing UE in step 4 of subclause 6.1.3.3.1 of the current specification.

Steps 5-10 refer to a Monitoring UE

5. The Monitoring UE sends a Discovery Request message containing the RPAUID to the ProSe Function in its HPLMN in order to be allowed to monitor for one or more Restricted ProSe Application IDs.

6. The ProSe Function in the HPLMN of the Monitoring UE sends an authorization request to the ProSe Application Server. If, based on the permission settings, the RPAUID is allowed to discover at least one of the Target RPAUIDs contained in the Application Level Container, the ProSe Application Server returns an authorization response.

7. If the Discovery Request is authorized, and the PLMN ID in the Target RPAUID indicates a different PLMN, the ProSe Function in the HPLMN of the Monitoring UE contacts the indicated PLMN's ProSe Function i.e. the ProSe Function in the HPLMN of the Announcing UE, by sending a Monitor Request message.

8. The ProSe Function in the HPLMN of the Monitoring UE may exchange authorization messages with the ProSe Application Server.

9. The ProSe Function in the HPLMN of the Announcing UE responds to the ProSe Function in the HPLMN of the Monitoring UE with a Monitor Response message including the ProSe Code, the corresponding Code-Receiving Security Parameters (see subclause 6.1.3.4.3.3) and an optional Discovery User Integrity Key (DUIK). The Code-Receiving Security Parameters provide the information needed by the Monitoring UE to undo the protection applied by the announcing UE. The DUIK shall be included as a separate parameter if the Code-Receiving Security Parameters indicate that the Monitoring UE shall use Match Reports for MIC checking. The ProSe Function in the HPLMN of the Monitoring UE stores the ProSe Code and the Discovery User Integrity Key (if it received one outside of the Code-Receiving Security Parameters).

NOTE 1: There are two configurations possible for integrity checking, namely, MIC checked by the ProSe Function, and MIC checked at the UE side. Which of the configuration is used is decided by the ProSe Function that assigned the ProSe Code being monitored, and signalled to the Monitoring UE in the Code-Receiving Security Parameters.

10. The ProSe Function in the HPLMN of the Monitoring UE returns the Discovery Filter and the Code-Receiving Security Parameters, along with the CURRENT_TIME and MAX_OFFSET parameters. The Monitoring UE takes the same actions with CURRENT_TIME and MAX_OFFSET as described for the Monitoring UE in step 9 of subclause 6.1.3.3.1 of the current specification. The UE stores the Discovery Filter and Code-Receiving Security Parameters.

Steps 11 and 12 occur over PC5.

11. 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. The UE forms the discovery message and protects it as described in 6.1.3.4.3.2. The four least significant bits of UTC-based counter are transmitted along with the protected discovery message.

12. 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. In order to find such a matching message, it processes the message as described in 6.1.3.4.3.3. If the Monitoring UE was not asked to send Match Reports for MIC checking, it stops at this step from a security perspective. Otherwise it proceeds to step 13.

NOTE 2: The UE checking the integrity of the discovery message on its own does not prevent the UE from sending a Match Report due to requirements in TS 23. 303 [2]. If such a Match Report is sent, then there is no security functionality involved.

Steps 13-16 refer to a Monitoring UE that has encountered a match.

13. If the UE has either not had the ProSe Function check the MIC for the discovered ProSe Code previously or the ProSe Function has checked a MIC for the ProSe Code and the associated Match Report refresh timer (see step 15 for details of this timer) has expired, then 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 Code and MIC. The ProSe Function checks the MIC.

14. The ProSe Function in the HPLMN of the Monitoring UE may exchange an Auth Req/Auth Resp with the ProSe App Server to ensure that Monitoring UE is authorised to discover the Announcing UE.

15. The ProSe Function in the HPLMN of the monitoring UE returns to the Monitoring UE an acknowledgement that the integrity check passed. 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 shall include the Match Report refresh timer in the message to the Monitoring UE. The Match Report refresh timer indicates how long the UE will wait before sending a new Match Report for the ProSe Code.

16. The Prose Function in the HPLMN of the Monitoring UE may send a Match Report Info message to the ProSe Function in the HPLMN of the Announcing UE.

6.1.3.4.2.2 Model B security flows

This subclause contains the message flows for the protection of a model B restricted discovery. The flows show how a particular discovery is protected. The exact details of the actual protection to be applied to a discovery message are given in subclause 6.1.3.4.3. The message flows apply when both the UEs are roaming or when one or both are in their HPLMN. If either of the UEs is not roaming some steps in the flows are omitted as detailed in the flows.

The flows are broken down into 4 boxed stages to better to relate to the procedures given in TS 23.303 [2]. The first, second and fourth stage correspond to subclause 5.3.3A.3, 5.3.3A.5 and 5.3.4A.2 of TS 23.303 [2] respectively. The interaction between the ProSe Function(s) and HSS is omitted for simplicity.

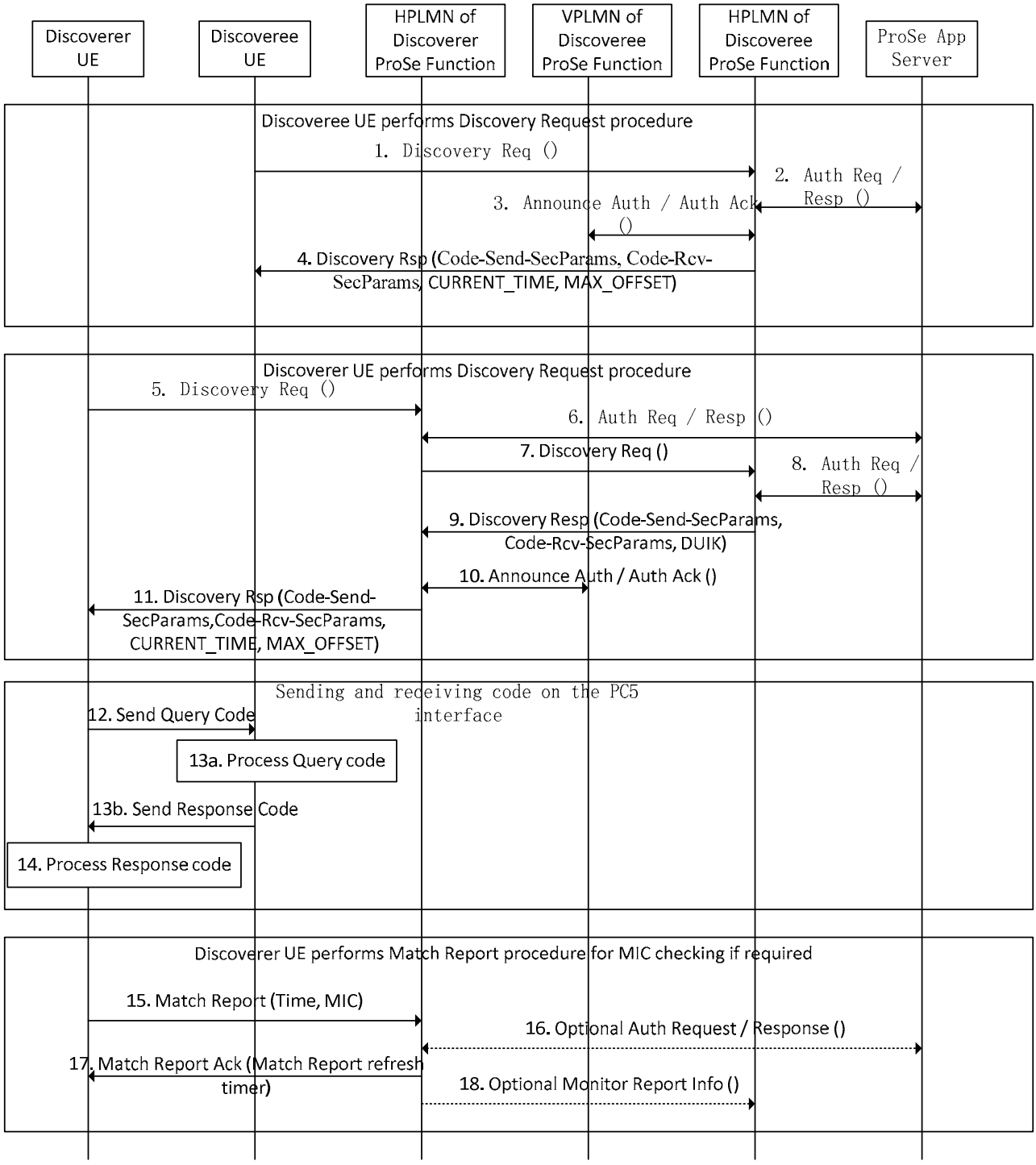

Figure 6.1.3.4.2.2-1: Flows for securing model B restricted discovery

Steps 1-4 refer to a Discoveree UE.

1. Discoveree UE sends a Discovery Request message containing the RPAUID to the ProSe Function in its HPLMN in order to get the ProSe Code to announce and associated security material. The command indicates that this is for ProSe Response (Model B) operation, i.e. for a Discoveree UE.

2. The ProSe Function may check for the announce authorization with the ProSe Application Server depending on ProSe Function configuration.

3. The ProSe Functions in the HPLMN and VPLMN of the Discoveree UE exchange Announce Auth. messages. If the Discoveree UE is not roaming, these steps do not take place.

4. The ProSe Function in the HPLMN of the Discoveree UE returns the ProSe Response Code and the Code-Sending Security Parameters, Discovery Query Filter(s) and their Code-Receiving Security Parameters corresponding to each discovery filter along with the CURRENT_TIME and MAX_OFFSET parameters. The Code-Sending Security Parameters provide the necessary information for the Discoveree UE to protect the transmission of the ProSe Response Code (see subclause 6.1.3.4.3.2) and are stored with the ProSe Response Code. The Code-Receiving Security Parameters provide the information needed by the Discoveree UE to undo the protection applied to the ProSe Query Code by the Discoverer UE (see subclause 6.1.3.4.3.3). The Code-Receiving Security Parameters shall indicate a Match Report will not be used for MIC checking. The UE stores each Discovery Filter with its associated Code-Receiving Security Parameters. The Discoveree UE takes the same actions with CURRENT_TIME and MAX_OFFSET as described for the Announcing UE in step 4 of subclause 6.1.3.3.1 of the current specification.

Steps 5-10 refer to a Discoverer UE

5. The Discoverer UE sends a Discovery Request message containing the RPAUID to the ProSe Function in its HPLMN in order to be allowed to discover one or more Restricted ProSe Application IDs.

6. The ProSe Function in the HPLMN of the Discoverer UE sends an authorization request to the ProSe Application Server. If, based on the permission settings, the RPAUID is allowed to discover at least one of the Target RPAUIDs contained in the Application Level Container, the ProSe Application Server returns an authorization response.

7. If the Discovery Request is authorized, and the PLMN ID in the Target RPAUID indicates a different PLMN, the ProSe Function in the HPLMN of the Discoverer UE contacts the indicated PLMN's ProSe Function i.e. the ProSe Function in the HPLMN of the Discoveree UE, by sending a Discovery Request message.

8. The ProSe Function in the HPLMN of the Discoveree UE may exchange authorization messages with the ProSe Application Server.

9. The ProSe Function in the HPLMN of the Discoveree UE responds to the ProSe Function in the HPLMN of the Discoverer UE with a Discovery Response message including the ProSe Query Code(s) and their associated Code-Sending Security Parameters, ProSe Response Code and its associated Code-Receiving Security Parameters (see subclause 6.1.3.4.3.3), and an optional Discovery User Integrity Key (DUIK) for the ProSe Response Code. The Code-Receiving Security Parameters provide the information needed by the Discoverer UE to undo the protection applied by the Discoveree UE. The DUIK shall be included as a separate parameter if the Code-Receiving Security Parameters indicate that the Discoverer UE shall use Match Reports for MIC checking. The ProSe Function in the HPLMN of the Discoverer UE stores the ProSe Response Code and the Discovery User Integrity Key (if it received one outside of the Code-Receiving Security Parameters). The Code-Sending Security Parameters provide the information needed by the Discoverer UE to protect the ProSe Query Code (see subclause 6.1.3.4.3.2)

NOTE 1: There are two configurations possible for integrity checking, namely, MIC checked by the ProSe Function, and MIC checked at the UE side; this is decided by the ProSe Function that assigned the ProSe Code being monitored, and signalled to the Monitoring UE in the Code-Receiving Security Parameters.

10. The ProSe Functions in the HPLMN and VPLMN of the Discoverer UE exchange Announce Auth. messages. If the Discoverer UE is not roaming, these steps do not take place.

11. The ProSe Function in the HPLMN of the Discoverer UE returns the Discovery Response Filter and the Code-Receiving Security Parameters, the ProSe Query Code and the Code-Sending Security Parameters along with the CURRENT_TIME and MAX_OFFSET parameters. The Discoverer UE takes the same actions with CURRENT_TIME and MAX_OFFSET as described for the Monitoring UE in step 9 of subclause 6.1.3.3.1 of the current specification. The UE stores the Discovery Response Filter and its Code-Receiving Security Parameters and the ProSe Query Code and its Code-Sending Security Parameters.

Steps 12 to 14 occur over PC5.

12. The Discoverer UE sends the ProSe Query Code and also listens for a response message, 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. The Discoverer UE forms the discovery

message and protects it as described in 6.1.3.4.3.2. The four least significant bits of UTC-based counter are transmitted along with the protected discovery message.

13a. The Discoveree 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 Discoverer UE's ProSe clock. In order to find such a matching message, it processes the message as described in 6.1.3.4.3.3.

NOTE 1a: Match Reports are not used for the MIC checking of ProSe Query Codes.

13b. The Discoveree sends the ProSe Response Code associated with the discovered ProSe Query Code. The Discoveree UE forms the discovery message and protects it as described in 6.1.3.4.3.2. The four least significant bits of UTC-based counter are transmitted along with the protected discovery message.

14. The Discoverer UE listens for a discovery message that satisfies its Discovery Filter. In order to find such a matching message, it processes the message as described in 6.1.3.4.3.3. If the Discoverer UE was not asked to send Match Reports for MIC checking, it stops at this step from a security perspective. Otherwise it proceeds to step 15.

NOTE 2: The UE checking the integrity of the discovery message on its own does not prevent the UE from sending a Match Report due to requirements in TS 23. 303 [2]. If such a Match Report is sent, then there is no security functionality involved.

Steps 15-18 refer to a Discoverer UE that has encountered a match.

15. If the Discoverer UE has either not had the ProSe Function check the MIC for the discovered ProSe Response Code previously or the ProSe Function has checked a MIC for the ProSe Response Code and the associated Match Report refresh timer (see step 17 for details of this timer) has expired, then the Discoverer UE sends a Match Report message to the ProSe Function in the HPLMN of the Discoverer 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 Response Code and MIC. The ProSe Function checks the MIC.

16. The ProSe Function in the HPLMN of the Discoverer UE may exchange an Auth Req/Auth Resp with the ProSe App Server to ensure that Discoverer UE is authorised to discover the Discoveree UE.

17. The ProSe Function in the HPLMN of the Discoverer UE returns to the Discoverer UE an acknowledgement that the integrity check passed. It also provides the CURRENT_TIME parameter, by which the UE (re)sets its ProSe clock. The ProSe Function in the HPLMN of the Discoverer UE shall include the Match Report refresh timer in the message to the Discoverer UE. The Match Report refresh timer indicates how long the UE will wait before sending a new Match Report for the ProSe Response Code.

18. The Prose Function in the HPLMN of the Discoverer UE may send a Match Report Info message to the ProSe Function in the HPLMN of the Discoveree UE.

6.1.3.4.3 Protection of the discovery messages over the PC5 interface

6.1.3.4.3.1 General

There are three general types of security that are used to protect the restricted discovery messages as they are transferred over the PC5 interface between the UEs.

Firstly, integrity protection is provided by appending a MIC as in Open Discovery (see subclauses 6.1.3.3.1). The MIC is calculated in the sending UE using a received Discovery User Integrity Key (DUIK) and may either be checked at the receiving UE using the supplied DUIK or at the ProSe Function using the DUIK.

Secondly, scrambling protection, which ensures that there is no relationship between the discovery messages sent by a particular UE, i.e. to prevent tracking of a UE over time. A scrambling keystream is calculated from the Discovery User Scrambling Key (DUSK) and the UTC-based counter associated with the discovery slot (see 6.1.3.4.3.5 for more details on the calculation).

Finally, message-specific confidentiality, which provides confidentiality protection for part of the discovery message. This is used either when several UEs use the same DUSK or if it is desired to obfuscate part of the discovery message

from some of the UEs that are allowed to discover the UE. A keystream is calculated from the Discovery User Confidentility Key (DUCK), the content of the message and the UTC-based counter associated with the discovery slot (see 6.1.3.4.3.6 for more details on the calculation).

The security procedures that are applied at the sending and receiving UE are controlled by the ProSe Function by sending the Code-Sending Security Parameters and/or Code-Receiving Security Parameters to the appropriate UE (i.e. the UE shall support all of integrity protection, scrambling and message specific confidentiality) To achieve integrity protection for a ProSe restricted discovery message, either a DUSK or a DUIK needs to be provided. When a DUCK is used to apply message specific confidentiality, a DUIK is required for integrity protection as more than one message is being protected. Examples of combinations of Discovery User Keys according to use case type is provided in Annex G..

At the receiving side, the scrambling protection must be undone before any matching can be attempted. Given that the operation of undoing message-specific confidentiality is computationally intense, the matching operation that precedes it should have a very minimal chance of a false positive. To this end, the ProSe Function should ensure the number of bits in a discovery message that can be matched after any scrambling has been undone is at least 16 (leading to a false positive chance of 1 in 65,536 or less).

