|  |  |
| --- | --- |
| 3GPP TR 33.835 V16.1.0 (2020-07) | |
| Technical Report | |
| 3rd Generation Partnership Project;  Technical Specification Group Services and System Aspects;  Study on authentication and key management for applications  based on 3GPP credential in 5G  (Release 16) | |
|  | |
| *5G-logo_175px* | 3GPP-logo_web |
|  | |
| The present document has been developed within the 3rd Generation Partnership Project (3GPP TM) and may be further elaborated for the purposes of 3GPP. The present document has not been subject to any approval process by the 3GPPOrganizational Partners and shall not be implemented. This Specification is provided for future development work within 3GPPonly. The Organizational Partners accept no liability for any use of this Specification. Specifications and Reports for implementation of the 3GPP TM system should be obtained via the 3GPP Organizational Partners' Publications Offices. | |

|  |
| --- |
| Keywords  5G, security,credentials, AKMA |
| ***3GPP***  Postal address  3GPP support office address  650 Route des Lucioles - Sophia Antipolis  Valbonne - FRANCE  Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16  Internet  http://www.3gpp.org |
| ***Copyright Notification***  No part may be reproduced except as authorized by written permission. The copyright and the foregoing restriction extend to reproduction in all media.  © 2020, 3GPP Organizational Partners (ARIB, ATIS, CCSA, ETSI, TSDSI, TTA, TTC).  All rights reserved.  UMTS™ is a Trade Mark of ETSI registered for the benefit of its members  3GPP™ is a Trade Mark of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners LTE™ is a Trade Mark of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners  GSM® and the GSM logo are registered and owned by the GSM Association |

Contents

Foreword 8

1 Scope 10

2 References 10

3 Definitions of terms, symbols and abbreviations 11

3.1 Terms 11

3.2 Symbols 11

3.3 Abbreviations 11

4 Key Issues 12

4.1 Key Issue#1: Security Anchor 12

4.1.1 Issue detail 12

4.1.2 Security Threat 14

4.1.3 Potential architectural requirement 14

4.2 Key Issue #2: Transport independent procedure definition 14

4.2.1 Issue details 14

4.2.2 Security Threats 15

4.2.3 Potential architecture requirements 15

4.3 Key Issue #3: Mutual authenticationbetween UE and anchor function 15

4.3.1 Issue details 15

4.3.2 Security Threats 15

4.3.3 Potential security requirements 15

4.4 Key Issue #4: Authentication framework 16

4.4.1 Issue details 16

4.4.2 Security Threats 16

4.4.3 Potential security requirements 16

4.5 Key Issue #5: User privacy 16

4.5.1 Issue details 16

4.5.2 Security Threats 16

4.5.3 Potential security requirements 17

4.6 Key Issue #6: Secure communication between UE and application server 17

4.6.1 Issue details 17

4.6.2 Security threats 17

4.6.3 Potential security requirements 17

4.7 Key Issue #7: Protecting subscriber's personal information in control and data traffic 17

4.7.1 Issue details 17

4.7.2 Security Threats 18

4.7.3 Potential security requirements 18

4.8 Key Issue #8: Protection of AKMA architecture interfaces 18

4.8.1 Issue details 18

4.8.2 Security Threats 18

4.8.3 Potential security requirements 18

4.9 Key Issue #9: Key separation for AKMA AFs 18

4.9.1 Issue details 18

4.9.2 Security Threats 18

4.9.3 Potential security requirements 18

4.10 Key Issue #10: Compliance with local rules and regulations 19

4.10.1 Issue details 19

4.10.2 Security Threats 19

4.10.3 Potential security requirements 19

4.11 Key Issue #11: Generic battery efficient end-to-end security 19

4.11.1 Issue details 19

4.11.2 Security threats 19

4.11.3 Potential security requirements 19

4.12 Key Issue #12: Key lifetimes 19

4.12.1 Issue details 19

4.12.2 Security Threats 19

4.12.3 Potential security requirements 20

4.13 Key Issue #13: API for AKMA keys in UE 20

4.13.1 Issue details 20

4.13.2 Security Threats 21

4.13.3 Potential security requirements 21

4.14 Key Issue #14: Key revocation 21

4.14.1 Issue details 21

4.14.2 Security Threats 21

4.14.3 Potential security requirements 21

4.15 Key Issue #15: Synchronization of keys when using established keys 21

4.15.1 Issue details 21

4.15.2 Security Threats 22

4.15.3 Potential security requirements 22

4.16 Key Issue #16: Application key freshness of AKMA 22

4.16.1 Issue details 22

4.16.2 Security Threats 22

4.16.3 Potential security requirements 22

4.17 Key Issue #17: AKMA push 22

4.17.1 Issue details 22

4.17.2 Security Threats 23

4.17.3 Potential security requirements 23

5 Candidate Solutions 23

5.1 Solution #1: Introducing third party key to AKMA 23

5.1.1 Introduction 23

5.1.2 Solution details 23

5.1.3 Evaluation 24

5.2 Solution #2: Access independent architecture solution for AKMA 25

5.2.1 Introduction 25

5.2.2 Solution details 25

5.2.2.1 Architecture and reference points 25

5.2.2.2 Procedures 26

5.2.2.2.1 Initiation 26

5.2.2.2.2 Authentication 26

5.2.2.2.3 Usage 27

5.2.3 Evaluation 27

5.3 Solution #3: Architecture solution for AKMA with standalone anchor 28

5.3.1 Introduction 28

5.3.2 Solution details 28

5.3.2.1 Architecture and reference points 28

5.3.2.2 Procedures 29

5.3.2.2.1 Initiation 29

5.3.2.2.2 Authentication 29

5.3.2.2.3 Usage 30

5.3.3 Evaluation 31

5.4 Solution #4: Bootstrapping authentication of AKMA 31

5.4.1 Introduction 31

5.4.2 Solution details 31

5.4.2.0 General 31

5.4.2.1 Potential Authentication procedure for 5G AKA 31

5.4.2.2 Potential Authentication procedure for EAP-AKA' 34

5.4.3 Evaluation 35

5.5 Solution #5: Transport independent procedure using existing protocols by applying OneM2M protocol binding mechanism 35

5.5.1 Introduction 35

5.5.2 Solution details 35

5.5.3 Evaluation 36

5.6 Solution #6: Transport independent procedure using existing protocols by introducing a protocol transfer gateway 36

5.6.1 Introduction 36

5.6.2 Solution details 37

5.6.2.1 Architecture reference model 37

5.6.2.1.1 Entities 37

5.6.2.1.2 Service based interfaces 37

5.6.2.2 Procedures 38

5.6.3 Evaluation 39

5.7 Solution #7: UE implementation scheme- AKMA framework and application on modem 39

5.7.1 Introduction 39

5.7.2 Solution details 39

5.7.3 Evaluation 39

5.8 Solution #8: UE implementation scheme- AKMA framework on UICC and application on modem 40

5.8.1 Introduction 40

5.8.2 Solution details 40

5.8.3 Evaluation 40

5.9 Solution #9: UE implementation scheme- Application Processor (AP) scheme with AKMA framework on modem 40

5.9.1 Introduction 40

5.9.2 Solution details 40

5.9.3 Evaluation 41

5.10 Solution #10: UE implementation scheme- Application Processor (AP) scheme with AKMA framework on UICC 41

5.10.1 Introduction 41

5.10.2 Solution details 41

5.10.3 Evaluation 42

5.11 Solution #11: UE implementation scheme- AKMA framework implemented on Secure Element (SE) 42

5.11.1 Introduction 42

5.11.2 Solution details 42

5.11.3 Evaluation 42

5.12 Solution #12: UE implementation scheme- AKMA framework implemented on application processor's OS 42

5.12.1 Introduction 42

5.12.2 Solution details 43

5.12.3 Evaluation 43

5.13 Solution #13: AKMA authentication via the control plane 43

5.13.1 Introduction 43

5.13.2 Solution details 44

5.13.2.1 Architecture and reference points 44

5.13.2.2 Procedures 44

5.13.2.2.1 Initiation 44

5.13.2.2.2 AKMA authentication with EAP-AKA' 45

5.13.2.2.3 AKMA authentication with 5G AKA 46

5.13.2.2.4 Usage 47

5.13.3 Evaluation 47

5.14 Solution #14: Key revocation 48

5.14.1 Introduction 48

5.14.2 Solution details 48

5.14.2.1 Revocation in Application function 48

5.14.2.2 Revocation in UE 48

5.15 Solution #15: Implicit bootstrapping 49

5.15.1 Introduction 49

5.15.2 Solution details 51

5.15.2.1 Authentication using EAP-AKA' 51

5.15.2.2 Authentication using 5G AKA 52

5.15.2.3 AKMA key refresh 52

5.15.3 Evaluation 53

5.16 Solution #16: Use of KSEAF as root key for KAKMA 54

5.16.1 Introduction 54

5.16.2 Solution details 54

5.16.2.1 AKMA Key Repository Service 54

5.16.2.1.1 AKMA Key Repository Service Serving Network Architecture Option 54

5.16.2.1.2 AKMA Key Repository Service Home Network Architecture Option 55

5.16.2.2 AKMA Established Key Use Procedure 56

5.16.2.2.1 Procedure 56

5.16.3 Evaluation 57

5.17 Solution #17: Efficient key derivation for end-to-end security 58

5.17.1 Introduction 58

5.17.2 Solution details 58

5.17.2.1 Architecture 58

5.17.2.2 Potential Procedures 59

5.17.2.2. Information flow 59

5.17.2.2.2 Key hierarchy 64

5.17.3 Evaluation 65

5.18 Solution #18: Key separation for AKMA AFs using counters 65

5.18.1 Introduction 65

5.18.2 Solution details 66

5.18.3 Evaluation 66

5.19 Solution #19: Reusing KAUSF for AKMA 67

5.19.1 Introduction 67

5.19.2 Solution details 67

5.19.3 Evaluation 68

5.20 Solution #20: Key identification when implicit bootstrapping is used 68

5.20.1 Introduction 68

5.20.2 Solution details 68

5.20.2.1 Option 1 – Key Identifier calculated from the keys 68

5.20.2.2 Option 2 – Reuse of ngKSI 68

5.20.3 Evaluation 69

5.21 Solution #21: Combining implicit bootstrapping solutions for usage of KAUSF or KSEAF as AKMA root key 70

5.21.1 Introduction 70

5.21.2 Solution details 70

5.21.2.1 Generic procedure 70

5.21.2.2 Home Network Option 71

5.21.2.3 Serving Network Option 71

5.21.2.4 Combined Option 72

5.21.3 Evaluation 72

5.22 Solution #22: Key freshness in AKMA 72

5.22.1 Introduction 72

5.22.2 Solution details 72

5.22.3 Evaluation 73

5.23 Solution #23: Implicit bootstrapping using NEF as the AKMA Anchor Functions 74

5.23.1 Introduction 74

5.23.2 Solution details 74

5.23.2.1 Architecture 74

5.23.2.2 Procedures 75

5.23.3 Evaluation 76

5.24 Solution #24: AKMA push 76

5.24.1 Introduction 76

5.24.2 Solution details 76

5.24.2.1 Architecture and reference points 76

5.24.2.2 Potential Procedures 77

5.24.2.2.1 Initiation 77

5.24.3 Evaluation 79

5.25 Solution #25: Key lifetimes 79

5.25.1 Introduction 79

5.25.2 Solution details 79

5.25.2.1 KAKMA lifetime 79

5.25.2.2 Application key lifetime 80

5.25.3 Evaluation 81

6 Evaluation and conclusion 81

6.1 Evaluation and conclusion on architecture and authentication procedures 81

6.2 Evaluation and conclusion on key management 81

6.2.1 Evaluation and conclusion on Key lifetimes (Key issue #12) 81

6.2.2 Evaluation and conclusion on Secure communication between UE and application server (Key issue #6) 82

6.3 Evaluation and conclusion on interfaces and protocols 82

6.4 Evaluation and conclusion on privacy 82

6.5 Evaluation and conclusion on API of AKMA in the UE 82

Annex A: Change history 83

# Foreword

This Technical Report 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.

In the present document, modal verbs have the following meanings:

**shall** indicates a mandatory requirement to do something

**shall not** indicates an interdiction (prohibition) to do something

The constructions "shall" and "shall not" are confined to the context of normative provisions, and do not appear in Technical Reports.

The constructions "must" and "must not" are not used as substitutes for "shall" and "shall not". Their use is avoided insofar as possible, and they are not used in a normative context except in a direct citation from an external, referenced, non-3GPP document, or so as to maintain continuity of style when extending or modifying the provisions of such a referenced document.

**should** indicates a recommendation to do something

**should not** indicates a recommendation not to do something

**may** indicates permission to do something

**need not** indicates permission not to do something

The construction "may not" is ambiguous and is not used in normative elements. The unambiguous constructions "might not" or "shall not" are used instead, depending upon the meaning intended.

**can** indicates that something is possible

**cannot** indicates that something is impossible

The constructions "can" and "cannot" are not substitutes for "may" and "need not".

**will** indicates that something is certain or expected to happen as a result of action taken by an agency the behaviour of which is outside the scope of the present document

**will not** indicates that something is certain or expected not to happen as a result of action taken by an agency the behaviour of which is outside the scope of the present document

**might** indicates a likelihood that something will happen as a result of action taken by some agency the behaviour of which is outside the scope of the present document

**might not** indicates a likelihood that something will not happen as a result of action taken by some agency the behaviour of which is outside the scope of the present document

In addition:

**is** (or any other verb in the indicative mood) indicates a statement of fact

**is not** (or any other negative verb in the indicative mood) indicates a statement of fact

The constructions "is" and "is not" do not indicate requirements.

# 1 Scope

The present document specifies key issues, derived requirements and potential solutions to support authentication and key management aspects for applications and 3GPP services based on 3GPP credentials in 5G, including the IoT use case. It analyses issues and requirements for:

- providing authentication and key management procedures to applications and 3GPP services in 5G scenarios which allow the UE to securely exchange data with an application server;

- decoupling these procedures from the transport protocol, in order to allow for the adaption to different application layer protocols.

The present document takes into account new solutions as well as potential adaptations to existing ones such as GBA described in TS 33.220 and BEST described in TS 33.163, in order to support the above mentioned requirements with procedures and protocols defined in SBA.

# 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 33.220: "Generic Authentication Architecture (GAA); Generic Bootstrapping Architecture (GBA)".

[3] 3GPP TS 33.163: "Battery Efficient Security for very low Throughput Machine Type Communication (MTC) device (BEST)".

[4] IETF RFC 3748: "Extensible Authentication Protocol (EAP)".

[5] 3GPP TS 33.905: "Recommendations for trusted open platforms".

[6] ["ISO/IEC JTC 1/SC 17 Cards and security devices for personal identification"](https://www.iso.org/committee/45144.html).

[7] 3GPP TS 27.007: "AT command set for User Equipment (UE) V15.3.0".

[8] IETF RFC 5191: "Protocol for Carrying Authentication for Network Access (PANA)".

[9] IEEE 802.1X™: "Port-Based Network Access Control".

[10] 3GPP TS 33.501: "Security architecture and procedures for 5G system (Release 15)".

[11] 3GPP TS 33.102: "3G Security; Security architecture (Release 15)".

[12] IETF RFC 5448: "Improved Extensible Authentication Protocol Method for 3rd Generation Authentication and Key Agreement (EAP-AKA')".

[13] 3GPP TS 33.223 (V15.0.0): "Generic Authentication Architecture (GAA); Generic Bootstrapping Architecture (GBA) Push function".

[14] 3GPP TS 23.434: " Service Enabler Architecture Layer for Verticals (SEAL); Functional architecture and information flows".

# 3 Definitions of terms, symbols and abbreviations

## 3.1 Terms

Void

## 3.2 Symbols

Void

## 3.3 Abbreviations

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

5GS 5G System

5GC 5G Core

AApF AKMA Application Function

AAuF AKMA Authentication Function

ABBAAnti-Bidding down Between Architectures

AKA Authentication and Key Agreement

AKAF AKMA Anchor Function

AKMA Authentication and Key Management for Applications

AKMA AF AKMA Application Function

AKRS AKMA Key Repository Service

AMF Access and Mobility Management Function

AP Application Processor

APDU Application Protocol Data Unit

ARPF Authentication credential Repository and Processing Function

AUSF Authentication Server Function

AUTN AUthentication TokeN

AV Authentication Vector

BEST Battery Efficient Security for very low Throughput Machine Type Communication (MTC) devices

BSF Bootstrapping Server Function

CK Cipher Key

CoAP Constrained Application Protocol

DoS Denial of Service

EAP Extensible Authentication Protocol

EAPoL EAP over LANs

EMSK Extended Master Session

KeyEPS Evolved Packet System

FQDN Fully Qualified Domain Name

GBA Generic Bootstrapping Architecture

GAA Generic Authentication Architecture

GUTI Globally Unique Temporary UE Identity

GPSI Generic Public Subscription Identifier

HPLMN Home Public Land Mobile Network

HRES Hash RESponse

HSS Home Subscriber Server

HTTP HyperText Transfer Protocol

HXRES Hash eXpected RESponse

IK Integrity Key

KDF Key Derivation Function

ME Mobile Equipment

MME Mobility Management Entity

MQTT Message Queuing Telemetry Transport

ngKSI Key Set Identifier in 5G

NAF Network Application Function

NAI Network Access Identifier

NAS Non Access Stratum

NEF Network Exposure Function

NF Network Function

OBU On Board Unit

PANA Protocol for Carrying Authentication for Network Access

PDCP Packet Data Convergence Protocol

PLMN Public Land Mobile Network

RES RESponse

SBA Service Based Architecture

SE Secure Element

SEAF SEcurity Anchor Function

SoR Steering of Roaming

SUCI SUbscription Concealed Identifier

SUPI SUbscription Permanent Identifier

UDM Unified Data Management

UE User Equipment

UICC Universal Integrated Circuit Card

UMTS Universal Mobile Telecommunications System

USIM Universal Subscriber Identity Module

UPF User Plane Function

XRES eXpected RESponse

# 4 Key Issues

## 4.1 Key Issue#1: Security Anchor

### 4.1.1 Issue detail

The GBA/GAA features specified in TS 33.220 [2] leverage the EPS/UMTS authentication infrastructure (especially the HSS) to provide the security between the UE and an application function in the network with which the UE interacts on the User Plane. It should be noted that GBA uses UMTS AKA and that the HSS provides the CK/IK to the BSF instead of KASME.

Figure 4.1.1-1 below shows the architecture of the features. GBA allows mutual authentication and the establishment of shared keys between the UE and BSF over the Ub interface. GAA, on the other hand, enables using such shared keys for protecting the access to a NAF. In principle, GBA keys can be used to secure any protocol between a UE and a NAF over the Ua interface.



Figure 4.1.1-1: GBA and GAA reference architecture from TS 33. 220 [2]

Since the AKMA feature is intended to leverage the 5GS authentication infrastructure to provide similar services, it is understood that GBA/GAA would be one of the starting points for the architectural design of AKMA. However, due to differences between the 5GS and EPS/UMTS there is no direct equivalent of the BSF and HSS in the 5GC. These differences include, but are not limited to, the following:

- The subscription data including the AKA credentials are stored in the UDM. However, it is another function, the AUSF, that is directly involved in the Primary Authentication procedure towards the serving PLMN.

- The Primary Authentication procedure establishes a shared key (KAUSF) between the UE and the AUSF while no such key exits in the EPS key hierarchy.

- The Primary Authentication is terminated in the AUSF by comparison to EPS where it is terminated in the MME.

- All the internal interfaces in the 5GC are SBA-based by comparison to the DIAMETER-based Zh and Zn interfaces in GBA.

As shown in Figure 4.1.1-2, the AKMA architecture will naturally include an AKMA Application Function with which the UE communicates over the User Plane. The AKMA AF interacts with an anchor function, the BSF-equivalent, in the 5G Core. It is only logical to assume that such an anchor function is needed to authenticate the UE and potentially to provide key management services towards the AKMA AF.



Figure 4.1.1-2: Role of the anchor function in the AKMA architecture

Therefore, solutions to this key issue need to address the following aspects.

- How the anchor function is realized.

- The interfaces involving the anchor function, the UE, the AKMA AF and other 5GS functions.

- The procedures flow for the UE authentication and the management of the resulting bootstrapped keys used to secure the communication between the UE and the AKMA AF.

### 4.1.2 Security Threat

Not applicable.

### 4.1.3 Potential architectural requirement

The AKMA architecture shall support an anchor function in the 5GC for UE authentication. This function can be realized by a standalone or an existing function.

## 4.2 Key Issue #2: Transport independent procedure definition

### 4.2.1 Issue details

In AKMA, application server needs to be able to securely exchange data with a UE based on the result of authentication and key derivation between mobile network and UE.

In AKMA, there are three different communication interfaces, namely, (1) the communication between UE and 3GPP network, (2) between UE and application server, and (3) between 3GPP network and application server. It is necessary to design the appropriate procedures. Considering the stage-3 work, the protocol used for AKMA procedure can be divided into two categories:

1. Using an existing transport protocol

The existing protocols for carrying parameters and transferring data refer to the protocols well designed and widely used by 3GPP, IETF and/or other standard organizations, e.g. PDCP layer protocol, TCP/IP, etc. Using such protocols can bring benefit for the procedure design, as the work can concentrated on the signaling/message flows. There will not be a need to pay much attention on considering how to design message type, format, and any other details as they are well defined in the protocols.

However, using existed protocol may bring some issue. If the communication is through specific application layer protocol, it will bring requirement for transport layer protocol. For example, if the communication is based on HTTP, then TCP is applied between UE and mobile network.

However, for some kinds of UEs, especially UE used for IoT, the resource is limited. It will influence UE can only implement few protocols due to its memory and calculation limitation. If application server communicates with UE by using specific application protocol, it implies that UE may could not implement other protocols. It raises the requirement for the communication between UE and mobile network. If the communication is based on specific protocol, some kinds of UE that could not implement such protocol is not able to support AKMA feature. That may limit AKMA usage.

2. Designing specific protocol for AKMA

Compared to using existed protocol, designing a specific protocol for AKMA allows for as much freedom as possible to design protocol types, formats and content. So specific protocol can be designed more flexible to fit for various lower layer protocols.

However, designing such specific protocol is generally difficult and it is debatable whether the protocol will be sufficiently robust. What is more, as it is newly defined, there will not be existing implementations. If only a custom designed protocol will be used, adoption of AKMA may be hampered by the lack of these implementations and competition of existing protocols. Depending on the use case, therefore, it should be considered to reuse existing protocols and only design new ones if existing protocols do not meet the specific requirements of AKMA.

### 4.2.2 Security Threats

N/A

### 4.2.3 Potential architecture requirements

Void

## 4.3 Key Issue #3: Mutual authenticationbetween UE and anchor function

### 4.3.1 Issue details

To allow UEs securely communicating and exchanging data with an application server using the authentication and key management procedures for applications in 5G scenarios, it is expected that the AKMA framework would be leveraged. Therefore, in order to establish secure communication between the UE and the application server, the UE and the anchor function need to be able to mutually authenticate each other based on the 5G authentication framework first before allowing the application server to leverage this authentication in order to establish secure communication between the UE and the application server.

### 4.3.2 Security Threats

Without authentication in the UE, an illegal UE may communicate with the anchor function and access AKMA services.

A fake anchor function may communicate with the UE that could potentially lead to the loss and exposure of user privacy.

### 4.3.3 Potential security requirements

The UE and the anchor function shall be able to mutually authenticate each other based on 5G credentials using the 5G authentication framework.

## 4.4 Key Issue #4: Authentication framework

### 4.4.1 Issue details

The 5GS AKMA framework needs an authentication framework so that only legitimate UEs can use the AKMA services. For example, it needs to be studied whether the AKMA authentication framework can leverage the fact that the primary authentication in the 5GS produces a key called the KAUSF at the AUSF in the HPLMN and the UE. The primary authentication meaning the one used to allow 5GS access to that UE. If that KAUSF could be the root key for the AKMA authentication framework, there would be no need for yet another authentication and therefore beneficial for IoT devices both signalling and processing wise. Recall that - in GBA/GAA architecture, the UE authentication (called bootstrapping) was separate and additional authentication than the primary/access authentication providing access to the 3GPP system.

A careful analysis is required on effects of potential security compromise of AKMA authentication on 3GPP primary authentication and vice-versa.

### 4.4.2 Security Threats

Without a proper security design, compromise on AKMA authentication can jeopardize security on 3GPP side.

### 4.4.3 Potential security requirements

The system shall support a secure authentication framework to allow only legitimate UEs to use AKMA services.

The system shall prevent a potential security compromise of AKMA authentication from propagating to the 3GPP primary authentication and NAS/AS security.

## 4.5 Key Issue #5: User privacy

### 4.5.1 Issue details

The Subscription Permanent Identifier (SUPI) is considered sensitive information, since attackers may identify an individual subscriber through his/her permanent ID. Combined with other kinds of information, such as geographic location, an attacker may be able to trace a subscriber, or obtain access to further sensitive information. Thus, the Subscription Permanent Identifier needs to be protected.

Meanwhile, the SUPI being the basis for providing any service in 5G a network, need to be known to the operator. This means that the operator is obliged to ensure that the SUPI is not revealed to any other parties.