6.1.3.4.3.2 Message Processing in the sending UE

The UE sending a discovery message receives the Code-Sending Security Parameters from the ProSe Function (as described in the security flows) to indicate how to protect the message. The Code-Sending Security Parameters may contain a DUSK and may contain a DUIK. The Code-Sending Security Parameters may contain both, a DUCK and an Encrypted_bits_mask.

The UE sending a discovery message does the following steps:

1. Form Discovery Message (e.g. add Suffix if only Prefix was allocated).
2. Calculate MIC if a DUIK was provided, otherwise set MIC to all zeros
3. Add message-specific confidentiality to the message if DUCK was received
4. Append MIC to the output of step 3.
5. Add scrambling over the output of step 4 if the DUSK was received.

6.1.3.4.3.3 Protected message processing in the receiving UE

The Code-Receiving Security Parameters received from the ProSe Function (as described in the security flows) are used to indicate to a UE how a received discovery message is protected. The Code-Receiving Security Parameters may contain a DUSK, may contain either a DUIK or an indication whether to use Match Reports for MIC checking. The Match Reports option is not allowed for ProSe Query Codes. The Code-Receiving Security Parameters may also contain both a DUCK and a corresponding Encrypted_bits_mask.

The UE receiving a Discovery Message does the following steps:

1. Undo scrambling (as in step 5 of sending UE) if a DUSK was received.
2. Check for match on the bits of the message that are not encrypted using message specific confidentiality. If no match, then abort.

NOTE 1: Some bits that the discovery filter indicates to be matched, may be encrypted by message-specific confidentiality at this stage. The UE can look for a match on the other bits after this step to minimise the amount of processing performed before finding a match.

3. Undo message-specific confidentiality if a DUCK was received (as in step 3 of sending UE)
4. Check for full match if only a match on non-encrypted bits was found in 3. If no match then abort
5. If a MIC check is required, check MIC directly (if a DUIK was given in the Discovery Filter Security Parameters) or via Match Report if indicated in the Discovery Filter Security Parameters.

NOTE 2: Requiring a checking of the MIC (at either the UE or via Match Reports) may only be omitted when the scrambling protection provides integrity protection of the bits of the message that are of interest to the receiving UE. Such integrity protection is only provided when (1) a given DUSK protects exactly one ProSe Code that the receiver matches, or (2) when message-specific confidentiality is applied to a ProSe Code but the receiving UE is not provided with the DUSK to remove the message-specific confidentiality and all the non-encrypted bits take a fixed value that the receiver matches. In the first case, if an attacker changes any bit of the message, the match will fail. In the second case, if an attacker changes a non-encrypted bit the match will fail and changing an encrypted does nothing as the receiving UE ignores these bits anyway. In latter case, the receiving UE could not successfully check the MIC.

6.1.3.4.3.4 Integrity protection description

The sending UE does the following

1. Compute output bitsequence from DUIK, Message, and UTC-based counter passed through a MIC calculation function in Annex A.2.

2. Take the 4 bytes of the output of the function and set that as the value of the MIC for this Message.

The receiving UE or ProSe Function does the exact same steps but also does a comparison between the computed MIC and the received MIC.

6.1.3.4.3.5 Scrambling description

The sending UE does the following:

1. Set the 4 LSBs of the UTC-based counter equal zero, for the purpose of this scrambling calculation only.

2. Compute the time-hash-bitsequence from DUSK and the UTC-based counter(modified as in step 1), passed through a keyed hash function.

3. XOR the time-hash-bitsequence with the entire Discovery Message (including MIC) being processed.

The receiving UE does the exact same steps except applied to the received message being processed.

6.1.3.4.3.6 Message-specific confidentiality description

The sending UE does the following:

1. Compute *Key_calc_mask* = (*Encrypted_bits_mask* XOR 0xFF..FF) || 0xFFFFFFFF

2. Calculate *Keystream* = KDF (DUCK, UTC-based counter, (*Key_calc_mas*k AND (Message || MIC))).

3. XOR (*Keystream* AND *Encrypted_bits_mask*) into the Message.

The receiving UE does the exact same steps except applied to the received Discovery Message being processed.

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).

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.
- 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. Furthermore 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.

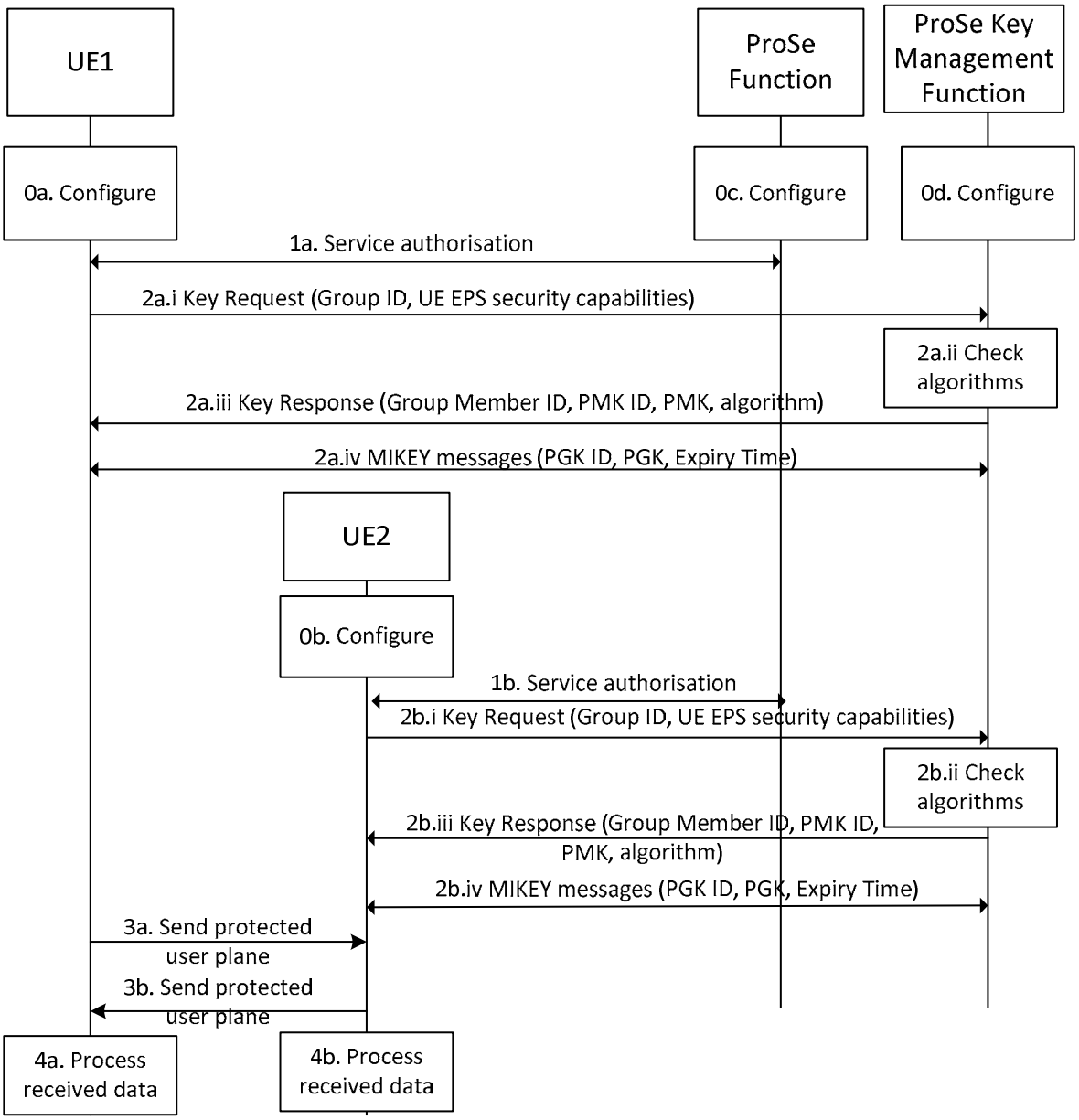

Figure 6.2.3.3-1: One-to-many security flows

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 in 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.

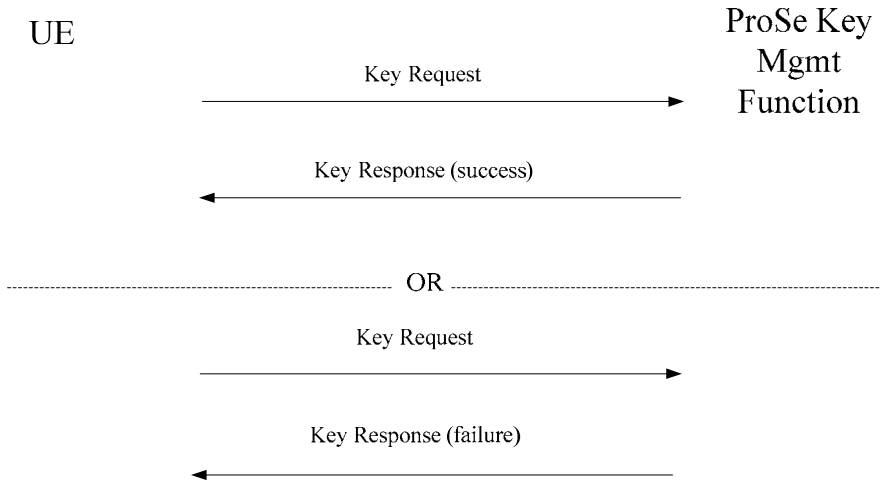

Figure 6.2.3.3.2.2-1: Key Request/Key Response transaction

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 communications;
- 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 group member identity and the security algorithm that should be used to protect the data; 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.

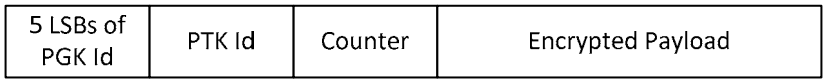

Figure 6.2.3.6.2-1: Security aspects of the PDCP packet for user plane data

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

Media security for Mission Critical Push-To-Talk (MCPTT) is defined in TS 33.179 [43]. The defined media security solution is designed to support public-safety users when operating either on-network or off-network (e.g. when using Proximity-based Services).

NOTE: The media security solution defined in TS 33.179 [43] is not back compatible with the ProSe media security solution defined in Rel-12 in this clause.

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.

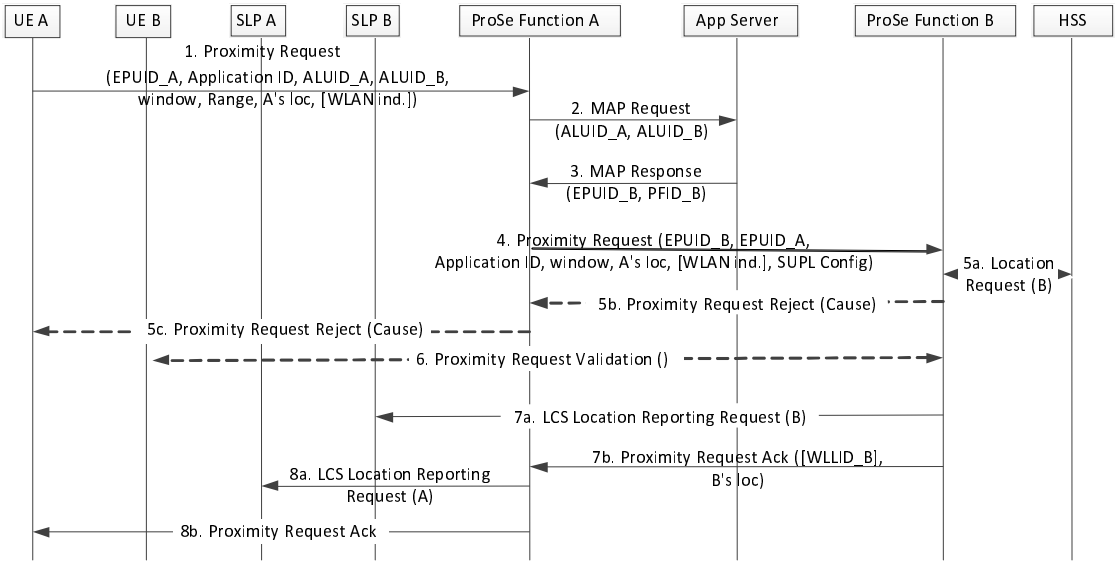

Figure 6.3.1.1-1: extract from 3GPP TS 23.303 [2], Proximity request procedure (clause 5.5.5)

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.

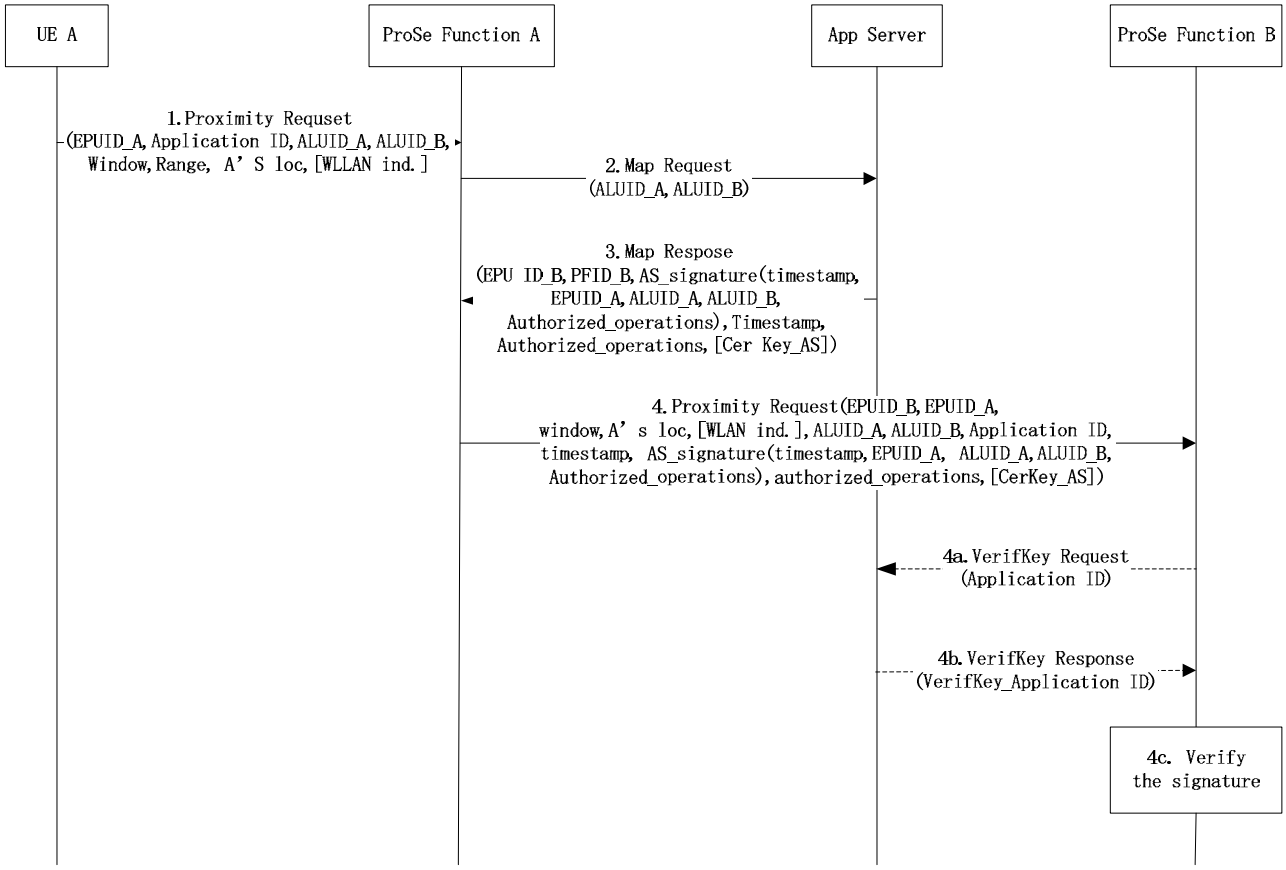

Figure 6.3.1.2-1: Application Server-signed Proximity Request

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.

6.5 Security for One-to-one ProSe Direct communication

6.5.1 General

The one-to-one ProSe Direct communication procedures are described in TS 23.303 [2]. One-to-one ProSe Direct Communication is used by two UEs that want to directly exchange traffic or when a remote UE attaches to a ProSe UE-to-network relay.

Security requirements are summarised in section 6.5.2. An overview of one-to-one ProSe Direct Communication is given in section 6.5.3. The authentication and key establishment procedures for the basic one-to-one communications are described in section 6.5.4. In section 6.5.5 the general security establishment that is used in all use cases is described.

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

6.5.2 Security Requirements

The following are the security requirements for ProSe Direct One-to-one Communication:

A ProSe-enabled UE shall use different security contexts for ProSe one-to-one communication with different ProSe-enabled UEs.

Direct link signalling ciphering shall be supported and may be used. Direct link signalling ciphering is a configuration option.

Direct link user plane ciphering shall be supported and may be used.

Direct link signalling integrity protection and replay protection shall be supported and used.

Direct link user plane packets between UEs shall not be integrity protected.

Establishment of the security between the UEs shall be protected from man-in-the-middle attacks.

The system should support mutual authentication of public safety UEs out of network coverage.

Compromise of a single UE should not affect the security of the others.

Authentication credentials should be securely stored in UE.