When an operator wants to provide authentication and key management to an application server, it needs to have the ability to exchange information about a subscriber to enable the application server to determine the identity of its user. Hence, there is a need for another kind of identifier (permanent and/or temporarily) to identify users between the 3GPP network and an application server. And the MNO should be able to map the other kind of identifier to the permanent identifier of the MNO domain.

### 4.5.2 Security Threats

The Subscription Permanent Identifier may be leaked to unauthorized parties.

The application server may be unable to identify the user.

The operator may be unable to identify the users SUPI based on the new identifier between 3GPP network and application server.

### 4.5.3 Potential security requirements

SUPI shall not be revealed to application servers.

The system shall allow privacy protection of the SUPI when exchanged between the UE and the network for the purposes of AKMA services.

The 3GPP network shall be able to recover the SUPI based on an alternative identifier used between 3GPP network and application server.

## 4.6 Key Issue #6: Secure communication between UE and application server

### 4.6.1 Issue details

In current BEST [3] and GBA [2] solutions, 3GPP network is responsible to derive keys from the root subscriber authentication key K (e.g., KE2Menc,KE2Mint, Ks\_(int/ext)\_NAF) for UE and application server. However, the application server may not want to use the key derived from the 3GPP network authentication key K. The application server may have a policy requiring the use of its own independently generated key (e.g., application specific key), but still require the use of features provided by the 3GPP network to distribute such a key. The proposed mechanism can satisfy the demand of application providers who do not wish to establish the secure connection by using only a 3GPP credential.

In some scenarios, such as when the UE sends sensitive data to application server, the application security policy may require that the 3GPP network operator does not have accesses to that data. In addition, the services provided by the application server may be accessed by multiple applications. Therefore, it is desirable that a solution that addresses this key issue supports establishment of separate application specific keys for each application that are served by the application server.

### 4.6.2 Security threats

3GPP network may get access to sensitive data transferred between UE andapplications which is protected by the key derived from 3GPP network, or from 3GPP network and a pre-shared key (i.e., non-3GPP credential) if the pre-shared key gets compromised.

### 4.6.3 Potential security requirements

The UE and the application server shall be able to derive a session key for end-to-end security based on keys derived from the 3GPP network such that the 3GPP network is not able to access sensitive data transferred between UE and the application server.

NOTE: Having a 3GPP specified solution for this requirement is needed in cases where operators want to offer application layer based services to their customers (e.g. to verticals) in line with functionality specified in 3GPP TS 23.434 [14]. In that case such solutions enable seamless migration from LTE based solutions (e.g. 3GPP TS 33.163 [3]) to 5G, and they enable offers to customers (e.g. verticals) with international/multi-operator deployments.

## 4.7 Key Issue #7: Protecting subscriber's personal information in control and data traffic

### 4.7.1 Issue details

This key issue is about potential personal information contained in various control and data traffic messages.

If AKMA architecture uses some form of content in control and or data traffic which is privacy sensitive, those content need to be protected against attacks.

By attacks, it is meant that unauthorized entities attempt to identify subscriptions by getting hold of the privacy sensitive content in one or more protocol messages.

### 4.7.2 Security Threats

Unprotected privacy sensitive content in control and or data traffic make it easier for attackers to potentially identify subscribers.

### 4.7.3 Potential security requirements

The system shall support protecting the privacy sensitive content in control and data traffic used in the AKMA architecture.

## 4.8 Key Issue #8: Protection of AKMA architecture interfaces

### 4.8.1 Issue details

The interfaces utilized by the AKMA architecture between the 5G system and the 3GPP services and application functions (commonly called AKMA AF) are supposed to transfer key material and therefore needs to be properly evaluated.

### 4.8.2 Security Threats

In case the interfaces used by AKMA architecture lack confidentiality, integrity and replay protection between authenticated endpoints it will be possible for an attacker to eavesdrop, alter data unnoticed and replay packets.

### 4.8.3 Potential security requirements

The interfaces utilized by the AKMA architecture between the 5G system and the 3GPP services and application functions shall support confidentiality, integrity and replay protection between authenticated endpoints.

## 4.9 Key Issue #9: Key separation for AKMA AFs

### 4.9.1 Issue details

In a scenario where the 5G system provides cryptographic keys to AKMA Application Functions (either 3GPP services or third party applications), it is important to have key separation. In the sense that two separate AKMA AFs never utilize the same key.

### 4.9.2 Security Threats

If there is no key separation it can lead to a situation where one AKMA AF can decrypt traffic intended for another AKMA AF.

It would also allow the possibility for an actor to inject malicious packages which the UE would conclude as cryptographically correct.

### 4.9.3 Potential security requirements

The AKMA architecture shall support key separation for different AKMA AFs.

## 4.10 Key Issue #10: Compliance with local rules and regulations

### 4.10.1 Issue details

In different parts of the world, different rules and regulations apply with respect to the usage of cryptography. A service like AKMA that is intended to be deployed in many places around the globe should therefore be adaptable to the local situation.

In the case of AKMA, the operator is the facilitator of a service that can be used to agree a key between two parties which may not be under control of the operator. As such, operators in different parts of the world may be subject to some regulations with respect to providing key material to third parties.

Another potential use case of AKMA is that the operator facilitates end-to-end protection between a UE and a party outside of the operator domain. Also, in such cases, restrictions may be enforced by the regulators.

In order to enhance adoption of the service, AKMA needs to be made regulations aware such that the service can be used irrespective of where the UE resides.

### 4.10.2 Security Threats

There are no threats.

### 4.10.3 Potential security requirements

AKMA service shall be made such that it can comply with rules and regulations of the serving network;

AKMA service shall be able to signal if services are not available under the regulations of the serving network

## 4.11 Key Issue #11: Generic battery efficient end-to-end security

### 4.11.1 Issue details

In case of a battery constrained UE that communicate to a 3rd party Application Server, it may be needed to enable end-to-end security (i.e. between UE and Application Server) that is battery efficient.

### 4.11.2 Security threats

Not applicable.

### 4.11.3 Potential security requirements

The solution shall support UEs that are battery constrained.

## 4.12 Key Issue #12: Key lifetimes

### 4.12.1 Issue details

For GBA, specified in [2], lifetimes are defined for the anchor key (Ks) and the derived sub-keys (Ks\_(ext/int) NAF). The maximum lifetime for a sub-key is equal to the lifetime of the anchor key.

Introducing a lifetime for anchor keys and derived sub-keys could be reasonable for AKMA as well.

### 4.12.2 Security Threats

If the anchor key and the derived sub-keys do not have a lifetime, an attacker may use compromised keys for a long time.

### 4.12.3 Potential security requirements

Both anchor keys and derived sub-keys shall be provided with a maximum lifetime.

The lifetime of the derived sub-keys shall not exceed the lifetime of the anchor key.

Either end on AKMA interfaces shall allow for renegotiation of keys when key lifetime is expired.

## 4.13 Key Issue #13: API for AKMA keys in UE

### 4.13.1 Issue details

In GBA, the Network Application Function (NAF) has an interface, Zn, towards the Bootstrapping Server Function (BSF) as is shown in Key Issue 1. The NAF can request NAF specific keys from the BSF over the Zn. Similar interface is also expected to be defined between the AKMA application function (AF) and the AKMA security anchor. The benefit of having such standardized network interface is self-evident as it provides multivendor interoperability, i.e. it enables AFs from different vendors and application developers to request AKMA keys from the security anchor.

The ultimate purpose of the AKMA feature is to provide keys, which are used to secure application communication between an AF and an application running in the UE (called AKMA app). It is assumed that there will be a counterpart of the AKMA security anchor in the UE side (called AKMA bootstrapping client). See figure 4.13.1-1.



Figure 4.13.1-1: API within UE for fetching AKMA keys

While the AFs in the network side will have a standardized interface for fetching AKMA keys, as described above, such interface or API is missing in the UE side. This means that application developers would need to design, perhaps considerably, different versions of their AKMA apps depending on how AKMA keys are made available in different types of UEs. This could be an obstacle in adopting the use of AKMA keys for applications. Such API was not developed for GBA, but recommendations in this problem space were recorded in TR 33.905 [5]. Considerations in TR 33.905 could be useful to investigate in relation to this Key Issue.

Traditionally, 3GPP has not specified interfaces within the UE, except for the interface between the ME and UICC, which is a multivendor interface. Similarly, the interface between the AKMA bootstrapping client and AKMA apps could be seen as a multivendor interface as the developers of AKMA apps are assumed to be different from ME vendors.

Having such standardized API for requesting AKMA keys in the UE would mean less design effort for application developers as it would introduce multivendor interoperability also in the UE side. Thereby making AKMA more attractive for applications to use AKMA.

Solutions to this Key Issue should study the following aspects:

*-* How an API between an AKMA bootstrapping client and AKMA app could look like?

- What parameters are sent between the AKMA bootstrapping client and AKMA app?

- If and how does the AKMA bootstrapping client ensure that only authorized AKMA apps receive keys?

- If and how could considerations in TR 33.905 be useful in relation to this Key Issue?

### 4.13.2 Security Threats

Not applicable.

### 4.13.3 Potential security requirements

Not applicable.

## 4.14 Key Issue #14: Key revocation

### 4.14.1 Issue details

In key issue #12, lifetimes for the anchor key and derived sub-keys are discussed. A potential requirement is made that the lifetime of derived sub-keys (application keys) shall not exceed the lifetime of the anchor key.

To avoid re-negotiation of all sub-keys when the anchor key expires, one possibility is to continue to use these keys until their individual lifetime expires.

However, failure of the negotiation of a new anchor key implies that the UE is no longer authenticated. But according to the above, the derived sub-keys might still be in use.

Hence, a revocation procedure for application keys is needed in case there is no longer a valid anchor key.

### 4.14.2 Security Threats

If application keys cannot be revoked, there is a risk that a UE continues to use applications although the re-authentication of the UE fails or if the anchor key is compromised.

If an attacker can revoke application keys, there is a risk of DoS.

### 4.14.3 Potential security requirements

It shall be possible for the home network to revoke application keys securely.

## 4.15 Key Issue #15: Synchronization of keys when using established keys

### 4.15.1 Issue details

During authentication, the UE and the network will derive a number of keys, in the following order:

- KAUSF;

- KSEAF;

- KAMF;

Some of these established keys (or newly introduced keys specifically for AKMA) could be used as bootstrapping keys for AKMA. A likely candidate is KAUSF, which is what will be referred to in this key issue.

The KAUSF, however, is not part of the security context and a new KAUSF is derived at both the UE and the AUSF with any authentication, even if the resulting security context is never taken into use. As such, the AUSF and the UE may have a different view of which key is the current KAUSF. Similarly, if a specific key for AKMA is derived at the moment of authentication, the UE and the AKMA server may also get out of sync. As a consequence, the AKMA service may not be established because the UE and the AUSF / AKMA server are out of sync.

### 4.15.2 Security Threats

No service if the UE and AUSF / AKMA server are out of sync.

### 4.15.3 Potential security requirements

If established keys are used for AKMA, the keys shall be identifiable.

Potential AKMA use of established keys shall not lead to a denial of service.

## 4.16 Key Issue #16: Application key freshness of AKMA

### 4.16.1 Issue details

AKMA as a key agreement scheme should guarantee of freshness of the application key KAF. That is if an AKMA AF that requests a key from the Anchor Function, that key might have already been used. In general key freshness is a desirable property of any method used to establish keys and should be included in AKMA.

### 4.16.2 Security Threats

If a KAF is used without freshness, then a weakness between UE and AKMA AF may allow an attacker to pretend to be a particular AKMA AF and obtain KAF. The attacker can masquerade as the UE towards the real AKMA AF.

### 4.16.3 Potential security requirements

It shall be possible to ensure the freshness of keys used between the UE and the AKMA application function.

## 4.17 Key Issue #17: AKMA push

### 4.17.1 Issue details

The GBA push feature specified in TS 33.223 [13] is a mechanism to bootstrap the security between a NAF and a UE, without forcing the UE to contact the BSF to initiate the bootstrapping. With the GBA push, the NAF can share a secret key with the UE, and to push messages to UE securely.

Considering that the push mechanism is an efficient way for the message transmission initiated by application function, and the interworking operation between AKMA and GBA for backward compatibility, it would be beneficial to support the push mechanism for the AKMA.

However, the GBA push security solution cannot be reused here. In this study, AKMA has defined different authentication procedures compared with authentication method specified in GBA, e.g., EAP AKA', 5G AKA, etc. The push information generated based on the different authentication procedures shall be identified by the UE. Another difference would be that the keys already specified (e.g., KAUSF) in TS 33.501 may be reused in AKMA to generate the AKMA anchor key. Therefore, a new security mechanism for the AKMA anchor function to generate the AKMA push information and for the UE to verify push information are required.

### 4.17.2 Security Threats

Not applicable.