6.5.3 Overview of One-to-one ProSe Direct communication

6.5.3.1 Description of differet layers of keys and their identities

ProSe Direct One-to-one communication uses 4 different layers of keys. These are the following:

Long term key: This is the key that is provisioned (see the individual cases in 6.5.4 for more information on the provisioning) into the UE and is the root of the security for one-to-one communications. It may be a symmetric key or public/private key pair depending on the particular use case. Authentication signalling (denoted as “Direct Authentication and Key Establishment” - see subclause 6.5.4) is exchanged between the UEs and possibly some entities in the network, for example in the ProSe UE-to-network relay case to derive the K_D. The long term key is identified by the Long term ID.

K_D: This is a 256-bit root key that is shared between the two entities communicating using ProSe Direct one-to-one communications. It may be refreshed by re-running the authentication signalling using the Long term key. In order to generate a $K_{D\text{-}sess}$ (the next layer of keys), nonces are exchanged between the communicating entities. K_D

may be kept even when the UEs have no active one-to-one communication session between them. The K_D ID is used to identify K_D.

$K_{D\text{-}sess}$: This is the 256-bit key that is the root of the actual security context that is being used (or at least in the process of being established) to protect the transfer of data between the UEs. During a communication between the UEs, the $K_{D\text{-}sess}$ may be refreshed by running the rekeying procedure (see subclause 6.X.5.3). The actual keys (see next bullet) that are used in the confidentiality and integrity algorithms are derived directly from $K_{D\text{-}sess}$. The 16 bit $K_{D\text{-}sess}$ ID identifies the $K_{D\text{-}sess}$.

A $K_{D\text{-}sess}$ ID with a zero value indicates no security is used and hence the UEs shall not assign an all zero value of $K_{D\text{-}sess}$ ID when creating a security context.

PEK and PIK: The ProSe Encryption Key (PEK) and ProSe Integrity Key (PIK) are used in the chosen confidentiality and integrity algorithms respectively. They are derived from $K_{D\text{-}sess}$ and are refreshed automatically every time $K_{D\text{-}sess}$ is changed.

6.5.3.2 Security states

A UE may be in one of the three different security states with respect to another UE as follows:

Provisioned-security: This is where a UE just has its own long term keys.

Partial-security: This is where a UE has recently communicated with another UE and still has the K_D that it used with the other UE, but no other derived keys.

Full-security: This is where a UE is actually communicating with another UE and has K_D, $K_{D\text{-}sess}$, PEK and PIK, the chosen confidentiality and integrity algorithms and PDCP counters used with each bearer.

Once a UE ends its communication session with another UE, it shall delete $K_{D\text{-}sess}$, PEK and PIK, the choice of algorithms and the counters. It may also delete K_D.

6.5.3.3 High level overview of security establishment

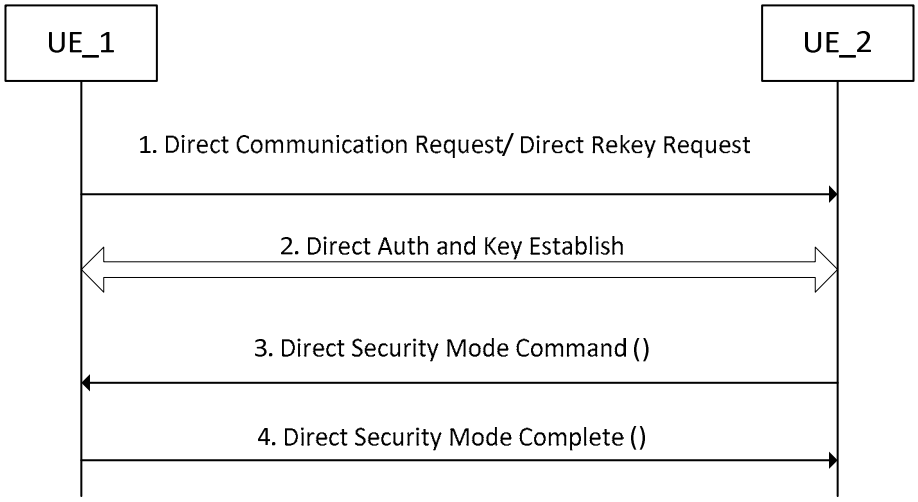

Figure 6.5.3.3-1: Overview of security establishment of ProSe Direct One-to-one communications

Figure 6.5.3.3-1 provides a high level overview of security establishment. In this flow, authentication and key establishment happens during steps 1 to 3 with the requirement that UE_2 must know K_D at the end of step 2. Step 2 may involve several messages, and these messages depend on the type of Long term key(s). More details on this are provided in subclause 6.5.4. The actual establishment of a security context happens during steps 1, 3 and 4. More details on this are provided in subclause 6.5.5.

The integrity and confidentiality protection is applied at the PDCP layer. The details of this are given in subclause 6.5.6.

6. 5.4 Direct Authentication and Key Establishment

6.5.4.1 General

There are various methods that ProSe Direct one-to-one communications may use to provide authentication and establishment of K_D. These methods may vary from case to case and the description of any necessary Direct Authentication and Key Establishment signalling that is needed in addition to the Direct Security Mode Command and Complete message will be covered with each specific case.

NOTE: None of the cases included in this release require any Direct Authentication and Key Establishment signalling.

6.5.5 Security Establishment procedures

6.5.5.1 General

There are two different cases when a security context may be established; to set up a new connection and to re-key an ongoing connection. These cases are described in the following subclauses.

6.5.5.2 Security establishment during connection set-up

The subclause describes how security is established during connection set-up. The signalling flow is shown in figure 6.5.5.2-1.

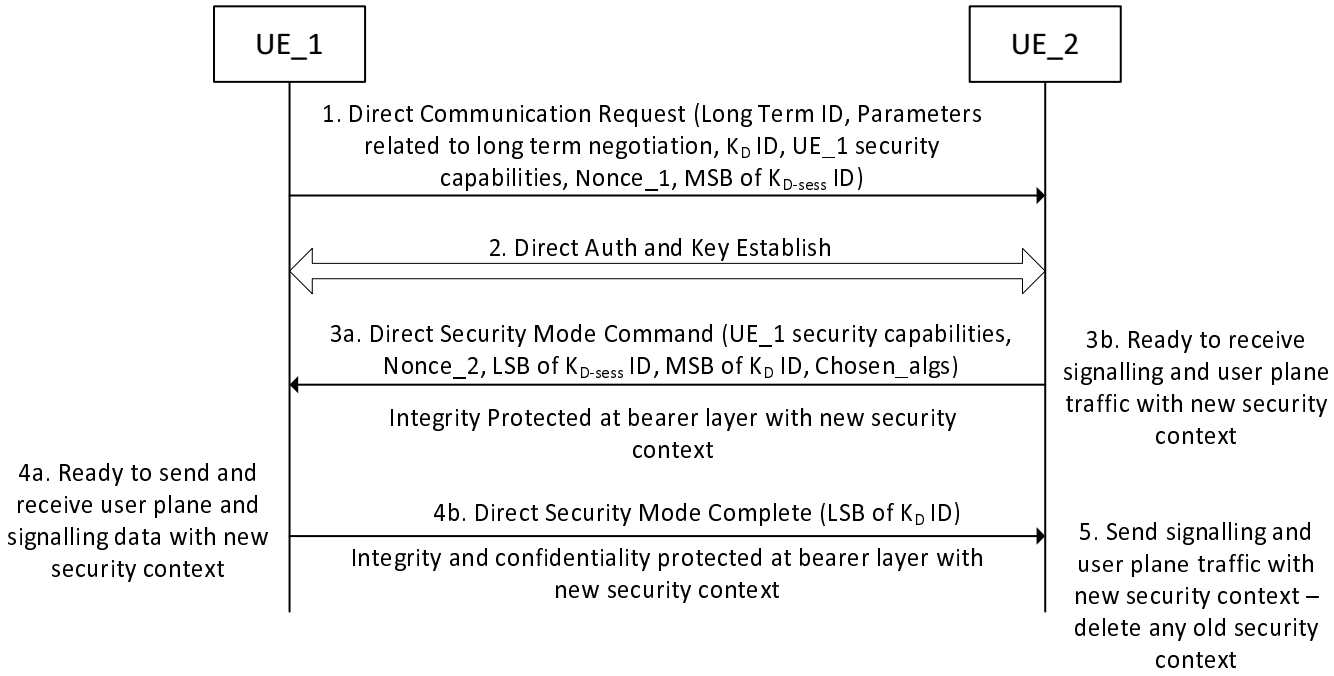

Figure 6.5.5.2-1: Security establishment at connection set-up

1. UE_1 has sent a Direct Communication Request to UE_2. This message shall include Nonce_1 (for session key generation), UE_1 security capabilities (the list of algorithms that UE_1 will accept for this connection) and the most significant 8-bits of the $K_{D\text{-sess ID}}$. These bits shall be chosen such that UE_1 will be able to locally identify a security context that is created by this procedure. The message may also include a K_D ID if the UE_1 has an exisiting K_D with the UE that it trying to communicate with. The absence of the K_D ID parameter indicates that UE_1 does not have a K_D for UE_2. The message shall also contain the necessary information to establish a K_D from the revelvant long terms keys held on the UE (see subclause 6.X.4). Long term ID is the info needed by the UE_2 in order to retrieve the right Long term Key.

2. UE_2 may initiate a Direct Auth and Key Establish procedure with UE_1. This is mandatory if the UE_2 does not have the K_D and K_D ID pair indicated in step 1, and signalling is needed to establish the keys for the particular use case.

3. UE_2 sends the Direct Security Mode Command to UE_1. It shall include the most significant bits of K_D ID if a fresh K_D is generated, Nonce_2 to allow a session key to be calculated and the Chosen_algs parameter to indicate which security algorithms the UEs will use to protect the data. The included bits of K_D ID shall uniquely identify the K_D at UE_2. UE_2 shall also return the UE_1 security capabilities to provide protection against bidding down attacks. UE_2 also includes the least significant 8-bits of $K_{D\text{-}sess}$ ID in the messages. This bits are chosen so that UE_2 will be able to locally identify a security context that is created by this procedure. UE_2 calculates $K_{D\text{-}Sess}$ from K_D and Nonce_1 and Nonce_2 (see Annex A.9) and then derives the confidentiality and integrity keys based on the chosen algorithms (Annex A.4).

   UE_2 then integrity protects the Direct Security Mode Command before sending it to UE_1. UE_2 is then ready to receive both signalling and user plane traffic protected with the new security context. UE_2 shall form the $K_{D\text{-}sess}$ ID from the most significant bits it received in message 1 and least significant bits it sent in message 3.

4. On receieving the Direct Security Mode Command, UE_1 shall calculate $K_{D\text{-}sess}$ and the confidentiality and integrity keys in the same way as UE_2. UE_1 shall check that the returned UE_1 security capabilities are the same as those it sent in step 1. UE_1 shall also check the integrity protection on the message. If both these checks pass, then UE_1 is ready to send and receive signalling and user traffic with the new security context. If most significant bits of K_D ID were included in the Direct Security Mode Command, UE_1 shall generate the least significant bits of K_D ID such that these bits uniquely identify K_D at UE_1 and shall store the complete K_D ID with K_D. UE_1 shall send an integrity protected and confidentiality protected (with the chosen algorithm which may be the null algorithm) Direct Security Mode Complete message to UE_2. UE_1 shall include the least significant bits of K_D ID in this message. UE_1 shall form the $K_{D\text{-}sess}$ ID from the most significant bits it sent in message 1 and least significant bits it received in message 3.

5. UE_2 checks the integrity protection on the received Direct Security Mode Complete. If this passes, UE_2 is now ready to send user plane data and control signalling protected with the new security context. UE_2 deletes any old security context it has for UE_1. UE_2 shall form the K_D ID from the most significant bits it sent in step 3 and least significant bits it received in the Direct Security Mode Complete.UE_2 shall store the complete K_D ID with K_D.

6.5.5.3 Rekeying security

By rekeying, the UEs ensure fresh session keys K_{D_sess} are used. Optionally the rekeying can also enforce refresh of K_D. Either UE may rekey the connection at any time. This shall be done before the counter for a particular LCID repeats with the current keys. A rekeying operation shall refresh the $K_{D\text{-}sess}$ and PEK and PIK, and may refresh K_D. A rekeying operation follows the flows given in figure 6.5.5.3-1.

NOTE 1: The UE that initiates a rekeying does not need to be the same one that iniated the connection set-up.

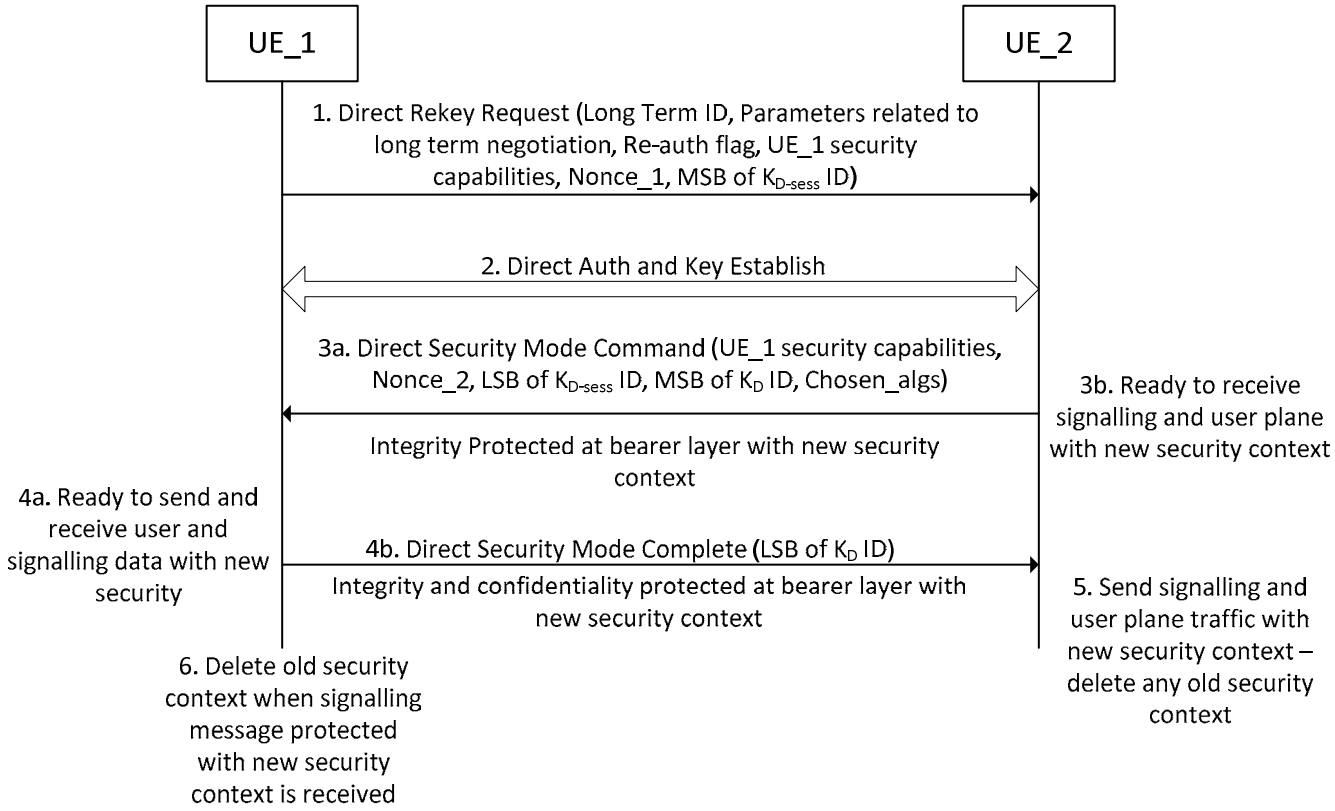

Figure 6.5.5.3-1: Security establishment during rekeying

1. UE_1 sends a Direct Rekey Request to UE_2. This message shall include Nonce_1 (for session key generation), UE_1 security capabilities (the list of algorithms that UE_1 will accept for this connection) and the most significant 8-bits of the $K_{D\text{-}sess\ ID}$. These bits are chosen such that UE_1 will be able to locally identify a security context that is created by this procedure. The message may also include a Re-auth Flag, if UE_1 wants to rekey K_D. The message shall also contain the necessary information to establish a K_D from the relevant long terms keys held on the UE (see subclause 6.5.4).

2. UE_2 may initiate a Direct Auth Key Establish procedure with UE_1. This is mandatory if UE_1 included the Re-auth Flag and signalling is needed to establish K_D.

3. This step is the same as step 3 in 6.5.5.2 except the most significant bits of K_D ID are included if a fresh K_D is generated during this rekeying.

4. This step is the same as step 4 in 6.5.5.2.

5. This step is the same as step 5 in 6.5.5.2.

6. When UE_1 receives a message integrity protected with the new security context, it shall delete any old security context is has still stored for UE_2.

6.5.6 Protection of the one-to-one traffic

6.5.6.1 General

Protection for the signalling and user plane databetween the UEs is provided at the PDCP layer. As the security is not preserved through a drop of the connection, all signalling messages that need to be sent before security is established, may be sent with no protection.

Editor's note: This exact message that may be sent without protection are FFS.

All other signalling messages shall be integrity protected and may be confidentiality protected except the Direct Security Mode Command which is sent integrity protected only.

The bearer with LCID = 28 shall be used to carry signalling messages that are not protected.

The bearer with LCID = 29 shall be used for Direct Security Mode Command and Direct Security Mode Complete.

The bearer with LCID = 30 shall be used for other signalling messages that are confidentiality and integrity protected.

The bearer with LCID = 1 to 10 may be used for user plane traffic with confidentiality protection.

6.5.6.2 Integrity protection

ProSe enabled Public Safety UEs shall implement 128-EIA1 and 128-EIA2 and may implement 128-EIA3 for integrity protection of one-to-one direct link signalling.