### 4.17.3 Potential security requirements

The system shall support the secure AKMA key push framework

# 5 Candidate Solutions

## 5.1 Solution #1: Introducing third party key to AKMA

### 5.1.1 Introduction

The secure transferring between the UE and the 3rd party not only requires secure connection, but to some extent protects data from leakage to untrusted parties even including MNOs, especially for some large CIoT corporations. Current GBA solution provides secure connection for the application providers based on 3GPP credentials, however, it lacks mechanism to ensure end to end security. Therefore, introducing a third-party key to AKMA is an optional ability provided by 3GPP networks to protect data from UE all the way to the application server. The 3rd party key is defined as a secret key shared by the application server and the UE for application level communication. According to 3rd party service security requirements, whenever necessary to application providers, they can choose to use derived keys from 3GPP credentials and 3rd party keys to secure the end to end connection. This way, application providers can control over the key material specifically.

### 5.1.2 Solution details

The proposed solution takes the current GBA procedure for example (Note: The related network elements and procedures in AKMA is FFS, the following figure 5.1.2-1 only illustrates the 3rd party key involving procedure). During the procedure using bootstrapped security association, after NAF fetches Ks\_(ext/int)\_NAF from BSF, if necessary, the 3rd party executes end to end key derivation and sends to UE an e2e flag indicating the use of combination key scheme. According to the e2e flag, the UE derives the end to end key which is used for the following secure connection between UE and 3rd party.



Figure 5.1.2-1: Third party key to AKMA

The e2e\_key is derived according to:

e2e \_key = KDF (Ks\_(ext/int)\_NAF, Ka);

where Ka is the 3rd party key defined in clause 5.1.1.

NOTE: Derivation algorithm is not detailed in this solution and could be designed in the normative work if this solution is recommended.

### 5.1.3 Evaluation

This solution addresses key issue#6-secure communication between UE and application server.

This solution provides an additional capability for application providers to choose using. If the application provider is not willing to rely on operators only, namely use session keys derived only from operators, it can choose to use an end to end key derived based on both operators (KAF) and the application provider itself (KA).

The advantage of this solution is the introduction of end to end key used between UE and application provider and this key is only kept by the UE and application provider itself, thus this solution meets the requirements stated in key issue#6.

## 5.2 Solution #2: Access independent architecture solution for AKMA

### 5.2.1 Introduction

This solution addresses KI#1, KI#2 and KI#4.

### 5.2.2 Solution details

#### 5.2.2.1 Architecture and reference points

The AKMA architecture includes two new Network Functions:

- The AKMA Authentication Function (AAuF), and

- The AKMA Application Function (AApF).

The AAuF is the authentication anchor that provides UE authentication services using the AKA credentials. The AAUF is responsible for authenticating the UE, generating the key material to be used between the UE and the AAPF and maintaining a UE AKMA context to be used for subsequent bootstrapping requests and hence possibly avoiding a full re-authentication run. This solution does not currently take any stand on how the AAuF is realized, i.e. whether by a standalone NF or by the AUSF.

The AAuF interacts with the UE over the a1 reference point. The AAuF interacts with the AUSF and the AApF using Service-Based Interfaces.

The AApF is the function that benefits from the AAuF authentication services. The AApF interacts with the UE over the a2 reference point and whenever needed requests keying material from the AAuF via Service-Based Interfaces.

Figure 5.2.2.1-1 below illustrates the proposed architecture.



Figure 5.2.2.1-1: AKMA reference architecture

#### 5.2.2.2 Procedures

##### 5.2.2.2.1 Initiation

In order to be able to secure the communication using AKMA, the UE and the AApF needs to first agree on its use. The procedure for negotiating the use of AKMA is given in Figure 5.2.2.2.1-1. The procedure is initiated by the UE sending a Request message not including any AKMA parameters and concluded by the AAuF sending an AKMA authentication required message. This is based on the GBA initiation procedure described in cl 4.5.1 of TS 33.220 [2].



Figure 5.2.2.2.1-1: Initiation procedure

##### 5.2.2.2.2 Authentication

The authentication procedure assumes the support of the EAP framework as specified in RFC 3748 [4] such that:

- The UE takes the role of the peer,

- The AAuF takes the role of a pass-through authenticator, and

- The AUSF takes the role of the backend authentication server.

The authentication procedure is initiated by the UE sending a Request message to the AAuF. Following the UE request the AAuF triggers the EAP authentication procedure by sending an AKMA authentication request to the AUSF. The AUSF and the UE would then engage in an exchange of EAP messages that is concluded by the AUSF sending an AKMA authentication response message to the AAuF carrying either an EAP success or an EAP failure. In case of success, the message includes as well the AKMA anchor key KAKMA. The AAuF forwards the EAP result message to the UE and in case of success includes the necessary AKMA parameters such as a temporary identifier and a validity time. The temporary identifier is used by the UE for subsequent Requests towards AApFs as long as the validity period has not elapsed.

When the UE is registered to the 5G System, the transport protocol for the EAP message over the User Plane depends on the type of the PDU session. For PDU sessions of IP type, the EAP messages are carried over IP using the PANA protocol specified in RFC 5191 [8]. For PDU sessions of Ethernet type, the EAP messages are carried using the EAPol protocol specified in IEEE 802.1X [9].

When the UE is not registered to the 5G System, it is required that the UE has IP connectivity as in the GBA feature. In such case the EAP messages are carried using the PANA protocol as described above.



Figure 5.2.2.2.2-1: Authentication procedure

##### 5.2.2.2.3 Usage

Once the UE has been successfully authenticated by the AAuF, the UE has the necessary keying material to establish secure communication with any AApF. In order to do that, the UE derives the application key KAF using the AApF identifier (FQDN) and possibly other parameters and supplies its temporary identifier to the AApF. The AApF then retrieves the application key from the AAuF.



Figure 5.2.2.2.3-1: Usage procedure

### 5.2.3 Evaluation

This solution addresses KI#1, KI#2 and KI#4. The solution includes a proposal for an authentication framework which includes an anchor function (AAuF) and it proposes the use of EAP-AKA' as a transport independent authentication procedure including the necessary adaptations for IP and Ethernet based PDU sessions.

The solution proposes a user plane authentication and key agreement procedure for the derivation of the AKMA anchor key KAKMA using an anchor function AAuF. The AAuF interfaces directly the UE and the AUSF for the realization of the authentication procedure. The solution proposes EAP-AKA' as the authentication method. The authentication procedure assumes the support of the EAP framework as specified in RFC 3748 [4] such that:

- The UE takes the role of the peer,

- The AAuF takes the role of a pass-through authenticator, and

- The AUSF takes the role of the backend authentication server.

The solution has potential impact on the following parts of the system:

- Potentially new NF: A new network function AAuF may need to be developed which interfaces with the UPF and can invoke the related SBA-based interfaces of the AUSF. The new NF is potentially standalone NF or part of another NF. The potential new NF needs to support the PANA and EAPoL protocols since the authentication method is EAP-AKA' and UP PDU sessions are of the IP or Ethernet type respectively.

- AUSF: The AUSF needs to implement new or re-use existing SBA interfaces for the authentication request and response from/to the AAuF.

- UE: The PANA and EAPoL protocols needs to be supported on the UE side.

The advantages of this solution are:

- The solution uses IP based interfaces which fulfils the requirements on the KI #2 for transport independence

The disadvantages of this solution are:

- Impact on the UE and core network for the support of the PANA and EAPoL protocols

- The solution is applicable for two out of three PDU session types (IP and Ethernet based).

- The solution introduces overhead by running a separate authentication

## 5.3 Solution #3: Architecture solution for AKMA with standalone anchor

### 5.3.1 Introduction

This solution addresses KI#1, KI#2 and KI#4.

### 5.3.2 Solution details

#### 5.3.2.1 Architecture and reference points

The AKMA architecture includes two new Network Functions:

* The AKMA Authentication Function (AAuF), and
* The AKMA Application Function (AApF).

The AAuF is the authentication anchor that provides UE authentication services. The AAuF is responsible for authenticating the UE, generating the key material to be used between the UE and the AApF and maintaining a UE AKMA context to be used for subsequent bootstrapping requests and hence possibly avoiding a full re-authentication run.

The AAuF interacts with the UE over the a1 reference point. The AAuF interacts with the UDM/ARPF and the AApF using Service-Based Interfaces.

The AApF is the function that benefits from the AAuF authentication services. The AApF interacts with the UE over the a2 reference point and whenever needed requests keying material from the AAuF via Service-Based Interfaces.

Figure 5.3.2.1-1 below illustrates the proposed architecture.



Figure 5.3.2.1-1: AKMA reference architecture

#### 5.3.2.2 Procedures

##### 5.3.2.2.1 Initiation

In order to be able to secure the communication using AKMA, the UE and the AApF needs to first agree on its use. The procedure for negotiating the use of AKMA is given in Figure 5.3.2.2.1-1. The procedure is initiated by the UE sending a Request message not including any AKMA parameters and concluded by the AAuF sending a required AKMA authentication message. This is based on the GBA initiation procedure described in cl 4.5.1 of TS 33.220 [2].



Figure 5.3.2.2.1-1: Initiation procedure

##### 5.3.2.2.2 Authentication

The authentication procedure assumes the support of the EAP framework as specified in RFC 3748 [4] such that:

- The UE takes the role of the peer.

- The AAuF takes the role of EAP authentication server.



Figure 5.3.2.2.2-1: Authentication procedure

The authentication procedure is initiated by the UE sending a Request message to the AAuF.

Following the UE request the AAuF requests AV from the UDM/ARPF.

AAuF triggers the EAP authentication procedure by sending an EAP request to the UE. The AAuF and the UE would then engage in an exchange of EAP messages that is concluded by the AAuF sending an AKMA authentication response message to the AAuF carrying either an EAP success or an EAP failure. In case of success, the AAuF derives the AKMA anchor key KAKMA.

The AAuF forwards the EAP result message to the UE and in case of success includes the necessary AKMA parameters such as a temporary identifier and a validity time. The temporary identifier is used by the UE for subsequent Requests towards AApFs as long as the validity period has not elapsed.

When the UE is registered to the 5G System, the transport protocol for the EAP message over the User Plane depends on the type of the PDU session. For PDU sessions of IP type, the EAP messages are carried over IP using the PANA protocol specified in RFC 5191 [8]. For PDU sessions of Ethernet type, the EAP messages are carried using the EAPol protocol specified in IEEE 802.1X [9].

When the UE is not registered to the 5G System, it is required that the UE has IP connectivity as in the GBA feature. In such case the EAP messages are carried using the PANA protocol as described above.

##### 5.3.2.2.3 Usage

Once the UE has been successfully authenticated by the AAuF, the UE has the necessary keying material to establish secure communication with any AApF. In order to do that, the UE derives the application key KAF using the AApF identifier (FQDN) and possibly other parameters and supplies its temporary identifier to the AApF. The AApF then retrieves the application key from the AAuF.



Figure 5.3.2.2.3-1: Usage procedure

### 5.3.3 Evaluation

This solution addresses KI#1, KI#2 and KI#4. The solution includes a proposal for an authentication framework which includes an anchor function (AAuF) and it proposes the use of EAP-AKA' as a transport independent authentication procedure including the necessary adaptations for IP and Ethernet based PDU sessions.

The solution proposes a user plane authentication and key agreement procedure for the derivation of the AKMA anchor key KAKMA using a standalone anchor function AAuF. The AAuF interfaces directly the UE and UDM/ARPF for the realization of the authentication procedure. The solution proposes EAP-AKA' as the authentication method. The authentication procedure assumes the support of the EAP framework as specified in RFC 3748 [4] such that:

- The UE takes the role of the peer,

- The AAuF takes the role of EAP authentication server

The solution has potential impact on the following parts of the system:

- Potentially new NF: A new network function AAuF may need to be developed which interfaces with the UPF and can invoke the related SBA-based interfaces of the UDM/ARPF. The potential new NF needs to support the PANA and EAPoL protocols since the authentication method is EAP-AKA' and UP PDU sessions are of the IP or Ethernet type respectively.

- UDM/ARPF: The UDM/ARPF may require distinguishing and possibly recording the network function which requests authentication.

- UE: The PANA and EAPoL protocols need to be supported on the UE side.

The advantages of this solution are:

- The solution uses IP based interfaces which fulfils the requirements on the KI #2 for transport independence

The disadvantages of this solution are:

- Impact on the UE and core network for the support of the PANA and EAPoL protocols

- The solution is applicable for two out of three PDU session types (IP and Ethernet based).

- The solution introduces overhead by running a separate authentication

## 5.4 Solution #4: Bootstrapping authentication of AKMA

### 5.4.1 Introduction

This solution addresses key issue #3: Mutual authenticate between UE and anchor function.

The key issue proposes that the UE and the anchor function are able to mutually authenticate each other based on 5G credentials using the 5G authentication framework, i.e., 5G AKA and EAP-AKA'. In addition, during the authentication between UE and anchor function, a shared key Ks between UE and anchor function is derived. It is assumed that the anchor function is connected to the AUSF.

### 5.4.2 Solution details

#### 5.4.2.0 General

When a UE wants to interact with an AKMA AF, and it knows that the bootstrapping procedure is needed, it shall first perform a bootstrapping authentication (see Figure 5.4.2.1-1). The authentication frameworks 5G AKA and EAP-AKA' in TS 33.501 are leveraged.

#### 5.4.2.1 Potential Authentication procedure for 5G AKA



Figure 5.4.2.1-1: The bootstrapping authentication procedure for 5G AKA

The authentication procedure for 5G AKA works as follows, see also Figure 5.4.2.1-1:

1. The UE sends a request towards the Anchor Function.

2. The Anchor Function shall invoke the Nausf\_UEAuthentication service by sending a Nausf\_UEAuthentication\_Authenticate Request message to the AUSF, in which the user identity and Anchor Function identifier shall be included.

3. The AUSF shall send a Nudm\_UEAuthentication\_Get Request to the UDM.

4. The UDM/ARPF shall create a 5G HE AV from RAND, AUTN, XRES\*, and KAUSF. The UDM shall then return the 5G HE AV to the AUSF.

5. The AUSF shall store the XRES\* temporarily. The AUSF shall compute the HXRES\* from XRES\* and Anchor Function key KAKMA from KAUSF and Anchor Function identifier. The AUSF shall then generate the 5G AV from the 5G HE AV received from the UDM/ARPF by replacing the XRES\* with the HXRES\* and KAUSF with KAKMA in the 5G HE AV.

6. The AUSF shall return the 5G SE AV (RAND, AUTN, HXRES\*) to the Anchor Function.

7. The Anchor Function shall send RAND, AUTN to the UE.

8. At receipt of the RAND and AUTN, the USIM shall verify AUTN and compute a response RES. The ME then shall compute RES\* from RES. The ME shall calculate KAUSF from CK||IK and KAKMA from KAUSF and Anchor Function identifier.

9. The UE shall return RES\* to the Anchor Function.

10. The Anchor Function shall then compute HRES\* from RES\*, and the Anchor Function shall compare HRES\* and HXRES\*. If they coincide, the Anchor Function shall consider the authentication successful from the Anchor Function point of view.

11. The Anchor Function shall send RES\* as received from the UE to the AUSF.

12. When the AUSF receives the RES\*, it shall compare the received RES\* with the stored XRES\*. If the RES\* and XRES\* are equal, the AUSF shall consider the authentication as successful.

13. The AUSF shall indicate to the Anchor Function whether the authentication was successful or not from the home network point of view. If the authentication was successful, the KAKMA shall be sent to the Anchor Function in the Nausf\_UEAuthentication\_Authenticate Response.

14. The Anchor Function shall calculate a temporary identifier to bind the subscriber identity to the keying material. The temporary identifier value shall be generated in format of NAI by taking the base64 encoded and the Anchor Function identifier, i.e. base64encode(RAND)@ Anchor Function identifier. The Anchor Function shall send a response message to the UE to indicate the success of the authentication. This message shall also include the temporary identifier and the key lifetime of KAKMA.

#### 5.4.2.2 Potential Authentication procedure for EAP-AKA'



Figure 5.4.2.2-1: The bootstrapping authentication procedure for EAP-AKA'

1. The authentication procedure for EAP-AKA' works as follows, cf. also Figure 5.4.2.2-1. The UE sends a request towards the Anchor Function.

2. The Anchor Function shall invoke the Nausf\_UEAuthentication service by sending a Nausf\_UEAuthentication\_Authenticate Request message to the AUSF, in which the user identity and Anchor Function identifier shall be included.

3. The AUSF shall send a Nudm\_UEAuthentication\_Get Request to the UDM.

4. The UDM shall subsequently send this transformed authentication vector AV' (RAND, AUTN, XRES, CK', IK') to the AUSF. The UDM/ARPF shall compute CK' and IK' from Anchor Function identifier.

5. The AUSF shall send the EAP-Request/AKA'-Challenge message to the Anchor Function.

6. The Anchor Function shall transparently forward the EAP-Request/AKA'-Challenge message to the UE.

7. At receipt of the RAND and AUTN, the USIM shall verify AUTN and compute a response RES. The ME shall derive CK' and IK'.

8. The UE shall send the EAP-Response/AKA'-Challenge message to the Anchor Function.

9. The Anchor Function shall transparently forward the EAP-Response/AKA'-Challenge message to the AUSF.

10. The AUSF shall verify the message, and if the AUSF has successfully verified this message it shall continue as follows, otherwise it shall return an error.

11. The AUSF derives EMSK from CK' and IK'. The AUSF uses the first 256 bits of EMSK as the KAUSF and then calculates Anchor Function key KAKMA from KAUSF and Anchor Function identifier. The AUSF shall send an EAP Success message to the Anchor Function inside Nausf\_UEAuthentication\_Authenticate Response, which shall forward it transparently to the UE. Nausf\_UEAuthentication\_Authenticate Response message contains the KAKMA.

12. The Anchor Function shall calculate a temporary identifier to bind the subscriber identity to the keying material. The temporary identifier value shall be generated in format of NAI by taking the base64 encoded and the Anchor Function identifier, i.e. base64encode(RAND)@ Anchor Function identifier. The Anchor Function shall send the EAP Success message to the UE. This message shall also include the temporary identifier and the key lifetime of KAKMA.

### 5.4.3 Evaluation

The solution proposes the bootstrapping authentication improvement based on 5G authentication framework, i.e., 5G AKA and EAP-AKA'. The authentication is via control plane which is different from GBA. For GBA, bootstrapping is performed via HTTP Digest AKA protocol which is the application protocol as specified in RFC 3310 [14] and is based on the 3GPP AKA TS 33.102 [11].

The authentication procedures are based on the standalone architecture which means the anchor function is a standalone function and connects to AUSF. During the authentication run, the shared key KAKMA is derived between the UE and AUSF. On the core network side, the KAKMA is derived by AUSF and sent to the anchor function. The authentication is applicable for AKMA specifically. If the UE initiates an AKMA service with AApF, the authentication run will be triggered afterwards.

The solution fulfils the security requirement of KI #3 which can be used to authenticate the UE and the anchor function.

## 5.5 Solution #5: Transport independent procedure using existing protocols by applying OneM2M protocol binding mechanism

### 5.5.1 Introduction

OneM2M is a global standard organization aimed at developing the technical specification of global service platform for IoT. It develops technical specifications which address the need for a common M2M Service Layer that can be readily embedded within various hardware and software, and relied upon to connect the myriad of devices in the field with M2M application servers worldwide. OneM2M has defined the exchanging message protocol between the entities (oneM2M Primitive), oneM2M core protocol to handle errors and bindings between core protocol and application layer transport protocol (CoAP, HTTP, MQTT). The protocol binding is when one or more than one interfaces are combined with other protocols, which is focused on message translation between oneM2M's request/response and binding target protocol's message.

### 5.5.2 Solution details

With reference to oneM2M protocol specifications [2], primitives are common service layer messages exchanged over the reference points in oneM2M architecture. In case of using an IP-based Underlying Network as illustrated in Figure 5.5.2-1, the primitives are mapped to application layer communication protocols such as HTTP, CoAP or MQTT which use TCP or UDP on the transport layer. The specification of primitives is independent of underlying communication protocols and allows introduction of bindings to other communication protocols.



Figure 5.5.2-1: Communication model using OneM2M protocol binding

By applying protocol binding mechanism to AKMA, UE and AKMA functions interact with each other through OneM2M primitives. Each CRUD+N （CREATE, RETRIEVE, UPDATE, DELETE and NOTIFY ）operation defined in OneM2M protocol consisting of request and response primitives, is to be mapped to CoAP methods or MQTT payload. As illustrated in Figure 5.5.2-1 (UE and AKMA functions can be both originators or receivers depending on interaction direction, the figure depicts UE sending requests to AKMA functions as an example), while UE sends requests to AKMA functions, it implements the binding function to map request messages to specific MQTT or CoAP messages for transferring. Upon receiving MQTT or CoAP messages, AKMA functions unbind the messages from specific transport protocol and execute the subsequent actions.

NOTE: This solution is incomplete due to lack of design on fitting protocol binding mechanisms into AKMA architecture, thus it’s not taken into consideration while concluding key issue#2.

### 5.5.3 Evaluation

This solution fulfils the requirement of transport independent procedure using existing protocols, thereby satisfying key issue #2.

## 5.6 Solution #6: Transport independent procedure using existing protocols by introducing a protocol transfer gateway

### 5.6.1 Introduction

To keep AKMA features applying for as many types of IoT devices as possible, a protocol transfer gateway/proxy can be introduced aiming at converting messages and communicating with terminals using different protocols. . In this solution, the AKMA architecture involving an AKMA transfer gateway (APTG) is introduced, corresponding procedures are proposed as well.

### 5.6.2 Solution details

#### 5.6.2.1 Architecture reference model



Figure 5.6.2.1-1: AKMA architecture reference model

##### 5.6.2.1.1 Entities

- AKMA Authentication Function (AAuF): the anchor function in AKMA is named as AAuF (AKMA Authentication Function). The AAuF interacts with the UE via the AKMA Protocol Transfer Gateway (APTG) over Service-Based interfaces.

- AKMA Application Function (AApF): interact with AAuF for AKMA application specific keys.

- AKMA Protocol Transfer Gateway (APTG): APTG translates messages between UE and AAuF. Since the UEs can be any of the devices running different application layer protocols of IoT (like MQTT，CoAP，etc.). APTG converts UE originated application messages to HTTP messages for AAuF processing. Similarly, APTG translates the messages sending from AAuF to UEs according to the UE types. In case of adding more IoT terminals based on different protocols, only the APTG is required to be upgraded.

NOTE: APTG could be co-located with AAuF.

##### 5.6.2.1.2 Service based interfaces

**Nausf:** Service-based interface exhibited by AUSF.

**Naauf:** Service-based interface exhibited byAAuF.

**Naapf:** Service-based interface exhibited by AApF.

**Naptg:** Service-based interface exhibited by APTG.

#### 5.6.2.2 Procedures



Figure 5.6.2.2-1: Authentication procedure

Step 1-2: UE and AApF agree on the use of AKMA with UE sending a request message including its application layer user ID to AApF, AApF then indicates the use of AKMA services by sending a response message, asking the UE to initiate the authentication request.

Step 3-8: The authentication procedure is initiated by the UE sending a Request message to the AAuF via APTG. APTG performs a “syntax" translation between the UE originated protocol and HTTP/HTTPS. Upon receiving the request from UE, APTG simply translates the message to HTTP/HTTPS message and forward it to AAuF. AUSF executes the authentication by checking the stored authentication result of requesting UE, where the result is obtained from the primary authentication. If the UE is legitimate, AUSF derives the intermediate key from KAUSF for AKMA use, which is KAKMA, and sends it to AAuF. Using KAKMA, AAuF derives the application specific key KAApF and keeps it in storage for subsequent use. AAuF generates a temporary identity named TID for the user, sends it to the UE via APTG along with the key lifetime of KAApF.

Step 9-12: UE derives the KAApF and initiates the AKMA use request message carrying TID to AApF. AApF asks AAuF for KAApF and its lifetime, indicating the UE of the use of KAApF by sending a response message.

### 5.6.3 Evaluation

This solution addresses key issue #1, #2, #3, #4.

It proposes to introduce a protocol transfer gateway into AKMA architecture based on solution #2, thus makes it possible for AKMA services being applied to every kind of IoT devices using different application protocols. Besides, this solution leverages AUSF to directly authenticate UEs without running a separate authentication procedure.

One of the advantages of this solution is it applies to all the IoT devices no matter what application protocols they are using. Another benefit of this solution is that it reuses the KAUSF and avoids running another AKA procedure, which simplifies the authentication procedures.

However, this solution requires additional stage 3 protocol design on interfaces between APTG and AAuF if they are separate network functions.

## 5.7 Solution #7: UE implementation scheme- AKMA framework and application on modem

### 5.7.1 Introduction

To enable authentication and application key management using AKMA, 3GPP AKA protocol can be leveraged to bootstrap application security. 3GPP AKA is running on UICC with CK and IK generated to be provided for session key derivation. An AKMA logic module should be implemented on UE to achieve AKMA procedures with network functions. In this scheme, the AKMA logic module which is named as AKMA framework in the following details is implemented on modem, with applications utilizing AKMA capabilities running on modem as well.