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

- The key used is PIK;
- Direction is set to 1 for direct link signalling transmitted by the UE that sent the Direct Security Mode Command for this security context and 0 otherwise;
- Bearer[0] to Bearer[4] are set to LCID;
- COUNT[0] to COUNT[15] are set to $K_{D\text{-}sess}$ ID;
- Counter is input into COUNT[16] to COUNT[31].

The receiving UE shall ensure that received messages are not replayed.

6.5.6.3 Confidentiality protection

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

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

- The key used in PEK;
- Direction is set as for integrity protection (see 6.5.6.2);
- Bearer[0] to Bearer[4] are set to LCID;
- COUNT[0] to COUNT[15] are set to $K_{D\text{-}sess}$ ID;
- Counter is input into COUNT[16] to COUNT[31].

6.5.6.4 Security contents in the PCDP header

The 16-bit $K_{D\text{-}sess}$ ID and 16-bit Counter parameters are carried in the PDCP header, along with any MAC that is needed for integrity protection. This is illustrated in the Figure 6.5.6.4-1.

$K_{D\text{-}sess}$ ID	Counter	(Ciphered) Payload	MAC when required

Figure 6.5.6.4-1: Security contexts of the PDCP header for one-to-one communications

If the configuration is not to use confidentiality protection for one-to-one communication user plane, then the UE shall set the values of the security information ($K_{D\text{-}sess}$ ID and Counter) to zero in the header of the user plane PDCP packets.

For the signalling messages that are not protected, the $K_{D\text{-}sess}$ and Counter in PDCP format are set to zeros in the header of the PDCP packet.

6.5.7 ProSe one-to-one commuication security using ECCSI and SAKKE

6.5.7.1 General

The solution uses the "Elliptic Curve-based Certificateless Signatures for Identity-based Encryption" (ECCSI) signature scheme, as defined in RFC 6507 [14]. And, Sakai-Kasahara Key Encryption (SAKKE), as defined in RFC 6508 [24] to generate a shared secret that is used as a K_D (root key) for establishing a secure connection between the two UEs.

6.5.7.2 Key and their identities

The UEs are provisioned with the required credentials (as defined in RFC 6507 [14] and RFC 6508[24) in advance, when the UEs have a secure access to their Key ManagementServer (KMS).

The KMS, common root of trust for the UEs, provisions the UEs with a set of credentials for ECCSI and SAKKE schemes.

Upon successful provisioning for ECCSI, each UE will be configured with the public key of the KMS, and a set of credentials associated with the UE's identity, which are: Secret Signing Key (SSK) and Public Validation Token (PVT). The UE must act as "signer" and "verifier". As a signer, the UE uses its SSK to sign a message, and when acting as a verifier, the UE uses the public key of the KMS and the signer's PVT to verify the signature.

Upon successful provisioning for SAKKE, each UE will be configured with the public key of the KMS, and a Receiver Secret Key (RSK) which is associated with the UE's identity. The sender UE uses the receiver's UE identity (receiving entity for SAKKE payload) and the public key of the KMS to create an encrypted SAKKE payload. The receiving UE uses its identity, its Receiver Secret Key and the public key of the KMS to decrypt SAKKE payload.

The public identity of a UE may be encoded in any format that is compatible with the guidelines provided in RFC 6509 [12]. For example, the public identity of a UE may be a concatenation of a fixed part (in the form of IMSI, SIP URI, TEL URI, other user@domain types of URI, etc.) and a varying part (in the form of a timestamp).

6.5.7.3 Security flows

6.5.7.3.1 Direct Connection Request

Figure 6.5.7.3.1-1 illustrates the establishment of a secure one-to-one ProSe Direct communication. It relies on the generic signalling set that is described in subclause6.5.5. The figure only includes the additional information in the messages.

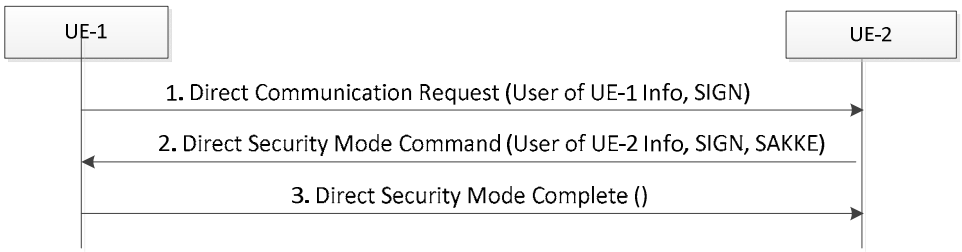

Figure 6.5.7.3.1-1: Security establishment at connection setup

1. UE-1 wishing to engage in one-to-one ProSe Direct Communication with UE-2 sends a Direct Communication Request message including the following parameters:

   - User of UE-1 Info = upper layer information identifying the user of UE-1. This information is used to derive the Signer's identifier (used by ECCSI).

   - SIGN – an ECCSI signature (as defined in RFC 6507 [14]) of the Direct Communication Request message. The signature is computed over the User of UE-1 Info parameters and the Nonce 1 (see subclause 6.5.5).

Editor's note: The exact format of input parameters to calculation of the signature is FFS

2. Upon reception of the Direct Communication Request message, UE-2 verifies the signature payload SIGN. If the verification test is successful, UE-2 presents the authenticated identity ("User of UE-1 Info") to the user of UE-2. If user of UE-2 decides to accept the request, UE-2 sends a Direct Security Mode Command message including the following parameters:

- User of UE-2 Info = upper layer information identifying the user of UE-2. This information is used to derive the Signer's identifier (used by ECCSI).

- SIGN – an ECCSI signature (as defined in RFC 6507 [14]) of the Direct Communication Response message. The signature is computed over the User of UE-2 Info, Nonce 1 and the SAKKE parameters.

Editor's note: The exact parameters of input parameters to calculation of the signature is FFS

SAKKE – UE-2 generates a 256-bit key Shared Secret Value (SSV), which is used as K_D (root key), and encodes SSV value into a SAKKE payload according to the algorithm described in RFC 6508 [24], using the KMS Pubic Key and the public identity of the user of UE-1.

Upon receipt of the Direct Security Mode message, the UE-1 verifies the signature payload SIGN. If the verification test is successful, it decrypts SAKKE payload to extract the SSV which is used as a K_D (root key) from which other keys can be derived.

3. Upon successful processing of the Direct Security Mode Command message, UE-1 responds with a Direct Security Mode Complete.

Both UEs shall store their own and the other UE's User Info with the K_D.

6.5.7.3.2 Direct Rekeying Request

Figure 6.5.7.3.2-1 illustrates rekeying of an existing secure connection when K_D is changed. It relies on the generic signalling set that is described in subclauses 6.5.5 over an existing secure connection. The figure only includes the additional information in the messages. When K_D is not changed the rekeying proceeds exactly as described in subclause 6.5.5.3.

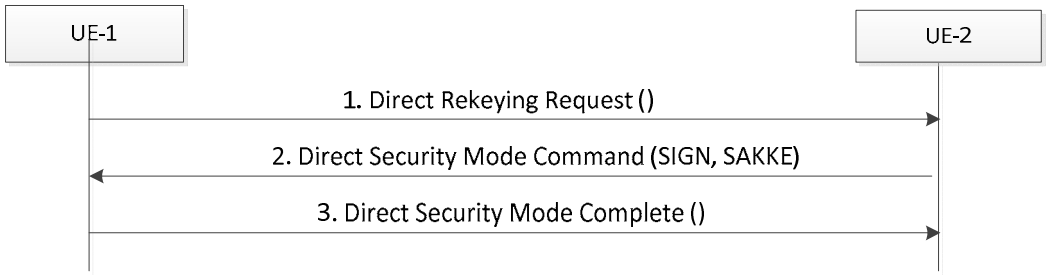

Figure 6.5.7.3.2-1: Security establishment when rekeying an existing connection

1. UE-1 send a Direct Rekeying Request to UE-2.

2. UE-2 responds with a Direct Security Mode Command message including the following parameters:

   - SIGN – an ECCSI signature (as defined in RFC 6507 [14]) of the Direct Communication Response message. The signature is computed over the User of UE-2 Info, Nonce 1 and the SAKKE parameters.

Editor's note: The exact parameters of input parameters to calculation of the signature is FFS

SAKKE – UE-2 generates a 256-bit key Shared Secret Value (SSV), which is used as K_D (root key), and encodes SSV value into a SAKKE payload according to the algorithm described in RFC 6508 [24], using the KMS Pubic Key and the public identity of the user of UE-1.

Upon receipt of the Direct Security Mode message, the UE-1 verifies the signature payload SIGN. If the verification test is successful, it decrypts SAKKE payload to extract the SSV which is used as a K_D (root key) from which other keys can be derived.

3. Upon successful processing of the Direct Security Mode Command message, UE-1 responds with a Direct Security Mode Complete.

Both UEs shall store their own and the other UE's User Info with the K_D.

6.6 Security for ProSe Public Safety Discovery

6.6.1 General

The ProSe Public Saftey Discovery procedures are described in TS 23.303 [2]. This clause details the security procedures for ProSe Public Safety Discovery.

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

6.6.2 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 used to protect the discovery message, the following requirements apply:

- The keys shall be protected in integrity and confidentiality during their distribution.
- Only authorized Public Safety ProSe-enabled UEs shall receive the keys.
- It shall be possible for the Public Safety ProSe-enabled UE to authenticate the network entity distributing the keys.
- It shall be possible for the Public Safety ProSe-enabled UE to store shared keys for past and future cryptoperiods.
- Authorized Public Safety Prose-enabled UEs shall securely store the shared keys.For the discovery messages over the PC5 interface, the following security requirements apply:
- The system shall support a method to mitigate the replay attack, source authenticity verification and integrity protection of public safety discovery messages.
- The system should provide a means of minimising the possibility of tracking of UEs based on the content of their discovery messages over time.
- The system shall support the confidentiality of information added to the messages.

6.6.3 Overview of ProSe Public Safety Discovery

6.6.3.1 General

There are two types of ProSe Public Safety Discovery described in TS 23.303 [2]: Relay Discovery (including the additonal Discovery messages) and Group Member Discovery. The security measures for both of these are identical and are reusing the following aspects:

- the key provisioning mechanism that ProSe one-to-many commuication uses, whereby a root key is fetched (the PGK – see subclause 6.2.3.1 of the present specification) along with associated security information; and
- the mechanisms defined for restricted discovery in terms of protecting the discovery messages over the air (see subclause 6.1.3.4.3 of the present specification with the needed DUIK, DUCK and DUSKs derived from the root key). It is optional to support scrambling for Public Safety Discovery.

Like open and restricted discovery, ProSe Public Safety Discovery also uses a UTC-based counter (see step 9 in clause 6.1.3.3) to provide freshness for the protection of the restricted discovery message on the PC5 interface. The parameters CURRENT_TIME and MAX_OFFSET are also provided to the UE from the PKMF to ensure that the obtained UTC-based counter is sufficiently close to real time to protect against replays.

6.6.3.2 Key and their identities

The Public Safety Discovery Key (PSDK) is the root key that is used for the protection of the Pubic Safet Discovery messages. It is identified by an 8-bit PSDK ID and each PSDK is assocciated with one or more Relay Service Codes and/or Discovery Group IDs. This association is achieved by allocating a 24-bit Key Type ID to the Relay Service Codes (RSCs) and Discovery Group IDs during the Key Request/Key Response procedure. The Key Type ID is also

included in the MIKEY message, so a delivered PSDK can be associated with the correct RSCs and/or Discovery Group IDs.

NOTE: The allocation of RSC and/or Discovery Group ID to a particular Key Type ID is specific to a UE and does not need to be common across all UEs.

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

All expired PSDK, except the most recently expired of the PSDK(s), should be deleted.

Public Safety discovery also uses the PMK and PMK ID for the MIKEY messages as described in subclauses 6.2.3.1 and 6.2.3.2 of the present specification.

6.6.4 Security flows

6.6.4.1 Overview

The configuration of the security material for the protection of direct discovery messages is shown in the figure 6.6.4.1-1:

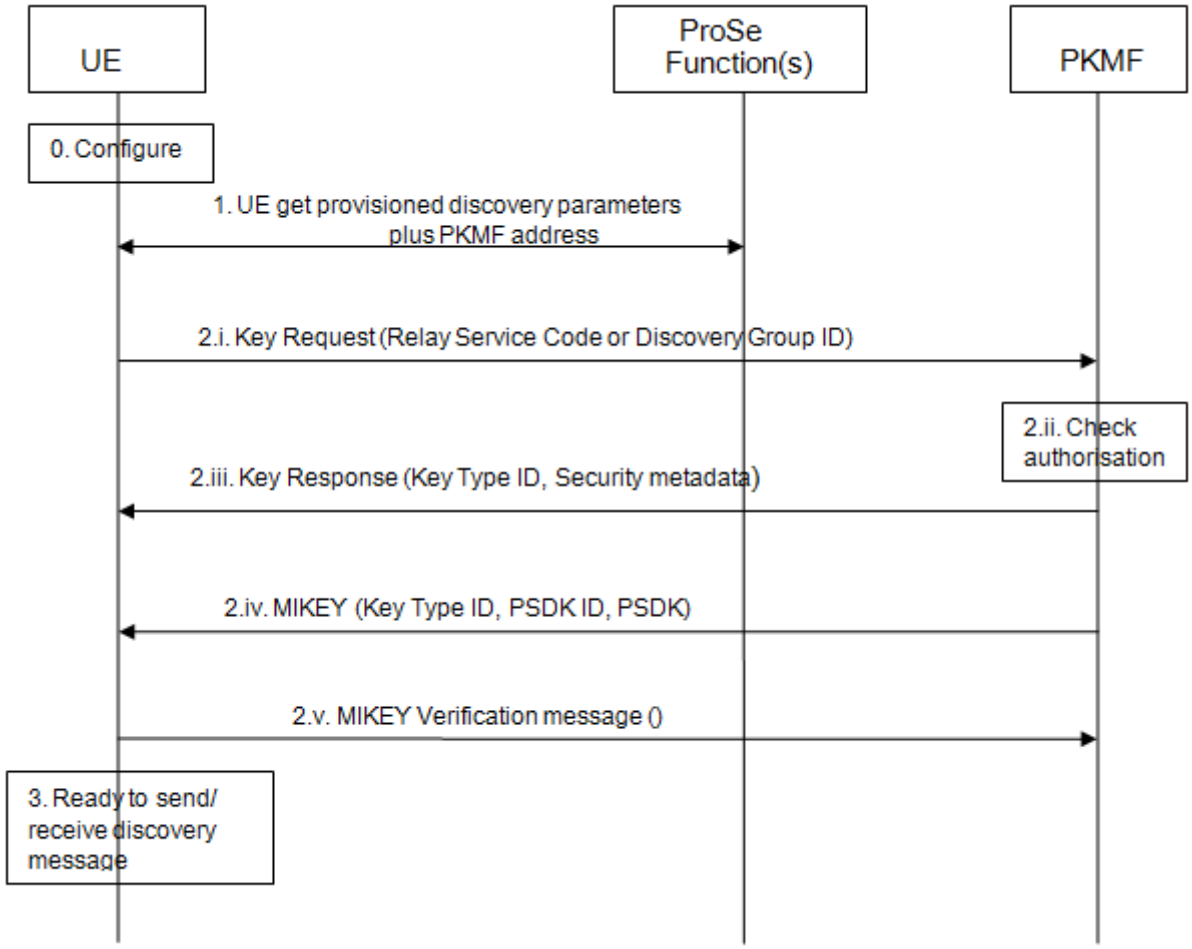

Figure 6.6.4.1-1: Configuration of parameters for public safety discovery

0: If needed the UE could be configured with any private keys, associated certificates or root certificate that may be needed 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.

1: The UE fetches the Relay Service Codes and Discovery Group IDs from its HPLMN ProSe Function. Each one of these is associated with the ProSe Key Management Function address that shall be used to fetch security parameters for these types of discovery. These parameters may also be pre-configured into the UE.

2.i: The Key Request and Response message are protected as described in subclause 6.Y.6. The UE sends the Key Request message to the ProSe Key Management Function including the relevant Relay Service Codes or the Discovery Group ID that it wishes to get security material for.

2.ii: The ProSe Key Management Function checks the authorization for the requested discoveries, and what types of protection to apply to it. The authorisation check performed by PKMF will be successful only if the Remote UE has subscribed to the relay service, the UE-to-Network relay is authorised to act as a relay, or if the requesting UE is in the group identified by the Discovery Group ID.

2.iii: The ProSe Key Management Function responds with the Key Response message. If the check of step 2.ii is successful, this message contains the *security metadata* associated with the discovery. Such *security metadata* include information on how to protect the discovery and the Key Type for that discovery.

2.iv: The ProSe Key Management Function sends the relevant PSDKs to the UE using MIKEY.

2.v. The UE responds with a MIKEY Verification message if requested by the PKMF

3. The UE is now ready to send or receive protected discovery messages.

6.6.4.2 Messages between UE and ProSe Key Management Function

6.6.4.2.1 General

There are two types of messages that are exchanged between the UE and ProSe Key Management Function. Firstly, there are the Key Request and Response messages. The UE uses these messages to request and receive the relevant PSDK and associated security material for the Public Safety Discoveries it wishes to protect. Secondly, there are the MIKEY messages, which the ProSe Key Management Function uses to send the PSDKs to the UE. These messages are detailed in the following subclauses.

6.6.4.2.2 Key Request and Key Response messages

The purpose of these messages is for the UE to request the PSDK from the ProSe Key Management Function. The UE knows from which ProSe Key Management Function(s) to get the needed PSDK(S), as the FQDN of the PKMF are either pre-provisioned or provided by the ProSe Function.

The UE shall not release the PDN connection used to receive MIKEY messages containing PSDKs until the UE has informed each ProSe Key Management Function that it no longer requires PSDK(s). This is to ensure that the ProSe Key Management Function is aware of the correct UE IP address for the purpose of performing PSDK deliveries as specified in clause 6.6.4.2.3.

If the UE detects that a PDN connection, which is used for receiving PSDKs 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 PSDK deliveries. Any new IP address should override any existing ones of the UE at the ProSe Key Management Function.

When sending a Key Request for a PSDK, the UE shall include all the relevant details of all types of keys that the UE is expecting to receive form the PKMF, e.g. any PGKs for one-to-many ProSe communication.

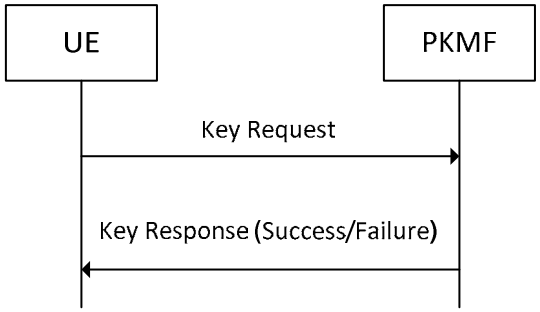

Figure 6.6.4.2.2-1: Key Request/Response message

The protection for the Key Request and Key Response message is described in subclause 6.6.6.

When sending a Key Request message to request the ProSe Key Management Function to send PSDKs or to change the Relay Service Codes or Group Discovery IDs for which it wants to receive keys, the UE shall include the following information;

- The UE's public safety discovery security capabilities, which indicate whether the UE supports scrambling or not;
- List of Relay Service Codes and Discovery Group IDs for which the UE would like to receive keys;
- For each Relay Service Code or Discovery Group ID, the PSDK ID of any keys for those discovery that the UE holds. If the UE holds no keys for this discovery, then it sends an all zero PSDK ID;
- List of Relay Service Codes and Discovery Group IDs 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 discoveries. 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. If the UE isn't then the ProSe Key Management Function responds with the appropriate error. The ProSe Key Management Function shall also check UE's public safety discovery security capabilities to find out whether UE supports scrambling or not, and responds with an appropriate error if not. The ProSe Key Management Function shall update the stored set of the discoveries 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 Relay Service Codes and Discovery Group IDs that were included in the Key Request message;
- For each discovery (relay or group member) that keys will be supplied for, the security meta-data that should be used to indicate how to protect the discovery message and the Key Type ID for the discovery; 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.
- CURRENT_TIME and MAX_OFFSET.

For the groups that the UE will get keys for, the UE shall store the received information associated with that discovery. 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 security meta-data inform the UE whether it needs to apply scrambling, message specific confidentiality and/or MIC checking (see subclause 6.1.3.4.3). If message-specific confidentiality is needed, then the security meta-data includes the Encrypted_bit_mask.

The UE uses CURRENT_TIME and MAX_OFFSET in the same way as from Open Discovery: the UE may start announcing only 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. The UE's ProSe clock is (re)set based on the provided CURRENT_TIME.

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

6.6.4.2.3 MIKEY messages

6.6.4.2.3.1 General

MIKEY is used to transport the PSDKs from the ProSe Key Management Function to the UE. MIKEY is used exactly as in subclause 6.2.3.3.2.3 (which is for carrying PGKs), except as follows:

- IDi payload is formed from the Key Type ID || PSDK ID @ FQDN of the ProSe Key Management Function at the PKMF.
- The UE recognises that it is receiving a PSDK rather than PGK based on the Key Type ID in the IDi payload.
- An all zero PSDK ID is used to trigger the UE to send a Key Request message.

6.6.5 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.

6.6.6 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 6.6.4.2.3.1.

6.6.7 Protection of discovery messages between the UEs

The protection of ProSe Public Safety Discovery Message over PC5 is very similar to that of Restricted Discovery. When sending and receieving a discovery message, the UE uses the PSDK that has not expired (using the time in the UTC based counter associated with the discovery slot to check expiry) and has the earliest expiration time to derive the needed subkeys for the security of that message.

In order to protect the discovery messages over PC5, the UE first calculates the necessary (as indicated in the security meta-data) DUSK, DUCK and DUIK for the particular discovery using the appropriate PSDK. To this end, a KDF is used to derive each of the keys indicated in the security meta-data, as follows:

- If the security meta-data indicates a DUSK should be used, then the UE derives the DUSK from the PSDK using a KDF as in Annex A.8.
- If the security meta-data indicates a DUCK should be used, and an Encrypted_bits_mask is included, then the UE derives the DUCK from the PSDK using a KDF as in Annex A.8

If the security meta-data indicates a DUIK should be used, then the UE derives the DUIK from the PSDK using a KDF as in Annex A.8.

A sending UE then follows subclause 6.1.3.4.3.2, while a receiving UE follows subclause 6.1.3.4.3.3 except that it never sends the discovery message to the ProSe Function for MIC checking.

6.7 Security for ProSe UE-to-network relays

6.7.1 General

The ProSe UE-to-network relays procedures are described in TS 23.303 [2]. This clause details the security procedures for ProSe UE-to-network relays.

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

6.7.2 Security Requirements

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

For the discovery, the security requirement in subclause 6.6.2 apply.

For the distribution of the keys used to protect the relay signalling, the following requirements apply:

- The keys shall be protected in integrity and confidentiality during their distribution.
- Only authorized Public Safety ProSe-enabled UEs shall receive the keys.

- It shall be possible for the Public Safety ProSe-enabled UE to authenticate the network entity distributing the keys.
- Authorized Public Safety Prose-enabled UEs shall securely store the shared keys.For the non-discovery communication between the Remote UE and relay, the requirements on one-to-one communication in subclause 6.5.2 apply except the one on mutual authentication of out of coverage UEs.

The system shall support mutual authentication between Remote UE and UE-to-Network Relay.

Both Remote UE and UE-to-Network Relay shall be authorised by the ProSe Function.

6.7.3 Overview of ProSe UE-to-network relay security

6.7.3.1 General

The ProSe UE-to-network relay procedures consist of two distinct phases, i.e. the discovery of the UE-to-network relay and the communication between the Remote UE and UE-to-network relay.

The security of the discovery messages uses the procedures provided in the current specification. The security of the communication between the Remote UE and UE-to-network relay uses the procedures described in clause 6.5 to establish the security context and protect the actual communication. The part of the security establishment that is specific to the UE-to-network relay use case is the establishment of the shared key K_D. The procedures for this operation are described in the next subclause.

Following the general sequence of flows for public safety one to one communication, a shared key K_D needs to be established. This key serves to derive session keys between the Remote UE and the UE-to-network relay.

In order to generate K_D, the Remote UE needs a ProSe Relay User Key (PRUK) and an associated 64-bit PRUK ID from a PKMF. The PRUK ID is used to identify the PRUK to the PKMF of the UE-to-network relay. The PRUK can be used to generate K_D for any of the relays under a particular PKMF and hence only one PRUK for each Remote UE is needed from a particular PKMF. This PRUK needs to the fetched by the Remote UE while it is still in coverage. This implies that the Remote UE must contact all the PKMFs of any potential relays it wants to be able to use.

The Remote UE fetches its PRUK from the PKMF using the Key Request/Response messages or may receive one through GBA PUSH as part of establishing the communication with the relay. The UE-to-network relay fetches the K_D that will be used to secure the communication by sending to its PKMF the PRUK ID or IMSI if the Remote UE does not have a PRUK for the relay or the supplied PRUK has been rejected. At the PKMF side, the corresponding PRUK is retrieved. The K_D is then derived from the PRUK using a K_D Freshness Parameter (a locally generated random number), which the PKMF then passes to the Remote UE via the UE-to-network Relay, a nonce sent by the Remote UE via the UE-to-network Relay and the Relay Service Code the Remote UE wishes to access. The UE-to-network Relay receives the K_D and the K_D Freshness Parameter, and stores the K_D. Having obtained the K_D Freshness Parameter enables the Remote UE to derive the same K_D as the PKMF did.

If the Remote UE receives a new PRUK in a Key Response message, it deletes any previous one for that PKMF. If it receives a new one through a GBA PUSH message, it shall overwrite any PRUK received through a GBA PUSH message that has not been successfully used to establish a relay connection. Once a PRUK received through a GBA PUSH Message has been used to calculate a K_D for a sucessful relay connection establishment, the Remote UE shall delete any previous PRUKs for this PKMF.

6.7.3.2 Security flows

6.7.3.2.1 Overview

This subclause describes the overall security flows for UE-to-network relays. These includes the basic attach flows, dealing with re-synchronisation errors in GPI and the rekeying flows.

6.7.3.2.1.1 Remote UE attaching to a ProSe UE-to-network relay

There are several possible ways that the parameters needed for UE-to-network relay can be provisioned onto the Remote UE and UE-to-network relay. The flow described in Figure 6. 7.3.2.1.1-1 is for the case when the general pararmeters are fetched from the ProSe Function(s). In other cases some of the steps can be omited if the relevant parameters are already in place. The figure shows only the parameters that are relevant to the UE-to-network relay security and not all the parameters carried by each message.

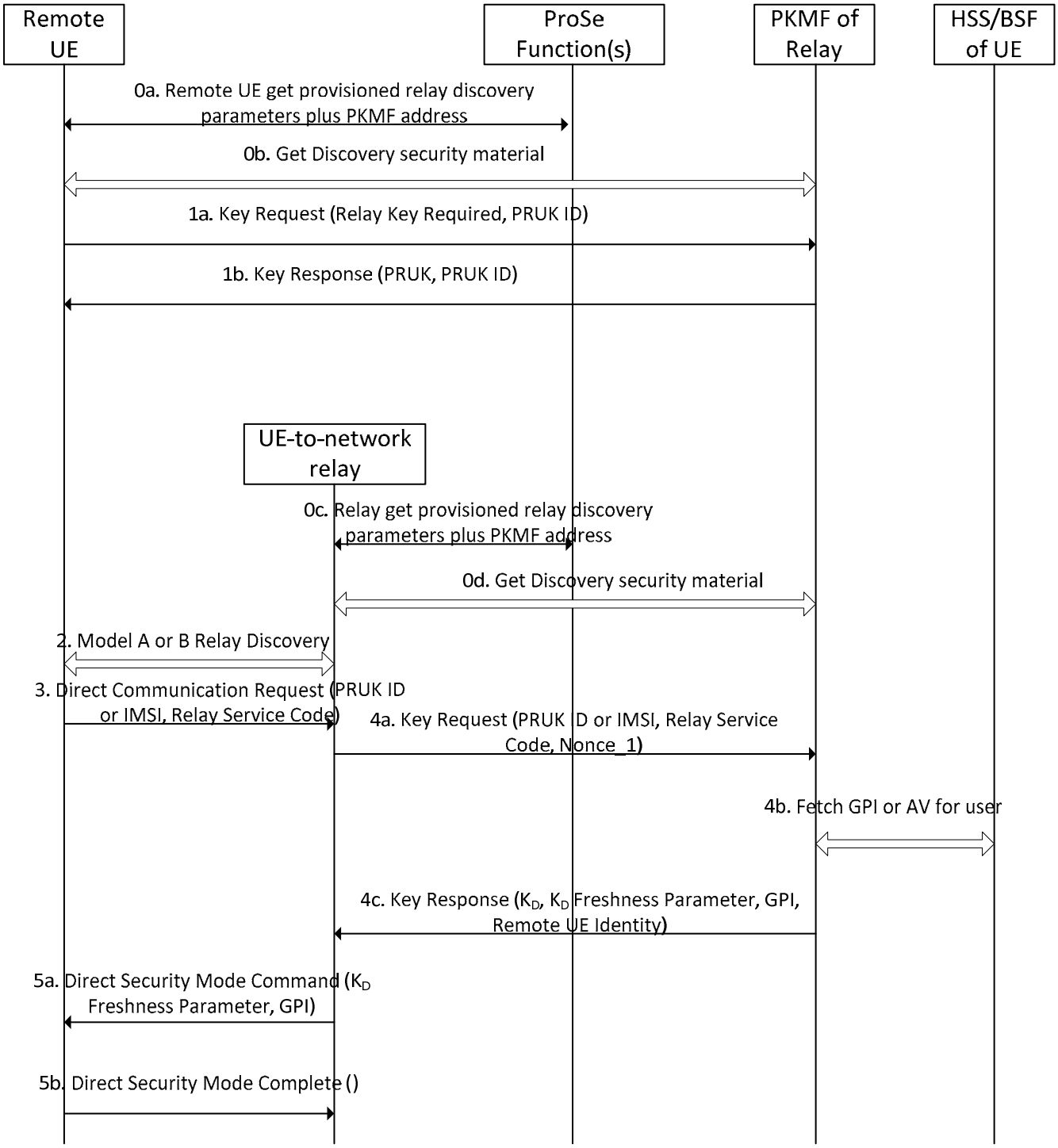

Figure 6.7.3.2.1.1-1: UE-to-network relay security flows

0. The Remote UE and the UE-to-network relay fetch the parameters necessary to act as a Remote UE and UE-to-network relay respectively (see TS 23.303[2]), the PKMF address for accessing the relay and the security parameters required to protect the relay discovery messages (see subclause 6.6).

NOTE: Part of step 0b may be performed at the same time as step 1a to 1b as the same messages are used to initiate fetching the keys for protecting relay discovery.

1a. The Remote UE sends a Key Request message to the PKMF of the UE-to-network relay. The message indicates that the Remote UE is requesting a ProSe Relay User Key (PRUK) from the PKMF. If the Remote UE already has a PRUK from this PKMF, the message shall also contain the PRUK ID of the PRUK.

1b. The ProSe Key Management Function shall check that the Remote UE is authorised to receive UE-to-network Relay service from one of its relays. This is done by using the Remote UE identity that is bound to the keys that established the TLS tunnel in which the message is sent. If the Remote UE is successfully authorised, the PKMF sends a Key Response message to the Remote UE that may contain a PRUK and PRUK ID. If a PRUK and PRUK ID are included, the Remote UE shall store these and delete any previously stored ones for this ProSe Key Management Function.

2. The Remote UE discovers the UE-to-network Relay using either model A or model B discovery.

3. The Remote UE sends a Direct Communication Request. The Long Term ID shall contain the PRUK ID of the PRUK that the Remote UE want to use to get relay connectivity if the Remote UE has a PRUK for this relay and an attempt to connect to this relay has not been rejected due to the PRUK ID not being recognised. Otherwise the Remote UE shall use its IMSI in the Long Term ID. The Direct Communication Request contains the Relay Service Code that the Remote UE would like to access.

4a. The UE-to-network relay sends a Key Request message to the PKMF. The message shall contain the PRUK ID or IMSI, the Relay Service Code and Nonce_1 (see subclause 6.5.5.2) provided by the Remote UE. The PKMF identifies the UE by the PRUK ID or IMSI. The PKMF checks the context of the Remote UE to confirm whether it can connect to the network via the selected ProSe UE-to-network Relay for the given Relay Service Code.

4b. If the PKMF confirms the Remote UE can connect to the network via the selected ProSe UE-to- network Relay, the PKMF decides if it requires a new PRUK for this UE, i.e. policy in the PKMF decides that PRUK ID needs refreshing or the relay provided the IMSI of the UE. If so the PKMF proceeds as follows:

If the PKMF supports the Zpn interface to the BSF of the UE, the PKMF shall request a GBA Push Info (GPI – see TS 33.223[38]) for the Remote UE from the BSF. When requesting the GPI, it includes a non-zero 64-bit PRUK ID in the P-TID field. On reception of the GPI, the PKMF uses Ks(_ext)_NAF as the PRUK.

If the PKMF support the PC4a interface to the HSS of the UE, then the PKMF shall request an Authentication Vector (AV) for the UE. On receiving the AV, the PKMF locally forms the GPI including a non-zero 64-bit PRUK ID in the P-TID field and sets PRUK as above.

4c. The PKMF generate a random number as the K_D Freshness Parameter. The PKMF uses the PRUK to calculate K_D with the Relay Service Code, Nonce_1 and K_D Freshness Parameter as inputs. The PKMF shall send the Remote UE Identity, K_D, K_D Freshness Parameter and the GPI if used to calculate a fresh PRUK to the UE-to-network relay.

5a. Using the supplied K_D to protect the message, the UE-to-network relay sends a Direct Security Mode Command message to the Remote UE (see 6.5.2.2). This message shall contain the K_D Freshness Parameter and the GPI if it received them from the PKMF.

5b. If the Remote UE receives a GPI, it calculates a new PRUK and associated PRUK ID (see step 4b above). The Remote UE derives K_D from its PRUK and the received K_D Freshness Parameter, Nonce_1 and the Relay Service Code (as described in Annex A.Y). It then processes the Direct Security Mode Command as described in 6.5.2.2. If this is successful, the Remote UE responds with a Direct Security Mode Complete message and the Remote UE and UE-to-network relay may start to exchange user data.

6.7.3.2.1.2 Re-synchronisation in GBA Push authentication

This subclause provides the flows when the UE discovers a synchronisation failure when processing the authentication challenge that was sent to it as part of the GPI (for details of synchronisation failures – see TS 33.102[42]). Synchronisation failures can happen in both the attachment flow (described in subclause 6.7.3.2.1.1) and the rekeying flow (described in subclause 6.7.3.1.3). The re-synchronisation flow is shown in figure 6.7.3.2.1.2-1, which only shows the contents of messages that are different for the synchronisation case.