### 5.7.2 Solution details

Figure 5.7.2-1 illustrates a UE implementation scheme that both AKMA framework and application are on modem. AKA module is running on UICC to receive AUTN and RAND as input from ME and return RES and CK/IK as output. AKMA framework can derive session keys and subsequent application keys based on CK and IK obtained from AKA module. Applications on modem interfaces with AKMA framework to obtain an application authentication identifier. AKMA framework requests for CK and IK via APDU (Application Protocol Data Unit) packets according to ISO 7816 [6] protocols. Besides, there could be other instructions, parameters like request/response, keys, identifiers, etc., transferred between AKMA framework and UICC.



Figure 5.7.2-1: UE implementation scheme-AKMA framework and application on modem

### 5.7.3 Evaluation

This solution addresses key issue#13，proposing a UE implementation scheme with AKMA framework and applications implemented on modem.

This scheme is IoT applicable since it can be implemented without UE application processors, applications are running on modem to utilize AKMA capabilities directly. This scheme is suitable for smart meters, etc. But this scheme requires modems to store the AKMA intermediate keys securely.

## 5.8 Solution #8: UE implementation scheme- AKMA framework on UICC and application on modem

### 5.8.1 Introduction

To enable authentication and application key management using AKMA, 3GPP AKA protocol can be leveraged to bootstrap application security. 3GPP AKA is running on UICC with CK and IK generated to be provided for session key derivation. An AKMA logic module should be implemented on UE to achieve AKMA procedures with network functions. In this scheme, the AKMA framework is on UICC, with applications utilizing AKMA capabilities running on modem.

### 5.8.2 Solution details

Figure 5.8.2-1 illustrates a UE implementation scheme with AKMA framework on UICC and the applications on modem. Modem sends instructions and parameters to UICC via APDU (Application Protocol Data Unit) packets according to ISO7816 [6] protocols. In this case, the key derivations are UICC-based.



Figure 5.8.2-1: UE implementation scheme-AKMA framework on UICC and application on modem

### 5.8.3 Evaluation

This solution addresses key issue#13，proposing a UE implementation scheme with AKMA framework implemented on UICC and applications implemented on modem.

This scheme is IoT applicable since it can be implemented without UE application processors, applications are running on modem to utilize AKMA capabilities directly. This scheme is suitable for smart meters, etc. The advantage using this scheme is that UICC providing AKMA interfaces to modem ensures the secure storage of AKMA intermediate keys. But it may impact the UICC to run AKMA procedures inside the UICC.

## 5.9 Solution #9: UE implementation scheme- Application Processor (AP) scheme with AKMA framework on modem

### 5.9.1 Introduction

To enable authentication and application key management using AKMA, 3GPP AKA protocol can be leveraged to bootstrap application security. 3GPP AKA is running on UICC with CK and IK generated to be provided for session key derivation. An AKMA logic module should be implemented on UE to achieve AKMA procedures with network functions. In this scheme, AKMA framework is implemented on modem, with application processors (AP) implemented on UE to enable applications utilizing AKMA capabilities.

### 5.9.2 Solution details

This solution is similar to the solution in clause 5.7 in terms of the interaction between AKMA framework and AKA module, while applications on application processor (AP) interfaces AKMA framework through AT commands specified in TS 27.007 [7].

* Open logical channel +CCHO
* Close logical channel +CCHC
* Generic UICC logical channel access +CGLA
* Restricted UICC logical channel access +CRLA

However, since the implementation of the above commands is optional in the specification, this kind of scheme is lack of mandatory command implementation.



Figure 5.9.2-1: AP scheme with AKMA framework on modem

### 5.9.3 Evaluation

This solution addresses key issue#13，proposing a UE implementation scheme with application processor involved and AKMA framework implemented on modem.

This scheme fits for UEs with separate application processors (APs), like V2X OBUs, etc. The drawback is this scheme requires modems to store the AKMA intermediate keys securely.

## 5.10 Solution #10: UE implementation scheme- Application Processor (AP) scheme with AKMA framework on UICC

### 5.10.1 Introduction

To enable authentication and application key management using AKMA, 3GPP AKA protocol can be leveraged to bootstrap application security. 3GPP AKA is running on UICC with CK and IK generated to be provided for session key derivation. An AKMA logic module should be implemented on UE to achieve AKMA procedures with network functions. In this scheme, AKMA framework is implemented on UICC, with application processors (AP) implemented on UE to enable applications utilizing AKMA capabilities.

### 5.10.2 Solution details

This solution is similar to the solution in clause 5.9 in terms of the interaction between AKMA framework and AKA module, while application processor (AP) interfaces with AKMA framework through AT commands specified in TS 27.007 [7]. As for AP interfacing AKMA framework via modem, there is the same issue due to AT command implementation introduced in clause 5.9.



Figure 5.10.2-1: AP scheme with AKMA framework on UICC

### 5.10.3 Evaluation

This solution addresses key issue#13，proposing a UE implementation scheme with application processor involved and AKMA framework implemented on UICC.

This scheme fits for UEs with separate application processors (APs), like V2X OBUs, etc. It requires UICC providing AKMA interfaces to APs via modem. The advantage using this scheme is it ensures the secure storage of AKMA intermediate keys in UICC. But it may impact the UICC to run AKMA procedures inside the UICC.

## 5.11 Solution #11: UE implementation scheme- AKMA framework implemented on Secure Element (SE)

### 5.11.1 Introduction

To enable authentication and application key management using AKMA, 3GPP AKA protocol can be leveraged to bootstrap application security. 3GPP AKA is running on UICC with CK and IK generated to be provided for session key derivation. An AKMA logic module should be implemented on UE to achieve AKMA procedures with network functions, which is implemented on a secure element in this scheme, with application processors implemented on UE to enable applications utilizing AKMA capabilities.

### 5.11.2 Solution details

In this solution, it is assumed that some intelligent terminals are equipped with secure elements (SE). In this case illustrated in Figure 5.11.2-1, AKMA framework can be implemented on SE. The application processor inputs CK and IK obtained from UICC to AKMA framework, and afterwards gets application authentication identifier from SE.



Figure 5.11.2-1: UE implementation scheme-AKMA framework implemented on SE

### 5.11.3 Evaluation

This solution addresses key issue#13，proposing a UE implementation scheme with AKMA framework implemented on SE.

The advantage of this scheme is the introduction of SE makes sure the security of AKMA intermediate keys storage. However, implementation with SE brings extra cost to UE, making it complicated while designing interfaces between SE and modem.

## 5.12 Solution #12: UE implementation scheme- AKMA framework implemented on application processor's OS

### 5.12.1 Introduction

To enable authentication and application key management using AKMA, 3GPP AKA protocol can be leveraged to bootstrap application security. 3GPP AKA is running on UICC with CK and IK generated to be provided for session key derivation. An AKMA logic module should be implemented on UE to achieve AKMA procedures with network functions. In this scheme, the AKMA framework is implemented on the application processor's operating system (OS) within the UE.

### 5.12.2 Solution details

Figure 5.12.2-1 illustrates the UE implementation scheme where AKMA framework is implemented on the application processor's operating system, the application authentication identifier is provided to upper layer applications via direct internal system calling.



Figure 5.12.2-1: AKMA framework implemented on application processor's OS

### 5.12.3 Evaluation

This solution addresses key issue#13，proposing a UE implementation scheme with AKMA framework implemented on application processors' OS.

The advantage of this scheme is that AKMA framework on AP makes it easier for applications to invoke AKMA interfaces. But it is obvious the security problems are exposed considering AKMA intermediate keys being leaked possibly.

## 5.13 Solution #13: AKMA authentication via the control plane

### 5.13.1 Introduction

This solution addresses KI#1, KI#2, KI#3 and KI#4.

In GBA [2], the bootstrapping (i.e. authentication to get a fresh master key Ks) requires only IP connectivity between the UE and BSF. While that means that GBA is access independent it also means that UEs need to support an additional authentication mechanism to run the AKMA bootstrapping compared to the access authentication mechanism. In case a UE has 5G connectivity, over 3GPP or non-3GPP access, it would be useful if the UE could re-use the access authentication mechanism to run AKMA bootstrapping via the control plane.

The re-use of access authentication mechanism is ***not*** meant to run primary authentication for AKMA purposes as this would interfere with the serving network authentication policy and serving network keys. Instead, the intention is to re-use the access authentication mechanism (as much as possible) in order to perform an independent AKMA authentication run with the purpose to produce an AKMA anchor key in the UE and home network AKMA anchor function (AAuF). Therefore, this solution is also independent of the key hierarchy resulting from the primary authentication.

On high level this solution works in the following way:

- UE sends an AKMA authentication request over NAS to AMF/SEAF.

- The AMF/SEAF recognizes that the request is about AKMA authentication and finds the correct home network entity, AAuF, and sends an AKMA authentication request to AAuF.

- AAuF contacts the UDM to get authentication vector for AKMA purposes.

- UDM provides authentication vector for AAuF and indicates the authentication method.

- AAuF performs 5G AKA or EAP-AKA' for AKMA purposes with the UE via the AMF/SEAF. Authentication messages between the UE and AMF/SEAF are sent over NAS.

- At the end of a successful AKMA authentication the UE and AAuF have a fresh AKMA anchor key KAKMA.

### 5.13.2 Solution details

#### 5.13.2.1 Architecture and reference points

The AKMA architecture includes two new Network Functions:

- The AKMA Authentication Function (AAuF), and

- The AKMA Application Function (AApF).

The AAuF is the authentication anchor that provides UE authentication services. The AAuF is responsible for authenticating the UE, generating the key material to be used between the UE and the AApF and maintaining a UE AKMA context to be used for subsequent requests from AApFs and hence possibly avoiding a full re-authentication run.

The AAuF interacts with the UE over the control plane via AMF/SEAF. The AAuF interacts with the UDM/ARPF, AMF/SEAF and the AApF using Service-Based Interfaces.

The AApF is the function that benefits from the AAuF authentication services. The AApF interacts with the UE over the a2 reference point and whenever needed requests keying material from the AAuF via Service-Based Interfaces.

Figure 5.13.2.1-1 below illustrates the proposed architecture

UDM

AUSF

SEAF

AAuF

UE

AApF

5GC

Figure 5.13.2.1-1: AKMA reference architecture

#### 5.13.2.2 Procedures

##### 5.13.2.2.1 Initiation

To be able to secure the communication using AKMA, the UE and the AApF needs to first agree on its use. The procedure for negotiating the use of AKMA is proposed in Figure 5.13.2.2.1-1. The procedure is initiated by the UE sending a Request message without including any AKMA parameters and concluded by the AAuF sending a required AKMA authentication message. This is based on the GBA initiation procedure described in clause 4.5.1 of TS 33.220 [2].



Figure 5.13.2.2.1-1: Initiation procedure

##### 5.13.2.2.2 AKMA authentication with EAP-AKA'

The authentication procedure assumes the support of the EAP framework as specified in RFC 3748 [4] such that:

- The UE takes the role of the peer.

- The AAuF takes the role of EAP authentication server.

Since the UE will not be authenticated by the serving network the AAuF does not send any key material to the AMF/SEAF.

UDM

AMF/SEAF

AAuF

UE

Auth Request

Auth Request

Auth Request

Auth Resp (AV)

EAP-AKA’ chall

EAP-AKA’ chall

EAP-AKA’ resp

EAP-AKA’ resp

EAP success (LT, tid )

EAP success (LT, tid )

Figure 5.13.2.2.2-1: Authentication procedure

The AKMA authentication procedure is initiated by the UE sending an AKMA authentication request message to the AMF/SEAF over NAS.

The AMF/SEAF recognizes that the request is about AKMA authentication, determines the correct AAuF and sends an AKMA authentication request to the AAuF.

The AAuF requests AV for AKMA purposes from the UDM/ARPF.

UDM provides authentication vector to the AAuF and indicates the authentication method.

The AAuF triggers the EAP authentication procedure by sending an EAP request / AKA' challenge to the AMF/SEAF.

The AMF/SEAF forwards the EAP request / AKA' challenge to the UE over NAS.

The UE processes the EAP request / AKA' challenge and, if successful, sends EAP response / AKA' challenge over NAS to the AMF/SEAF.

The AMF/SEAF forwards the EAP response / AKA' challenge to the AAuF.

The AAuF verifies EAP response / AKA' challenge and, if successful, derives the AKMA anchor key KAKMA.

The AAuF sends an EAP success, temporary identifier pointing to KAKMA. and a validity time to the AMF/SEAF.

The AMF/SEAF forwards the EAP success, temporary identifier and validity time to the UE over NAS.

Upon receiving the message from the AMF/SEAF, the UE derives the AKMA anchor key KAKMA.

The temporary identifier is used by the UE for subsequent requests towards AApFs as long as the validity period has not elapsed.