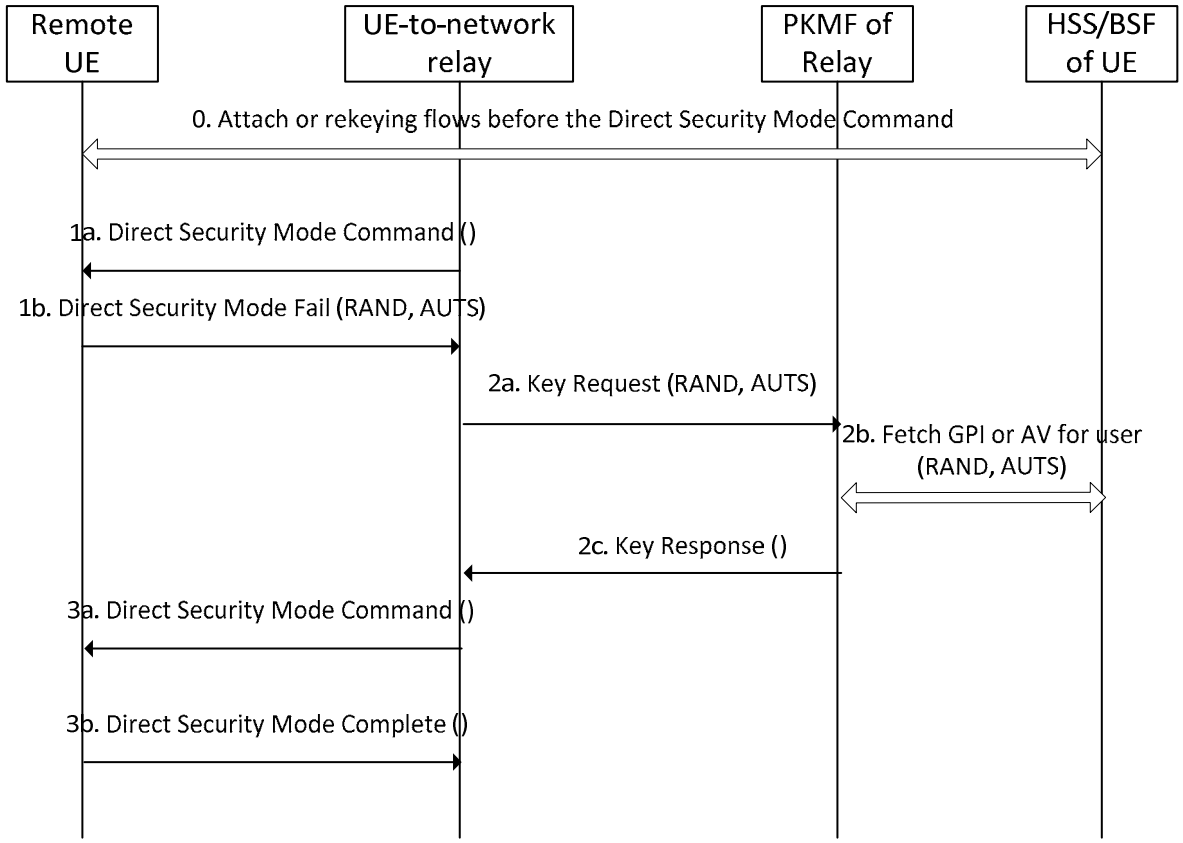

Figure 6. 7.3.2.1.2-1: Re-synchronisation flows

This steps are the identical to the Attach and rekeying flows up to the Direct Security Mode Command in these flows

1a. The Direct Security Mode Command contains the same parameters as the Direct Security Mode Command in the Attach or rekeying flows. On processing the GPI parameter contained in this message, the ME receives a synchronisation failure response for the USIM.

1b. The Remote UE send a Direct Security Mode Failure to the relay that contains the RAND and AUTS parameters

2a. The relay send a Key Request message to the PKMF of the Relay and includes the RAND and AUTS received from the Remote UE.

2b. If the PKMF supports the Zpn interface to the BSF of the UE, the PKMF shall request a GBA Push Info (GPI – see TS 33.223[38]) for the Remote UE from the BSF and include the RAND and AUTS parameters. When requesting the GPI, it includes the 64-bit PRUK ID in the P-TID field. On reception of the GPI, the PKMF uses Ks(_ext)_NAF as the PRUK.

If the PKMF support the PC4a interface to the HSS of the UE, then the PKMF shall request an Authentication Vector (AV) for the UE and include the RAND and AUTS parameters. On receiving the AV, the PKMF locally forms the GPI including the 64-bit PRUK ID in the P-TID field and sets PRUK as above.

2c, 3a and 3b. These messages are identical to the corresponding messages in the Attach and rekeying flows.

6.7.3.2.1.3 Rekeying procedures

Due to the asymmetric nature of the keying for relays, there are three rekeying cases that need to be considered. Firstly there is the rekeying that only changes $K_{D\text{-}Sess}$ but not K_D. Secondly there is the rekeying that is initiated by the Remote UE that changes K_D and finally rekeying that is initiated by the relay that changes K_D. Each of these is described in turn.

Rekeying without changing K_D happens exactly as described in subclause 6.5.5.3. The Remote UE includes its PRUK ID as the Long Term ID in the Direct Rekey Request messages, whereas the relay does not include a Long Term Key ID.

A rekeying that changes K_D that is triggered by the Remote UE is shown in figure 6.7.3.2.1.3-1. The steps follow subclause 6.5.5.3 and only message contents that are specific to this use are included

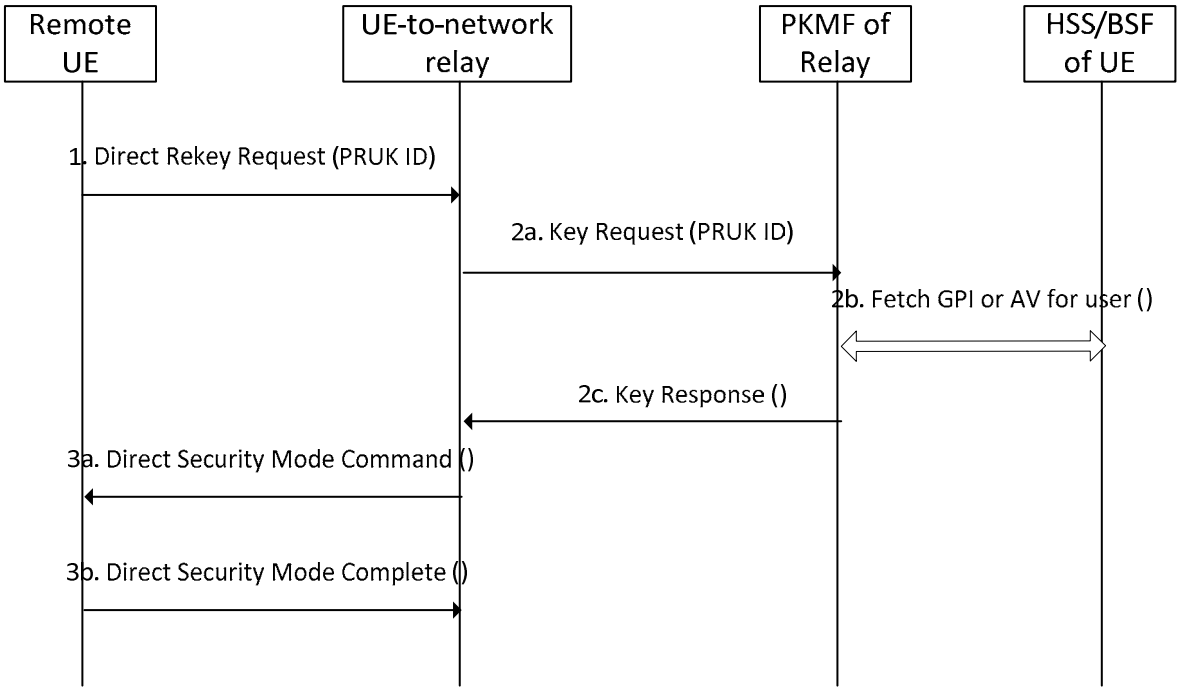

Figure 6.7.3.2.1.3-1: Rekeying K_D when rekeying was initiated by the Remote UE

1; The Remote UE send a Direct Rekey Request including its PRUK ID to the Relay.

2a, 2b, 2c, 3a and 3b. The content and handling of these messages are the same as messages in the Attach flow (subclause 6. 7.3.2.1.1) except the security contexts are handled as in subclause 6.5.5.3.

A rekeying that changes K_D that is initiated by the relay proceeds as follows. The relay sends a Direct Key Request that does not contain a Long Term ID. Instead of responding to this message, the Remote UE sends its own Direct Rekey Request and the procedure continues as for the rekeying that changes K_D that is triggered by the Remote UE.

6.7.3. 2.2 Messages between the Remote UE and ProSe Key Management Function

6.7.3. 2.2.1 General

Key Request and Response messages are exchanged between the Remote UE and ProSe Key Management Function. The Remote UE uses these messages to request a PRUK to use with relays. These messages are detailed in the following subclauses.

6.7.3. 2.2.2 Key Request and Key Response messages

The purpose of these messages is for the Remote UE to request the PRUK from the ProSe Key Management Function. The Remote UE knows from which ProSe Key Management Function(s) to get the needed PRUK(s) as the FQDN(s) of the PKMF(s) are either pre-provisioned or provided by the ProSe Function in the HPLMN of the Remote UE.

When sending a Key Request for a PRUK, the Remote UE shall include all the relevant details of all types of keys that the Remote UE is expecting to receive from the PKMF, e.g. any PGKs for one-to-many ProSe communication.

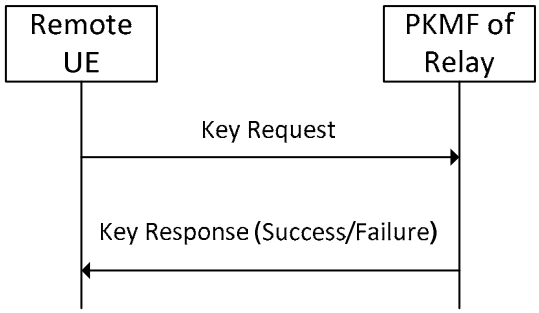

Figure 6.7.3. 2.2.2-1: Key Request/Response for Remote UE

The protection for the Key Request and Key Response message is described in subclause 6.7.3.4.

When sending a Key Request message to request the ProSe Key Management Function to send to either get a PRUK or ensure its PRUK is upto date, the Remote UE shall include the following information;

- An indication of whether the Remote UE wants to receive PRUKs from this PKMF.
- PRUK ID (if any) that the Remote UE has for this PKMF. If it has none it send an all zero PRUK ID.

The ProSe Key Management Function shall check that the Remote UE is authorised to receive PRUKs. This is done by using the Remote UE identity that is bound to the keys that established the TLS tunnel in which the message is sent. If the Remote UE is not authorised, then the ProSe Key Management Function responds with the appropriate error.

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

- If the Remote UE is authorised to receive PRUKs, then the message may include a PRUK and PRUK ID
- Otherwise, a status code to indicate why PRUKs will not be supplied.

If a PRUK and PRUK ID are included, the Remote UE shall store these and delete any previously stored ones that were obtained from this ProSe Key Management Function.

6.7.3. 2.3 Messages between the Relay and ProSe Key Management Function

6.7.3. 2.3.1 General

There are only Key Request/Response messages exchanged between the UE-to-network Relay and ProSe Key Management Function.

6.7.3. 2.3.2 Key Request and Key Response messages

The purpose of these messages is for the Relay to request the K_D from the ProSe Key Management Function.

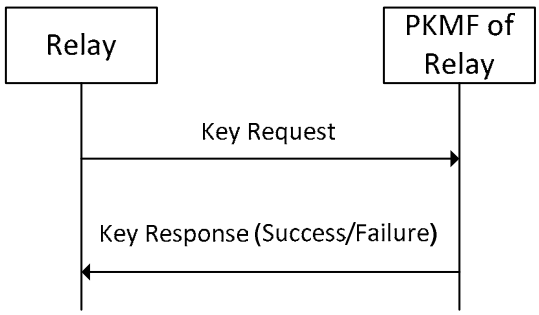

Figure 6.7.3. 2.3.2-1: Key Request/Response for Relay

The protection for the Key Request and Key Response message is described in subclause 6. 7.3.4.

When sending a Key Request message to request the ProSe Key Management Function to request a K_D, the UE-to-network relay shall include the following information;

- Either the PRUK ID or the IMSI that was provided in the Direct Communication Request message or Direct Rekey Request;
- Relay Service Code that the Remote UE requested to use;
- Nonce_1 that was sent from the Remote UE to the UE-to-network relay in either the Direct Commuications Request or Direct Rekey Request that triggered this Key Request; and
- RAND and AUTS, in the case of a synchronisation failure of the AV in the GPI.

The ProSe Key Management Function shall check that the Relay is authorised to serve the identified Remote UE for the supplied Relay Service Code. This is done by using the Relay's identity that is bound to the keys that established the TLS tunnel in which the message is sent. If the Relay is not authorised or the Remote UE can not be identified, then the ProSe Key Management Function responds with the appropriate error.

If the ProSe Key Management Function decides to provide a K_D to the UE-to-network relay, then it generates a random number that it sends as the K_D Freshness parameter to the UE-to-network relay. The ProSe Key Management Function also calculates the K_D (as described in Annex A.Y) from either the PRUK related to the supplied PRUK ID or the new PRUK if this is to be updated (see subclause 6.X.3.2.1.1). In addition to these parameter, the ProSe Key Management Function also provides a Remote UE Identity that the UE-to-network relay provides to the MME. The Remote UE Identity is either the IMSI, MSISDN or a 128-bit string.

NOTE: In general, IMSI should not be sent outside of the operator network in order to protect user privacy. The UE-to-Network Relay cannot be regarded as a network entity in the traditional sense e.g. as an eNB. On the other hand, the PKMF may have a sufficient level of trust in a UE-to-Network Relays to provide the IMSI. Instead of sending the IMSI, the PKMF can send a 128-bit string to the UE-to-Network Relay instead of the IMSI. The string should be such that the MME map the character string to a wanted Remote UE identity (e.g. IMSI) but that the UE-to-Network Relay cannot deduce the Remote UE identity. How this mapping is done in the MME has not be specified by SA3. The mapping information needs to be provisioned into the MME.

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

- For a successful case,
  - a K_D;
  - K_D Freshness parameter;
  - an optional GPI; and
  - Remote UE Identity.
- Otherwise, a status code to indicate why K_D will not be supplied.

6.7.3.3 Protection of traffic between Remote UE or Relay and ProSe Function

In order to protect the messages between the Remote UE/UE-to-network Relay and ProSe Function, the Remote UE/UE-to-network relay 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.

6.7.3.4 Protection of traffic between Remote UE or Relay and ProSe Key Management Function

In order to protect the messages between the Remote UE/ UE-to-network Relay and ProSe Key Management Function, the Remote UE/UE-to-network Relay 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.

6.7.3.5 Protection of traffic between Remote UE and Relay

The signalling and user plane traffic sent between the Remote UE and UE-to-network relay are protected as described in subclause 6.5.6

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].

A.2 Calculation of the MIC value

When calculating a MIC using the Discovery Key for open discovery or the DUIK for restricted discovery, 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 for open discovery or ProSe Restricted Code for restricted discovery.
- 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, DUIK, 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 and $K_{D\text{-}sess}$

When calculating a PIK or PEK from PTK or $K_{D\text{-}sess}$, 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 if PEK is being derived or 0x01 if PIK is being derived
- L0 = length of P0 (i.e. 0x00 0x01)
- P1 = algorithm identity
- L1 = length of algorithm identity (i.e. 0x00 0x01)

NOTE: Void.

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

The input key shall be the 256-bit PTK or the 256-bit $K_{D\text{-}sess}$.

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.

A.5 Calculation of scrambling bits for discovery

When calculating the time-hash-bitsequence for discovery, 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 = 0x4C
- P0 = UTC-based counter for scrambling associated with the discovery slot – see subclause 6.1.3.4.3.5.
- L2 = length of above (i.e. 0x00 0x04).

The input key shall be the 256-bit DUSK.

The time-hash-bitsequence keystream is set to the 216 least significant bits of the output of the KDF.

A.6 Calculation of message-specific confidentiality keystream for discovery

When calculating the message-specific confidentiality keystream for discovery, 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 = 0x4D
- P0 = UTC-based counter associated with the discovery slot – see subclause 6.1.3.4.3.6.
- L2 = length of above (i.e. 0x00 0x04).
- P1 = (Key_calc_mask AND (Message || MIC)) – see subclause 6.1.3.4.3.6
- L2 = length of above (i.e. 0x00 0x1B).

The input key shall be the 256-bit DUCK.

The message-specific confidentiality keystream is set to the 184 least significant bits of the output of the KDF.

A.7 Calculation of K_D for UE-to-network relays

When calculating K_D from PRUK, 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 = 0x48
- P0 = Relay Service Code
- L0 = length of Relay Service Code (i.e. 0x00 0x03)
- P1 = Nonce_1
- L1 = length of Nonce_1 (i.e. 0x00 0x10)
- P2 = K_D Freshness Parameter
- L2 = length of Nonce_2 (i.e. 0x00 0x10)

The input key shall be the 256-bit PRUK.

A.8 Calculation of discovery keys from PSDK

When calculating a DUSK, DUCK or DUIK from the PSDK, 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 = 0x4F
- P0 = 0x00 if DUSK is being derived, 0x01 if DUCK is being derived, or 0x02 if DUIK is being derived
- L0 = length of P0 (i.e. 0x00 0x01)
- P1 = algorithm identity
- L1 = length of algorithm identity (i.e. 0x00 0x01)

The algorithm identity shall be set to 0x00. Later releases may define other values.