##### 5.13.2.2.3 AKMA authentication with 5G AKA

The procedure for running 5G AKA for AKMA purposes is described below. The AAuF terminates the 5G AKA authentication in the network side. It should be noted that since the UE will not be authenticated by the serving network, the AAuF does not send the 5G SE AV to the AMF/SEAF, but instead the AAuF processes the 5G HE AV and 5G SE AV itself. Also, the AAuF does not send any key material to the AMF/SEAF.

UDM

AMF/SEAF

AAuF

UE

Auth Request

Auth Request

Auth Request

Auth Resp (AV)

Auth Resp(RAND,AUTN)

RAND, AUTN

RES\*

RES\*

Result (LT, tid )

Result (LT, tid )

Figure 5.13.2.2.3-1: Authentication procedure

The AKMA authentication procedure is initiated by the UE sending an AKMA authentication request message to the AMF/SEAF over NAS.

The AMF/SEAF recognizes that the request is about AKMA authentication, determines the correct AAuF and sends an AKMA authentication request to the AAuF.

The AAuF requests AV for AKMA purposes from the UDM/ARPF.

UDM provides authentication vector to the AAuF and indicates the authentication method.

The AAuF triggers the 5G AKA AKMA authentication procedure by sending the RAND and AUTN to the AMF/SEAF.

The AMF/SEAF forwards the RAND and AUTN to the UE over NAS.

The UE processes the RAND and AUTN and, if successful, sends RES\* over NAS to the AMF/SEAF.

The AMF/SEAF forwards the RES\* to the AAuF.

The AAuF verifies RES\* and, if successful, derives the AKMA anchor key KAKMA.

The AAuF sends a result indication, temporary identifier pointing to KAKMA. and a validity time to the AMF/SEAF.

The AMF/SEAF forwards the result indication, temporary identifier and validity time to the UE over NAS.

Upon receiving the message from the AMF/SEAF, the UE derives the AKMA anchor key KAKMA.

The temporary identifier is used by the UE for subsequent requests towards AApFs as long as the validity period has not elapsed.

##### 5.13.2.2.4 Usage

Once the UE has been successfully authenticated by the AAuF, the UE has the necessary keying material to establish secure communication with any AApF. In order to do that, the UE derives the application key KAF using the AApF identifier (FQDN) and possibly other parameters and supplies its temporary identifier to the AApF. The AApF then retrieves the application key from the AAuF.



Figure 5.13.2.2.4-1: Usage procedure

### 5.13.3 Evaluation

This solution addresses KI#1, KI#2, KI#3 and KI#4. The solution includes a proposal for an authentication framework (KI#4) which includes an anchor function (AAuF, KI#1) and it proposes the use of control plane as a transport independent (KI#3) mutual authentication procedure (KI#2). The mutual authentication procedure is initiated by the UE using NAS messages which indicate that the NAS messages carry AKMA authentication messages; nevertheless, the procedure in the 5GC is similar to the primary authentication (with two methods, 5G AKA and EAP-AKA') with the difference that the AKMA anchor function (AAuF) plays the role of the AUSF.

The solution has potential impact on the following parts of the system:

- AMF: The AMF needs to process new NAS messages with the indication that the messages are to be used for AKMA authentication procedure. The AMF needs to interface the potential new NF (AAuF) that plays a similar role as the AUSF in primary authentication.

- Potentially new NF: A new network function AAuF, may need to be developed which interfaces with the AMF and can invoke the related SBA-based interfaces of the UDM/ARPF.

- UDM/ARPF: The UDM/ARPF may require distinguishing and possibly recording the network function which requests authentication.

- UE: New NAS messages with AKMA authentication procedure indication. The AKMA authentication procedure on the UE is similar to the primary authentication.

The advantages of this solution are:

- Minimal impact on the core network side: only the AMF and UDM potentially needs to be updated with minor updates.

- Minimal impact on the UE. The UE uses the existing NAS protocol for mutual authentication.

- The solution is not dependent on internal keys such as KAUSF, KSEAF for the KAKMA derivation.

The disadvantages of this solution are:

- A new authentication procedure on the 5GC.

## 5.14 Solution #14: Key revocation

### 5.14.1 Introduction

This solution addresses the key issue #14. According to the key issue, it is possible to revoke the application keys in a secure way.

### 5.14.2 Solution details

#### 5.14.2.1 Revocation in Application function

Between the anchor function and the application functions, there needs to be some secure interface in place providing confidentiality and authenticity, since this is used to transport the application keys. The revocation can be performed over that same interface and benefit from existing security procedures.

The anchor functions needs to keep a list of recipient AFs for each UE to keep track of which AFs to send a revocation to. the revocation request needs to at least include the application key identifier.



Figure 5.14.2.1-1

#### 5.14.2.2 Revocation in UE

For revocation in the UE there are the following options:

1) Send revocation request from Anchor function to UE bootstrapping client.

2) Let the application function handle the revocation by not providing any service to the UE when the application key is revoked.

3) Let the bootstrapping client in the UE handle revocation itself when authentication fails.

For the revocation to be authenticated, option1 could use the old anchor key. This is not optimal since the key is expired.

Option 2 might not work if the application function is not functioning properly due to crash or similar.

Option 3 relies on the UE itself being responsible for the revocation. Since the UE bootstrapping client and anchor function perform mutual authentication, the bootstrapping client will know when the authentication fails. The bootstrapping client can then revoke the application keys in the respective application.

The bootstrapping client needs to save a list of identities of derived keys and their respective application, to be able to perform the revocation.

The interface between the bootstrapping client and the application needs to provide confidentiality and authenticity, since this is used to transport the application keys. The revocation can be performed over that same interface and benefit from existing security procedures.

NOTE: Procedures after revocation are not included in this solution, this could be left for normative work if needed.



Figure 5.14.2.2-1

## 5.15 Solution #15: Implicit bootstrapping

### 5.15.1 Introduction

This solution addresses the key issue #3 (mutual authentication).

To save roundtrips for the bootstrapping, the AKMA anchor key could be generated at the time of primary authentication. It could configurable whether the AKMA key is generated or not depending on operator settings. The solution does not address whether the AKMA key is derived as part of the primary authentication in any case (pre-generated option) or when there is a need by AKMA to generate KAKMA (on-demand option). The details of the on-demand versus pre-generated key options are subject to normative work.

Note that this solution requires that the same subscription, and therefore the same credentials, are used for 5G access and for AKMA.

The AKMA anchor could be either the AUSF or a separate entity, here named AKMA Anchor Function, AKAF. Or it could be two entities co-located for convenience.

The AKMA anchor key, KAKMA could be derived either as a sibling (Figure 5.15.1-1) or child key (Figure 5.15.1-2) in relation to KAUSF.

The different options above are analysed in this solution.



Figure 5.15.1-1 KAKMA sibling to KAUSF



Figure 5.15.1-2 KAKMA child to KAUSF

This solution only applies to the derivation of the anchor key KAKMA.

Application keys are derived as described in TS 33.220 [2].

The preferred option for deriving KAKMA is the child key option as it does not have any impact on the UDM.

### 5.15.2 Solution details

#### 5.15.2.1 Authentication using EAP-AKA'

The procedure for EAP-AKA' defined in clause 6.1.3.1 of TS 33.501 [10] is the following. “The AUSF derives EMSK from CK' and IK' as described in RFC 5448[12] and Annex F. The AUSF uses the most significant 256 bits of EMSK as the KAUSF and then calculates KSEAF from KAUSF as described in clause A.6."

The proposed addition to the statement above for KAKMA is the following.

KAKMA is derived from EMSK.

This solution corresponds to the key hierarchy where the KAUSF and KAKMA are sibling keys.

However, this solution causes some problems if the AKMA key needs to be refreshed without performing a primary authentication, see clause 5.15.3.3.

Another option is to go for the hierarchy option where the KAKMA is a child key to KAUSF. In this case, KAKMA will be a sibling key to KSEAF and it can be derived similarly as KSEAF for example using another FC value and a counter value (to be used for re-fresh, see clause 5.15.3.3).

#### 5.15.2.2 Authentication using 5G AKA

For 5G AKA, the KAUSF is derived by the UDM (not by AUSF as in EAP-AKA´). Hence the UDM would be appropriate for derivation also of the KAKMA. Here there is no straight forward solution as for EAP-AKA´. One possible solution is to derive KAKMA similar to KAUSF but using another FC value.

This solution corresponds to the key hierarchy where the KAUSF and KAKMA are sibling keys.

However, this solution also causes some problems if the AKMA key needs to be refreshed without performing a primary authentication, see clause 5.15.3.3.

As for EAP-AKA´, derive KAKMA could be derived instead as a child key to KAUSF.

#### 5.15.2.3 AKMA key refresh

A new primary authentication also derives a new AKMA key, but application keys can continue to exist (see key issue #12 Key life-times).

If the new authentication fails, the AKMA key is revoked (see separate key issue #14).

The solutions with sibling keys do not support re-fresh of KAKMA without a primary authentication. To solve this, it could be possible to use the other key hierarchy option and derive KAKMA from KAUSF. This way, the refresh of KAKMA might be possible by a separate procedure creating some freshness parameters to the derivation of KAKMA. This could be a sequence number held by the AKAF and or AUSF.



Figure 5.15.2.3-1: AKMA key refresh

This refresh procedure is applicable regardless of which options is used for the primary authentication (EAP-AKA´ or 5G AKA).

The AKMA key also needs to be refreshed in the UE. This requires some synchronization between the UE and the AKAF or between UE and AUSF. Alternatively, the synchronization needs to be rely on both entities having synchronized time and that they choose to refresh the AKMA key when its lifetime approached its end. In Figure 5.15.2.3-2 an alternative where key refresh is signalled from AKAF to UE is shown.

Editor's note: How this signalling is to be made is TBD.



Figure 5.15.2.3-2: Signal key refresh to UE

### 5.15.3 Evaluation

The solution addresses key issue KI#3 (Mutual authentication) since it proposes procedures to address mutual authentication between UE and the AKMA anchor function (AKAF) by re-using the results of the primary authentication. The solution works regardless if the primary authentication uses EAP-AKA' or 5G AKA.

The solution outlines two main options for the derivation of the AKMA anchor key with respect to the KAUSF, the sibling option and the child option. The preferred option for the KAKMA key derivation is the child solution as it does not have any impact on the UDM. The details of the on-demand (pull) versus pre-generated key (push) options are subject to normative work. The way KAKMA is derived from EMSK is subject to normative work.

Note that this solution requires that the same subscription, and therefore the same credentials, are used for 5G access and for AKMA. With this solution there is no need for a separate and specific AKMA authentication.

The child option for the derivation of KAKMA involves only the AUSF for both 5G AKA and EAP-AKA'. The advantage of the KAKMA being the child key with respect to KAUSF, is that the KAKMA can be derived from the existing KAUSF using the procedure proposed in this solution, without the need for a new primary authentication run. One of the drawbacks of the child key option is that KAUSF is already used for other purposes (such as Steering of Roaming (SoR) as described in TS 33.501 [10]).

For the options on AKAF being a separate entity or not, the advantage of the AKAF being a separate unit is to separate it from the authentication functions, since the AKAF will also have interfaces to external application servers. However, if it is a separate unit, the interface between AUSF and AKAF need to have integrity and confidentiality protection for the transfer of AKMA keys.

The solution has impact on the following entities:

- UDM

- New parameter keeping track of AKMA key is to be derived by AUSF

- Potentially communicate AKMA usage to AUSF and UE (unless it is statically configured)

- AUSF

- Store the KAUSF after the completion of the primary authentication

- Derive KAKMA from KAUSF and parameters

- Transfer KAKMA, sequence number and ID to AKAF

- Support key refresh procedure

- AKAF

- Receive and store KAKMA, counter and ID

- Support key refresh procedure using counter

- Derive application keys on request from AFs

- UE

- Derive AKMA key

- Support key refresh procedure

The advantages of the solution are:

- No need for separate AKMA authentication.

- The same subscription, and therefore the same credentials, are used for 5G access and for AKMA.

The disadvantages of the solution are:

- KAUSF is used for other purposes (such as Steering of Roaming (SoR) as described in TS 33.501 [10])) which may result in complex key management if the refresh of KAKMA results in a forced primary authentication run.

## 5.16 Solution #16: Use of KSEAF as root key for KAKMA

### 5.16.1 Introduction

This solution addresses key issue #10 by proposing an architecture that allows for using KSEAF as AKMA root key. In addition, it supports key issue #1 by proposing a logical connection between the SEAF and the anchor function introduced in solutions #2 and #3.

In order to fulfil key issue #10, this solution introduces to use KSEAF by introducing a generic architecture where the AAuF communicates with a key repository service (AKMA Key Repository Service – AKRS) that provides services to store a key together with an identifier and to retrieve (derivations) of the stored key upon request. In this solution, the AKRS is a service offered of the SEAF/AMF or the AUSF even though technically, the ARKS could also be a standalone function. This does not change the solution.