The input key shall be the 256-bit PSDK.

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.

A.9 Calculation of $K_{D\text{-}sess}$ from K_D

When calculating $K_{D\text{-}sess}$ from K_D, 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 = 0x4E
- P0 = Nonce_1
- L0 = length of Nonce_1 (i.e. 0x00 0x10)
- P1 = Nonce_2
- L1 = length of Nonce_2 (i.e. 0x00 0x10)

The input key shall be the 256-bit K_D.

Annex B (informative): Void

Annex C (informative):
Void

Annex D (informative): Void

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>

<xs:complexType name="IMSI-info">
    <xs:sequence>
      <xs:element name="MCC" type="xs:integer"/>
      <xs:element name="MNC" type="xs:integer"/>
      <xs:element name="MSIN" 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="MSISDN-info">
    <xs:sequence>
      <xs:element name="CC" type="xs:integer"/>
      <xs:element name="NDC" type="xs:integer"/>
      <xs:element name="SN" 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="PRUK-info">
    <xs:sequence>
      <xs:element name="PRUKID" type="xs:hexBinary" />
      <xs:element name="PRUK" 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="KDReq-info">
    <xs:sequence>
      <xs:element name="PRUKID" type="xs:hexBinary" minOccurs="0" />
      <xs:element name="IMSI" type="IMSI-info" minOccurs="0" />
      <xs:element name="RelayServiceCode" type="xs:string" />
      <xs:element name="Nonce1" type="xs:hexBinary" />
      <xs:element name="RAND" type="xs:hexBinary" minOccurs="0" />
      <xs:element name="AUTS" type="xs:hexBinary" minOccurs="0" />
      <xs:element name="anyExt" type="anyExtType" minOccurs="0"/>
      <xs:any namespace="##any" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:anyAttribute namespace="##any" processContents="lax"/>
  </xs:complexType>

  <xs:complexType name="KDResp-info">
    <xs:sequence>
      <xs:element name="KeyKD" type="xs:hexBinary" />
      <xs:element name="KDFreshnessParameter" type="xs:hexBinary" />
      <xs:element name="GPI" type="xs:hexBinary" minOccurs="0" />
      <xs:element name="RemoteUEIMSI" type="IMSI-info" minOccurs="0" />
      <xs:element name="RemoteUEMSISDN" type="MSISDN-info" minOccurs="0" />
      <xs:element name="RemoteUEOtherID" type="xs:hexBinary" 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>

  <xs:complexType name="RelayDiscoveryKey-Request">
    <xs:sequence>
      <xs:element name="RelayServiceCode" type="xs:string"/>
      <xs:element name="PSDKId" 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="Relay-Reject">
    <xs:sequence>
      <xs:element name="RelayServiceCode" type="xs:string"/>
      <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="Relay-Response">
    <xs:sequence>
      <xs:element name="RelayServiceCode" type="xs:string"/>
      <xs:element name="KeyTypeID" type="xs:string"/>
      <xs:element name="ProtectionProfile" type="xs:hexBinary"/>
      <xs:element name="EncryptedBitMask" type="xs:hexBinary" 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>

  <xs:complexType name="GroupMemberDiscoveryKey-Request">
    <xs:sequence>
      <xs:element name="DiscoveryGroupID" type="xs:string"/>
      <xs:element name="PSDKId" 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="GroupMember-Reject">
    <xs:sequence>
      <xs:element name="DiscoveryGroupID" type="xs:string"/>
      <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="GroupMember-Response">
    <xs:sequence>
      <xs:element name="DiscoveryGroupID" type="xs:string"/>
      <xs:element name="KeyTypeID" type="xs:string"/>
      <xs:element name="ProtectionProfile" type="xs:hexBinary"/>
      <xs:element name="EncryptedBitMask" type="xs:hexBinary" 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>

  <!-- 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"/>
      <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="PSDiscoverySecurityCapabilities" type="xs:hexBinary" minOccurs="0"
maxOccurs="1"/>
      <xs:element name="RelayDiscoveryKeyRequest" type="RelayDiscoveryKey-Request" minOccurs="0"
maxOccurs="unbounded"/>
      <xs:element name="RelayDiscoveryKeyStop" type="xs:integer" minOccurs="0"
maxOccurs="unbounded"/>
      <xs:element name="GroupMemberDiscoveryKeyRequest" type="GroupMemberDiscoveryKey-Request"
minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="GroupMemberDiscoveryKeyStop" type="xs:integer" minOccurs="0"
maxOccurs="unbounded"/>
      <xs:element name="PRUKID" type="xs:hexBinary" minOccurs="0" />
      <xs:element name="KDReqinfo" type="KDReq-info" minOccurs="0"/>
      <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="anyExt" type="anyExtType" minOccurs="0"/>
```

```
    </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="RelayNotSupported" type="Relay-Reject" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="RelayResponse" type="Relay-Response" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="GroupMemberNotSupported" type="GroupMember-Reject" minOccurs="0"
maxOccurs="unbounded"/>
      <xs:element name="GroupMemberResponse" type="GroupMember-Response" minOccurs="0"
maxOccurs="unbounded"/>
      <xs:element name="Key-info" type="PMK-info" minOccurs="0"/>
      <xs:element name="CurrentTime" type="xs:dateTime" minOccurs="0" />
      <xs:element name="MaxOffset" type="xs:integer" minOccurs="0" />
      <xs:element name="PRUKinfo" type="PRUK-info" minOccurs="0"/>
      <xs:element name="PRUKError" type="xs:integer" minOccurs="0"/>
      <xs:element name="KDRespinfo" type="KDResp-info" minOccurs="0"/>
      <xs:element name="KDError" type="xs:integer" 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 one <PRUKId> element containing the parameter defined in subclause E.5.2.2.10;
5) zero or one <KDRequest> element, each of which consists of:
    a) zero or one <PRUKId> element containing the parameter defined in subclause E.5.2.2.10;
    b) zero or one <IMSI> element containing the parameter defined in subclause E.5.2.2.12;
    c) one <RelayServiceCode> element containing the parameter defined in subclause E.5.2.2.13;
    d) one <Nonce1> element containing the parameter defined in subclause E.5.2.2.15;
    e) zero or one <RAND> element containing the parameter defined in subclause E.5.2.2.16;
    f) zero or one <AUTS> element containing the parameter defined in subclause E.5.2.2.17;
    g) zero or one <anyExt> element containing elements defined in future releases;
    h) zero, one or more elements from other namespaces defined in future releases; and
    i) zero, one or more attributes defined in future releases;
6) zero or more <GroupKeyStop> element containing the parameter defined in subclause E.5.2.2.5
7) zero or one <PSDiscoverySecurityCapabilities> element contains the parameter defined in subclause E.5.2.2.22.
8) zero or more <RelayDiscoveryKeyRequest> element, each of which consists of:
    a) a <RelayServiceCode> element containing the parameter defined in subclause E.5.2.2.23;
    b) one or more <PSDKId> element containing the parameter defined in subclause E.5.2.2.24; c) zero, one or more elements defined in future releases; and
    d) zero, one or more attributes defined in future releases;
9) zero or more <RelayDiscoveryKeyStop> element containing the parameter defined in subclause E.5.2.2.23.
10) zero or more <GroupMemberDiscoveryKeyRequest> element, each of which consists of:
    a) a <DiscoveryGroupID> element containing the parameter defined in subclause E.5.2.2.25;
    b) one or more <PSDKId> element containing the parameter defined in subclause E.5.2.2.26;
    c) zero, one or more elements defined in future releases; and
    d) zero, one or more attributes defined in future releases;

11) zero or more <GroupMemberDiscoveryKeyStop> element containing the parameter defined in subclause E.5.2.2.25. 10.

12) zero or one <anyExt> element containing elements defined in future releases;

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

14) 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 <PRUKinfo> element, each of which consists of:

   a) a <PRUKID> element containing the parameter defined in subclause E.5.2.2.10;

   b) a <PRUK> element containing the parameter defined in subclause E.5.2.2.11;

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

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

6) zero or one <PRUKError> element element containing the parameter defined in subclause E.5.2.2.5;

7) zero or one <KDResponse> element, each of which consists of:

   a) a <KeyKD> element containing the parameter defined in subclause E.5.2.2.18;

   b) a <KDFrehsnessParameter> element containing the parameter defined in subclause E.5.2.2.19;

   c) zero or one <GPI> element containing the parameter defined in subclause E.5.2.2.20;

   d) zero or one <RemoteUEIMSI> element containing the parameter defined in subclause E.5.2.2.12;

e) zero or one <RemoteUEMSISDN> element containing the parameter defined in subclause E.5.2.2.14;

f) zero or one <RermoteUEOtherID> element containing the parameter defined in subclause E.5.2.2.21;

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

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

8) zero or one <KDError> element element containing the parameter defined in subclause E.5.2.2.5;

9) zero or more <RelayNotSupported> element, each of which consists of :

a) a <RelayServiceCode> element containing the parameter defined in subclause E.5.2.2.23;

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;

10) zero or more <RelayResponse> element, each of which consists of:

a) a <RelayServiceCode> element containing the parameter defined in subclause E.5.2.2.23;

b) a <KeyTypeID> element containing a parameter defined in subclause E.5.2.2.28.

c) a <ProtectionProfile> element containing the parameter defined in subclause E.5.2.2.26;

d) zero or one <EncryptedBitMask> element containing the parameter defined in subclause E.5.2.2.27;

e) zero or one <anyExt> element containing elements defined in future releases;

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

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

11) zero or more <GroupMemberDiscoveryNotSupported> element, each of which consists of :

a) a <DiscoveryGroupID> element containing the parameter defined in subclause E.5.2.2.25;

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;

12) zero or more <GroupMemberDiscoveryResponse> element, each of which consists of:

a) a <DiscoveryGroupID> element containing the parameter defined in subclause E.5.2.2.25;

b) a <KeyTypeID> element containing a parameter defined in subclause E.5.2.2.28.

c) a <ProtectionProfile> element containing the parameter defined in subclause E.5.2.2.26;

d) zero or one <EncryptedBitMask> element containing the parameter defined in subclause E.5.2.2.27;

e) zero or one <anyExt> element containing elements defined in future releases;

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

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

13) zero or one <CurrentTime> element contains the parameter defined in subclause E.5.2.2.29.

14) zero or one <MaxOffset> element contains the parameter defined in subclause E.5.2.2.30.

15) zero or one <anyExt> element containing elements defined in future releases;

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

17)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 for this one-to-many communication group or discovery (relay or group member).

2 The ProSe Key Management Function does not supply keys for this one-to-many communication group or discovery (relay or group member).

3 UE is not authorised to receive keys for this one-to-many communication group or discovery (relay or group member).

4 UE requested to stop receiving PGKs for this group or PSDKs for this discovery (relay or group member).

5 UE not authorised to receive PRUKs from this PKMF.

6 PRUK ID or IMSI not recognised.

7 UE-to-network relay not authorised to serve this UE.

8-255 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.

E.5.2.2.10 PRUK ID

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

E.5.2.2.11 PRUK

This parameter is a key that is shared by the Remote UE and PKMF and is used to generate keys for protecting Remote UE to UE-to-network relay connections. It is a 32 octet long binary parameter.

E.5.2.2.12 IMSI

This parameter is used to carry the Remote UE's IMSI. The coding of IMSI is defined in 3GPP TS 23.003 [39].

E.5.2.2.13 Relay Service Code

This parameter is used to indicate the Relay Service Code for which the UE is requesting service. It is a 24-bit long string.

E.5.2.2.14 MSISDN

This parameter is used to carry the Remote UE's MSISDN. The coding of MSISDN is defined in 3GPP TS 23.003 [39].

E.5.2.2.15 Nonce 1

This random number is generetaed by the Remote UE to ensure fresh keys. It is a 16 octet long binary parameter.

E.5.2.2.16 RAND

This parameter is the RAND from the authemtication challenge used in the GBA PUSH authentication. It is a 16 octet long binary parameter.

E.5.2.2.17 AUTS

This parameter is the AUTS that is generated in a synchronisation failure. It is a 14 octet long binary parameter.

E.5.2.2.18 Key K_D

This parameter is a key that is shared by the Remote UE and UE-to-netwrok relay and is used to generate keys for protecting Remote UE to UE-to-network relay connections. It is a 32 octet long binary parameter.

E.5.2.2.19 K_D Freshness parameter

This random number is generetaed by the PKMF to ensure the K_D that it derives is fresh. It is a 16 octet long binary parameter.

E.5.2.2.20 GPI

This is the GPI parameter that is sent to the UE as part of GBA PUSH proecdures (see TS 33.223[38]). It is a binary parameter.

E.5.2.2.21 Remote UE other identity

This parameter is a permanent identity of the Remote UE and is used when the KMF does not supply an IMSI of the Remote UE (e.g. the PKMF considers it a privacy issue sending IMSI to non-network elements). It is a 16 octet long binary parameter.

E.5.2.2.22 Public Safety Discovery Security Capabilities

This parameter is used to indicate the UE's security capabilities for protecting Public Safety Discovery messages. It is a binary parameter of length 1 octet that is encoded as shown in table E.5.2.2.22-1.

Table E.5.2.2.22-1: Public Safety Discovery Security Capabilities IE

Scrambling protection bit (octet 1, bit 8) 0 Scrambling is not supported 1 Scrambling is supported Bits 1- 7 of octet 1 are spare and shall be coded as zero.

E.5.2.2.23 Relay Service Code

This parameter is used to indicate the Relay Service Code for which the UE is requesting keys. It is a 24-bit long string.

E.5.2.2.24 PSDK ID

This parameter is used to indicate the PSDK IDs for a particular Relay Service Code or Discovery Group ID. It is an integer in the 0-255 range.

E.5.2.2.25 Discovery Group ID

This parameter is used to indicate the Discovery Group ID for which the UE is requesting keys. It is a 24-bit long string.

E.5.2.2.26 Protection Profile

This parameter is used to indicate which out of scrambling, integrity protection and message-specific confidentiality will be applied to protect the discovery. It is a binary parameter of length 1 octet that is encoded as shown in table E.5.2.2.26-1.

Table E.5.2.2.26-1: Selected security algorithm information element

Scrambling protection bit (octet 1, bit 8) 0 Scrambling is not used for this discovery 1 Scrambling is used for this discovery Integrity protection bit (octet 1, bit 7) 0 Integrity protection is not used for this discovery 1 Integrity protection is used for this discovery Message specific confidentiality bit (octet 1, bit 6) 0 Message specific confidentiality is not used for this discovery 1 Message specific confidentiality is used for this discovery Bits 1-5 of octet 1 are spare and shall be coded as zero.

E.5.2.2.27 Encrypted bit mask

This parameter is included if message-specific confidentilaity is to used for the discovery. It is used to indicate which bits of the message have message specific confidentiality applied to them. It is a binary parameter of length 23 octets.

E.5.2.2.28 Key Type ID

This parameter is used to indicate which discovery (relay or group member) the key belongs to when it arrives in a MIKEY message. It is a 24-bit long string parameter.

E.5.2.2.29 Current time

This parameter proviodes the UE with the time at the PKMF, and along with the Max Offset parameter, is used to ensure that the time the UE assoicates with the discovery slot is reasonably close to the real time. It is of type Date and Time.

E.5.2.2.30 Max Offset

This parameter indicates how close the time associated with the discovery slot needs to be to the time provided by the PKMF. Max Offset is measured in seconds. It is an integer in the 1-32 range.

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:

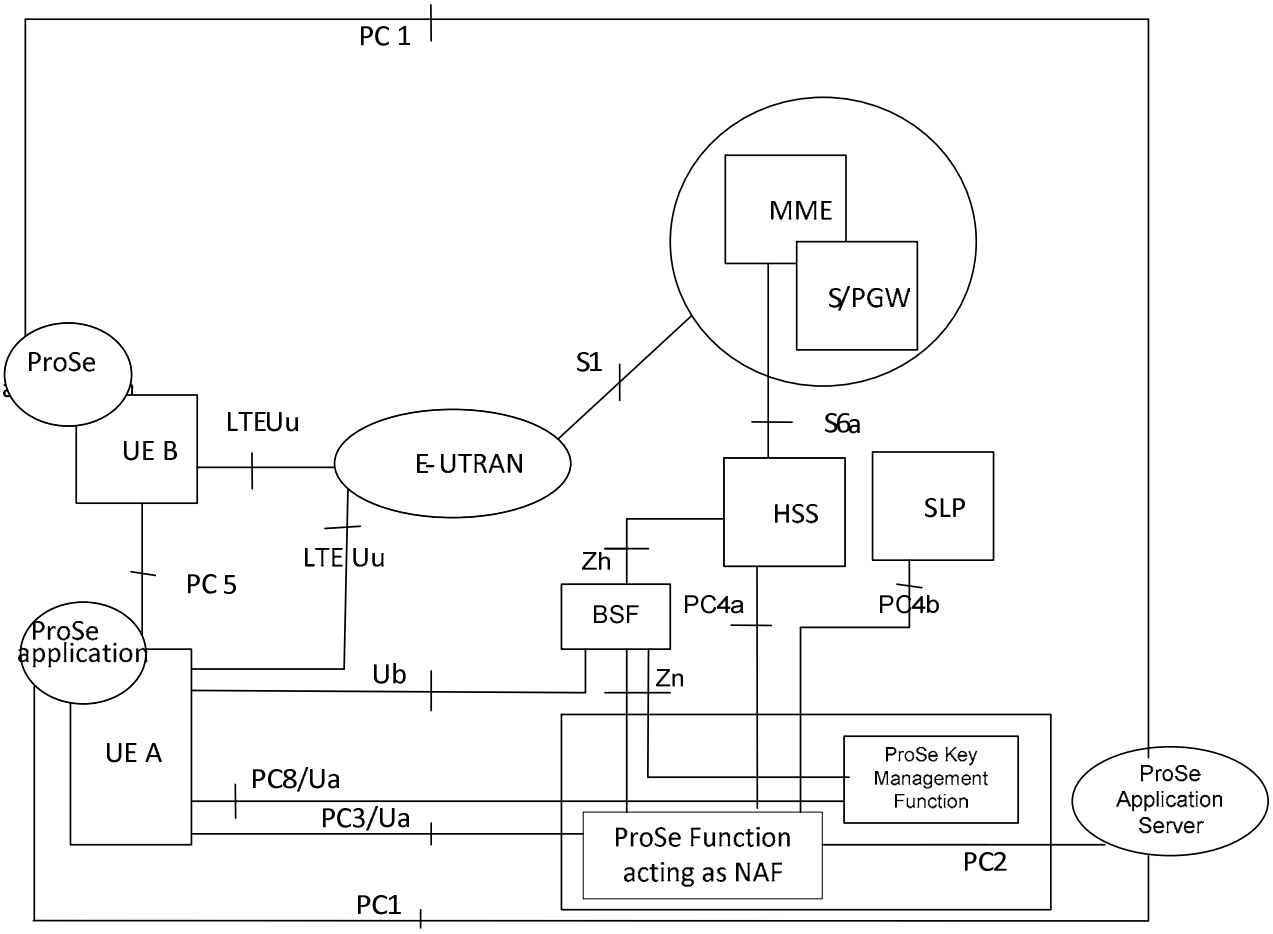

Figure F.2-1: Prose Function using standalone BSF.

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:

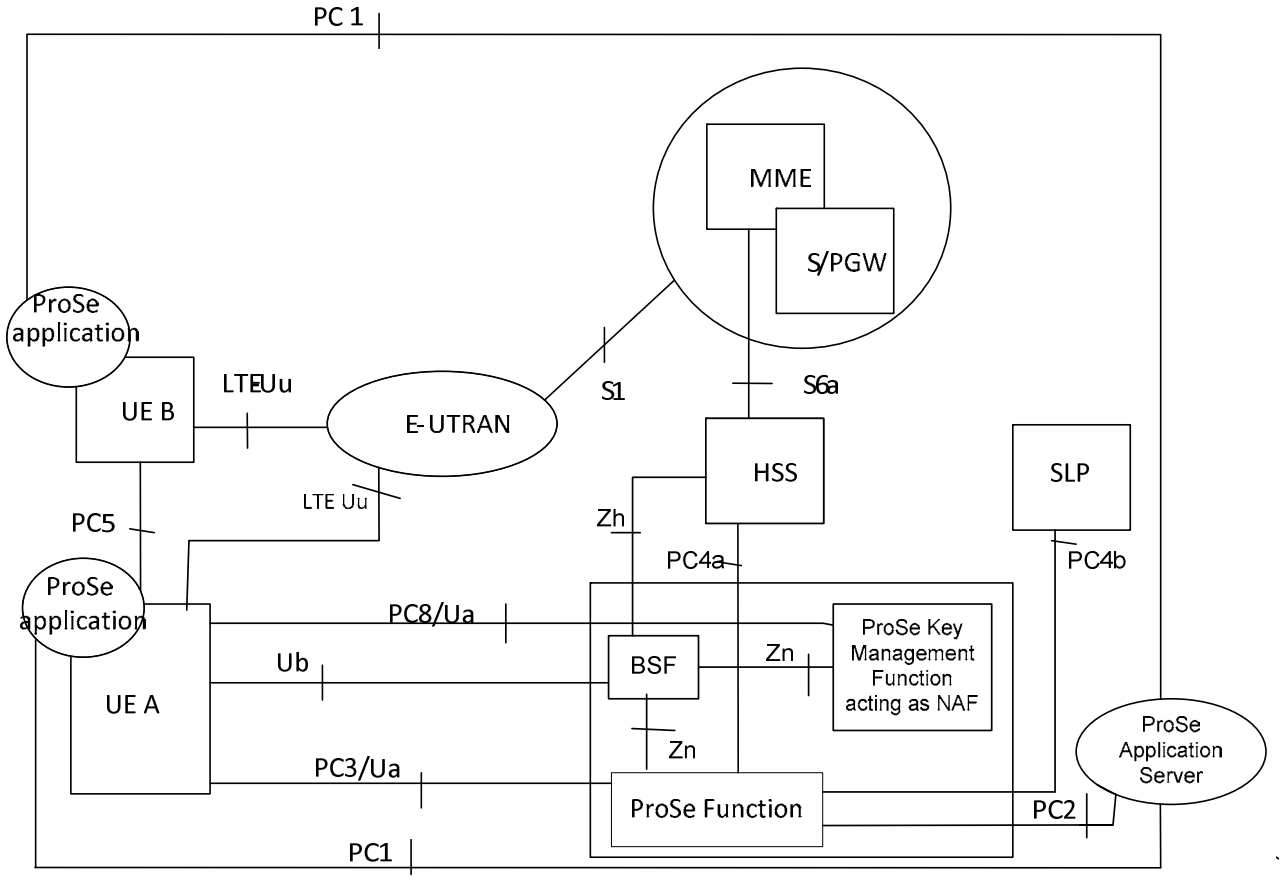

Figure F.3-1: BSF – Prose Function colocation

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:

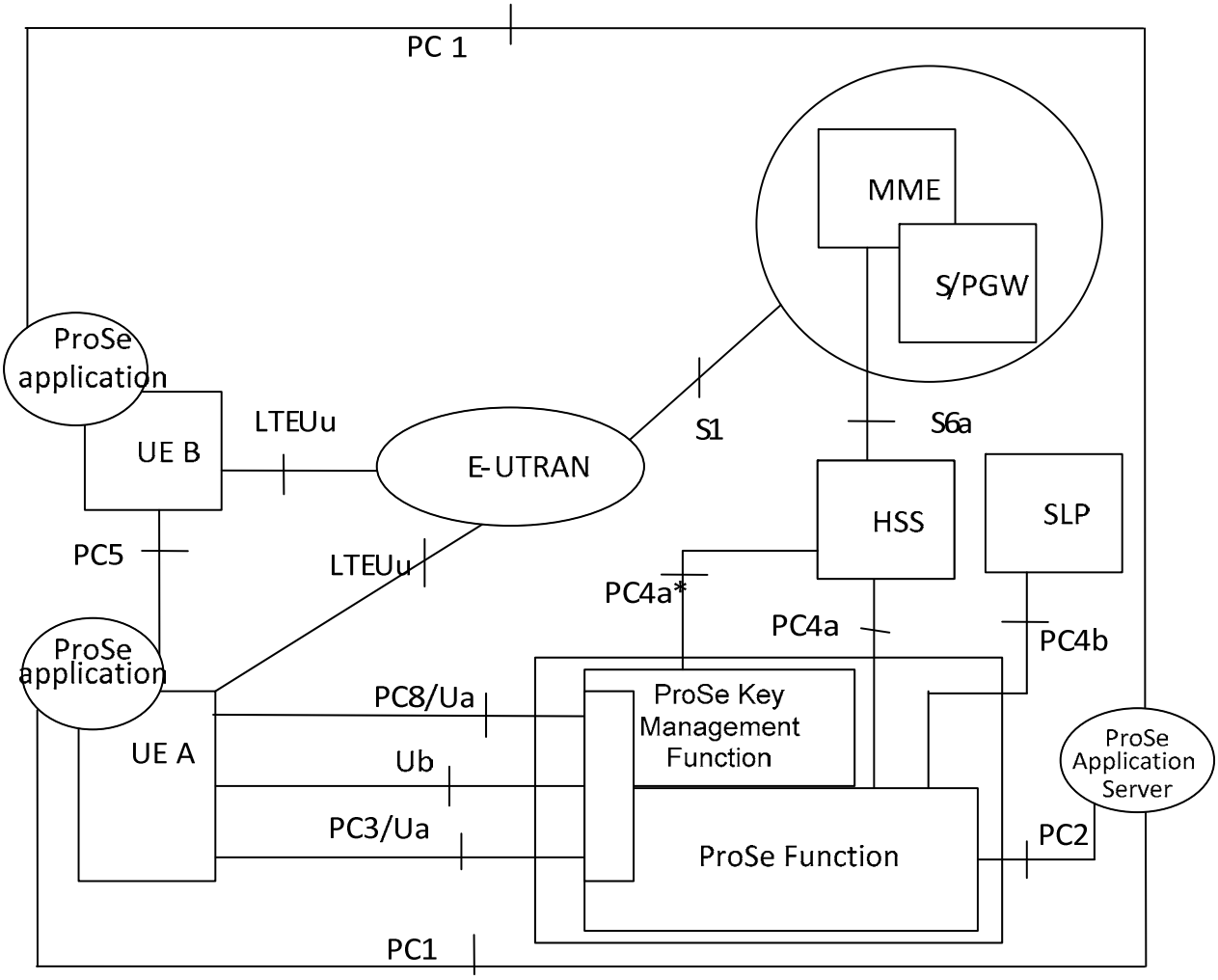

Figure F.4-1: Prose Function with bootstrapping entity

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): Protection of Restricted Discovery and Public Safety Discovery messages

G.1 General

Both Restricted Discovery messages and Public Saftey Discovery message are to be protected by some combination of scrambling, message-specific confidentiality and integrity protection (see subclause 6.1.3.4.3). The particular combinations that may be applied to each of these cases are not restricted by the standard, but are controlled by the ProSe Function(s) (in the case of Restricted Discovery) and the PKMF (in the case of Public Safety Discovery). This control is applied by the ProSe Function or PKMF signalling the chosen security measures for a particular discovery to the UE. The UE then applies the requested security to discovery messages that it sends or receives. This Annex provides information on the security provided by the combinations of security mechanisms.

G.2 Different combinations of security mechanisms

Table G.2-1 provides details of the security provided and sample use case(s) for the various combinations of security mechanisms that are possible for Restricted Discovery and Public Safety Discovery. This table is provided to help with the configuration in the ProSe Function or PKMF of the appropriate security mechanisms for a particular discovery and does not restrict the combination of security mechansisms that the UE supports for either Restricted or Public Safety Discovery (for details on UE processing of discovery messages, see subclauses 6.1.3.4.3.2 and 6.1.3.4.3.3).

Table G.2-1: Impact and example use case for the various combinations of security mechanisms

Sending UE	Receiving Side	Security provided	Comments and/or example use case
DUIK	DUIK	Discovery message is only integrity protected.	Relay Discovery where the Relay has no requirement for tracking or confidentiality protection.
DUSK	DUSK	Hiding of the parts of the message that identify the user (e.g. ProSe Code) to prevent the user being discovered or tracked by unauthorised users. Note: only works if there is exactly one message to be protected with the DUSK (as then the scrambling provides integrity protection).	Restricted discovery where exactly one ProSe code is protected by thisDUSK.
DUSK and DUIK	DUSK	As above	As above
	DUSK and DUIK	Hiding of the parts of the message that identify the user (e.g. ProSe Code) to prevent the user being discovered or tracked by unauthorised users. When only one user uses the DUSK, it prevents tracking of the user. When multiple users are using the same DUSK, the XOR difference between the unprotected discovery messages will be preserved and would enable some tracking of a user. Integrity protection of the whole message. Confidentiality protection of the whole message when only one user uses the DUSK	Restricted Discovery when one user uses DUSK to send Code with Application Controlled Extension bits Restricted Discovery with several users sending unique codes and no Application Controlled Extension bits.
DUSK, DUIK and DUCK	DUSK	Hiding of the parts of the message that identify the user (e.g. ProSe Code) to prevent the user being discovered or tracked by unauthorised users. Note: only works if there is exactly one message to be protected with the DUSK (as then the scrambling provides integrity protection)	DUSK applies to exactly one full code and ciphered Application Controlled Extension bits included in the message but are not decryptable by a receiver that only gets DUSK, as it was not given DUCK.
	DUSK, DUIK and DUCK	Hiding of the parts of the message that identify the user (e.g. ProSe Code) to prevent the user being discovered or tracked by unauthorised users. When only one user uses the DUSK, it prevents tracking of the user. When multiple users are using the same DUSK, the XOR difference between the unprotected discovery messages will be preserved and would enable some tracking of a user. Integrity protection of the whole message, plus confidentiality protection of part of the message.	Restricted Discovery with more than one sender using the same DUSK and there are Application Controlled Extensions bits that are encrypted/decrypted using DUCK.
DUCK and DUIK	DUCK and DUIK	Integrity protection of the whole message and confidentiality protection of part of the message.	Group Member Discovery where the Discovery Group ID is sent in the clear and the rest of the message is protected by message-specific confidentiality.

The "Sending UE" column describes what keys need to be known to the sending UE in the case of Restricted Discovery or signalled for use by the PKMF in the case of Public Safety Discovery. The "Receiving side" column shows what keys need to be sent to the receiving UE for Restricted Discovery, except if ProSe Function MIC checking is required (when the DUIK is kept at the ProSe Function in the HPLMN of the receiving UE), or signalled for use by the PKMF. The "Security provided" column details the security that this combination of security mechanisms provide. The final column provides example use case(s) where this combination of security mechanisms can be used.

Annex H (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		
			058	1	VPLMN clarification	12.4.0	13.0.0
		SP-150299	066	-	UE obtaining the current time UTC from the BSF		
2015-09	SP-69	SP-150472	074	1	Matching scope section with content of spec - ProSe Rel-13	13.0.0	13.1.0
			076	-	Correction to terminology - ProSe Rel-13		
			081	-	Correction of the MIKEY message details for one-to-many comms		
			083	1	Correction of the XML for Key Requests and Response		
		SP-150471	085	2	Security requirements and security procedures for reference point PC2		
			086	2	Security flows for restricted discovery		
			088	1	Details of one-to-one commuincation security		
2015-12	SP-70	SP-150725	092	-	Modification on Model A security flows	13.1.0	13.2.0
			093	1	Modification on Model B security flows		
		SP-150726	097	1	Annex A update of MIC computation for discovery messages		
		SP-150725	098	1	Corrections to ProSe one-to-one communication		
			100	1	Miscellaneous editorials in 33.303		

			101	1	Details of the scrambling and message specific confidentiality calculations for restricted discovery		
			102	1	Details of the security provided for restricted discovery and public safety discovery		
			103	1	Align LCID allocation with RAN specification		
			104	1	Protecting Public Safety Discovery messages		
		SP-150726	106	2	Usage of AEAD_AES_128_GCM_12 in ProSe		
			108	-	Alignment with 23.303		
2016-03	SP-71	SP-160051	0089	4	Security for UE-to-network relay communications	13.2.0	13.3.0
2016-03	SP-71	SP-160051	0099	2	Protection for restricted discovery to reduce processing at the monitoring UE-side	13.2.0	13.3.0
2016-03	SP-71	SP-160051	0109	1	Clarification on authorisation check in the Public Safetry Discovery and correction to Figure 6.6.4.1-1	13.2.0	13.3.0
2016-03	SP-71	SP-160051	0110	1	Corrections to 6.1.3.4.2.2	13.2.0	13.3.0
2016-03	SP-71	SP-160051	0114	1	Configuration of not applying confidentiality protection in one-to-one communication	13.2.0	13.3.0
2016-03	SP-71	SP-160051	0115	1	Security Key Agreement for ProSe UE-2-UE Direct Communication	13.2.0	13.3.0
2016-03	SP-71	SP-160051	0117	1	Clarification in 6.1.3.3.1	13.2.0	13.3.0
2016-03	SP-71	SP-160051	0118	-	Modifications on the figure of flows for securing model B restricted discovery	13.2.0	13.3.0
2016-03	SP-71	SP-160051	0119	1	Security requirements of ProSe direct discovery	13.2.0	13.3.0
2016-03	SP-71	SP-160051	0121	-	Correction to fig F.4-1	13.2.0	13.3.0
2016-03	SP-71	SP-160051	0123	1	Protection of public safety restricted discovery	13.2.0	13.3.0
2016-03	SP-71	SP-160051	0124	-	Providing details of the replay protection and XML signalling for public safety discovery	13.2.0	13.3.0
2016-03	SP-71	SP-160051	0126	1	Specifying the derivation of KD-sess for ProSe one-to-one communications	13.2.0	13.3.0
2016-03	SP-71	SP-160051	0127	1	Removal of public safety media security from TS 33.303	13.2.0	13.3.0

Change history							
Date	**Meeting**	**TDoc**	**CR**	**Rev**	**Cat**	**Subject/Comment**	**New version**
2016-06	SA#72	SP-160451	0128	2	F	Corrections to TS33.303	13.4.0
2016-06	SA#72	SP-160451	0129	2	F	Mandatory confidentiality protection for ProSe Public Safety Discovery	13.4.0
2016-06	SA#72	SP-160451	0130	1	F	Adding a missing FC value to TS 33.303	13.4.0
2016-06	SA#72	SP-160451	0132	1	F	Editorial corrections to TS 33.303	13.4.0
2017-03	SA#75	-	-	-		Promotion to Release 14 without technical change	14.0.0
2017-06	SA#76	SP-170428	0134	1	A	Correcting the XML for the UE to PKMF interface	14.1.0
2018-06	-	-	-	-	-	Update to Rel-15 version (MCC)	**15.0.0**

History

Document history		
V15.0.0	July 2018	Publication