In order to enable using KSEAF for AKMA, this solution proposes to include an information element in the AKMA service request message that indicates which key the UE prefers to use and to include an information element in the AKMA service response message that indicates which key the network has selected to be used for this run of the AKMA service.

### 5.16.2 Solution details

#### 5.16.2.1 AKMA Key Repository Service

##### 5.16.2.1.1 AKMA Key Repository Service Serving Network Architecture Option

In order to support the functionality of established key usage, an AKMA Key Repository Service (AKRS) is included in the AKMA Architecture. In this version of the architecture, the AKRS connects to the SEAF. This architecture option is therefore called the 'Serving Network'-option. The AKRS has the following functionality:

- Offering an interface to the AAuF to retrieve a KAKMA derived from KSEAF for AKMA purposes;

- Storing the KSEAF

Note: Storage of the KSEAF is offered in the SEAF already, so colocation of this service with the SEAF would reduce duplicate storage.

The service can be collocated with the AMF/SEAF but can also be run standalone. In this solution, collocation is assumed, and the service is referred to as AKRS.

The figure 5.16.2.1.1-1 below illustrates the proposed architecture.

5GC

UDM

AUSF

SEAF  
AKRS

AAuF

SBI

UE

a1

AAuP

a2

Figure 5.16.2.1.1-1: Architecture showing collocated AKRS and connections to the AauF

##### 5.16.2.1.2 AKMA Key Repository Service Home Network Architecture Option

In order to support the functionality of established key usage, an AKMA Key Repository Service (AKRS) is included in the AKMA Architecture. This service has the following functionality:

- Offering an interface to the AAuF to retrieve a KAKMA derived from KSEAF for AKMA purposes;

- Storing the KAUSF or KSEAF

Note 1: Storage of the KAUSF is offered in the AUSF already, so colocation of this service with the AUSF would reduce duplicate storage.

Note 2: The KSEAF can be calculated from the KAUSF. As such, there is no need to store both the KSEAF and the KAUSF.

The service can be collocated with the AUSF but can also be run standalone. In this solution, collocation is assumed, and the service is referred to as AKRS.

The figure 5.16.2.1.2-1 below illustrates the proposed architecture.

5GC

UDM

AUSF  
AKRS

SEAF

AAuF

SBI

UE

a1

AAuP

a2

Figure 5.16.2.1.2-1: Architecture showing collocated AKRS and connections to the AAuF

#### 5.16.2.2 AKMA Established Key Use Procedure

##### 5.16.2.2.1 Procedure

This procedure takes the place of the "Authentication Procedure" in solution #2, clause 5.2.2.2.2 and takes place after an initiation procedure as detailed in clause 5.2.2.2.1.

The established key use procedure is initiated by the UE sending a request message to the AAuF including a key identifier (KI), and a flag indicating that the UE would like to use KSEAF for AKMA purposes. The AAuF verifies whether the use is allowed according to local policy and regulations and sends an "EstablishedKeyUseForAKMARequest" message to AKRS instance on the SEAF or AUSF.

Upon reception of the message, the SEAF or AUSF fetches the appropriate key from storage and calculates the AKMA Key as follows:

KAKMA = KDF (Input key, "AKMA", AKMA Counter)

Note: If the instance on the AUSF fetches the KAUSF, it first needs to calculate the KSEAF before KAKMA can be calculated.

Where the Input key is or KSEAF and the AKMA counter is kept to avoid key repetition in case of multiple requests. Subsequently, the SEAF forwards the KAKMA and the value of the AKMA Counter to the AAuF together with a RAND and an XRES calculated from KAKMA and the RAND.

After reception of the key, the AAuF will authenticate the UE as follows by sending a message to the UE containing a flag indicating which key was used for calculation of KAKMA, the AKMA Counter, the random value RAND, and a MAC calculated as follows:

MAC = KDF (KAKMA, "AKMA MAC", RAND)

Upon reception of this message, the UE will calculate the KAKMA according to the key that was used, verify the MAC and if successful respond with a RES calculated from KAKMA and the RAND to the AAuF. The AAuF verifies that the RES is the same as the XRES and if so, replies with a service response including the temporary identifier and validity time.

The procedure is shown in figure 5.16.2.2.1-1.

UE

AAuF

AUSF

Service request (SUCI/SUPI, Key Flag)

Established Key Use request (UE ID)

Decide whether key reusage is allowed

Fetch key from storage, derive KAKMA

Key Use response (KAKMA, RAND, XRES)

Service response (Key, Flag RAND, MAC)

Verify MAC, calculate RES

Response (RES)

Verify RES

Service response (Temp ID, validity timer)

Figure 5.16.2.2.1-1: Established Key Use procedure

As a result of the procedure the following has been achieved:

- A KAKMA has been derived from KSEAF;

- The UE and key anchor have authenticated each other using the newly derived KAKMA

### 5.16.3 Evaluation

This solution addresses key issue #10 by proposing a method for deriving an AKMA root key from KSEAF. In order to do so, this solution introduces an AKMA Key Repository Service which can be collocated with either the SEAF or the AUSF.

In order to derive the AKMA key, the solution proposes two routes:

- Deriving KAKMA directly from KSEAF;

- Deriving KAKMA from KAUSF with having KSEAF in between.

The solution works for both the case that the AKRS is present in the serving network, as well as the case that the AKRS resides the home network.

In the serving network option, the authentication is performed between the UE and the serving network without involving the home network. Consequently, the home network has no control over the authentication, which may pose problems with charging and liability. Therefore, this option is not preferred.

In the home network option, an authentication is performed between the UE and the home network based on KSEAF. Because KSEAF is known to the serving network, the serving network could pose as the AUSF which also leads to a situation of loss of home network control over the authentication. The solution lacks a mitigating measure for this attack. Therefore, this option is not preferred either.

## 5.17 Solution #17: Efficient key derivation for end-to-end security

### 5.17.1 Introduction

This solution addresses Key Issues #1, #3, #4, #5. For key derivation for end-to-end security a solution is given that adds only a minimal amount of extra communication between the UE and the 3GPP network. The solution is based on the standard primary authentication and key agreement between UE and the 3GPP network as described in TS 33.501 [10]. In addition, it uses an enterprise key KEnterprise that is pre-shared between UE and AKMA Application Function (AApF) in order to derive an end-to-end encryption key KE2Eenc and an end-to-end integrity key KE2Eint. These two end-to-end keys may be used to protect the communication between the UE and the AApF. This is similar to the use of such keys as described in TS 33.163 [3].

### 5.17.2 Solution details

#### 5.17.2.1 Architecture

The AKMA reference architecture assumed for this solution is as depicted in Figure 5.17.2.1-1. It assumes the existence of an AKMA Authentication Function (AAuF) connected to the SEAF and to the AUSF. Moreover, the AAuF has connections to one or more AKMA Application Functions (AApF). The UE has control plane connectivity to the AAuF via the SEAF and it has user plane connectivity to the AApF directly.

The solution assumes that the AAuF is only involved for AKMA enabled UEs, which is indicated to the SEAF by the inclusion of an AKMA specific information element (e.g. AApF ID). For non AKMA enabled UEs the standard primary authentication procedure applies.

The solution enables both EAP-AKA' or 5G AKA to be used as primary authentication procedure. For EAP-AKA' the following roles are assumed:

- The UE takes the role of the peer.

- The SEAF takes the role of pass-through authenticator.

- The AAuF takes the role of pass-through authenticator.

- The AUSF takes the role of the backend authentication server.



Figure 5.17.2.1-1: AKMA reference architecture

#### 5.17.2.2 Potential Procedures

##### 5.17.2.2. Information flow

The information flows describing the solution are depicted in Figure 5.17.2.2.1-1, Figure 5.17.2.2.1-2, Figure 5.17.2.2.1-3, and Figure 5.17.2.2.1-3. The first diagram depicts the initiation of the primary authentication and key agreement procedure using the AKMA Authentication Function (AAuF) between SEAF and AUSF. The second and third diagram depict the authentication procedures for EAP-AKA' and 5G-AKA, respectively, using the AAuF between SEAF and AUSF. The fourth diagram depicts the calculation of the end-to-end encryption and integrity keys.



Figure 5.17.2.2.1-1: Initiation of authentication and selection of authentication method with AKMA additions

The initiation of authentication and selection of authentication method with AKMA additions works as follows, cf. Figure 5.17.2.2.1-1:

The SEAF receives a message from the UE during a procedure establishing a signalling connection (over N1). In the message the UE shall use the SUCI or 5G-GUTI in the Registration Request. In addition, it also includes an AKMA Application Function ID (AApF ID). This information element indicates to the SEAF that it shall involve an AKMA Authentication Function (AAuF) and it informs the AAuF for which AApF the AKMA authentication needs to be provided.

The SEAF shall invoke the Naauf\_UEAuthentication service by sending a Naauf\_UEAuthentication\_Authenticate Request message to the AAuF. The SEAF shall include either the SUCI or the SUPI according to TS 33.501 [10], the serving network name according to TS 33.501 [10], and the AApF ID received from the UE.

The AAuF shall invoke the Nausf\_UEAuthentication service by sending a Nausf\_UEAuthentication\_Authenticate Request message to the AUSF, and include the SUCI, SUPI and/or SN-name according to TS 33.501 [10].

After performing the appropriate checks according to TS 33.501 [10], the AUSF invoke the Nudm\_UEAuthentication\_Get service by sending a Nudm\_UEAuthentication\_Get Request to the UDM/ARPF. This message shall include the SUCI, SUPI and/or serving network name according to TS 33.501 [10].

Upon reception of the Nudm\_UEAuthentication\_Get Request, the UDM shall invoke SIDF if a SUCI is received. SIDF shall de-conceal SUCI to gain SUPI before UDM can process the request.

Based on SUPI, the UDM/ARPF shall choose the authentication method, based on the subscription data.



Figure 5.17.2.2.1-2: Authentication procedure for EAP-AKA' with AKMA additions

The authentication procedure for EAP-AKA' with AKMA additions works as follows, cf. Figure 5.17.2.2.1-2:

1. The UDM/ARPF shall first generate an authentication vector with Authentication Management Field (AMF) separation bit = 1 as defined in TS 33.102 [11]. The UDM/ARPF shall then compute CK' and IK' as per the normative Annex A of TS 33.501 [10] and replace CK and IK by CK' and IK'.
2. The UDM shall subsequently send this transformed authentication vector AV' (RAND, AUTN, XRES, CK', IK') to the AUSF from which it received the Nudm\_UEAuthentication\_Get Request together with an indication that the AV' is to be used for EAP-AKA' using a Nudm\_UEAuthentication\_Get Response message.

In case SUCI was included in the Nudm\_UEAuthentication\_Get Request, UDM will include the SUPI in the Nudm\_UEAuthentication\_Get Response.

1. The AUSF shall send the EAP-Request/AKA'-Challenge message to the AAuF in a Nausf\_UEAuthentication\_Authenticate Response message.
2. The AAuF shall transparently forward the EAP-Request/AKA'-Challenge message to the SEAF in a Naauf\_UEAuthentication\_Authenticate Response message.
3. The SEAF shall transparently forward the EAP-Request/AKA'-Challenge message to the UE in a NAS message Authentication Request message. The ME shall forward the RAND and AUTN received in EAP-Request/AKA'-Challenge message to the USIM. This message shall include the ngKSI and ABBA parameter. ngKSI will be used by the UE and AMF to identify the partial native security context that is created if the authentication is successful. The SEAF shall set the ABBA parameter as defined in Annex A.7.1 of TS 33.501 [10].
4. At receipt of the RAND and AUTN, the USIM shall verify the AUTN and the freshness of the AV' by checking whether AUTN can be accepted as described in TS 33.102 [11]. If so, the USIM computes a response RES. The USIM shall return RES, CK, IK to the ME. If the USIM computes a Kc (i.e. GPRS Kc) from CK and IK using conversion function c3 as described in TS 33.102 [11], and sends it to the ME, then the ME shall ignore such GPRS Kc and not store the GPRS Kc on USIM or in ME. The ME shall derive CK' and IK' according to Annex A.3 of TS 33.501 [10].
5. The UE shall send the EAP-Response/AKA'-Challenge message to the SEAF in a NAS message Authentication Response message.
6. The SEAF shall transparently forward the EAP-Response/AKA'-Challenge message to the AAuF in a Naauf\_UEAuthentication\_Authenticate Request message.
7. The AAuF shall transparently forward the EAP-Response/AKA'-Challenge message to the AUSF in a Nausf\_UEAuthentication\_Authenticate Request message.
8. The AUSF shall verify the message, and if the AUSF has successfully verified this message it shall continue as follows, otherwise it shall return an error to the SEAF via the AAuF. AUSF shall inform UDM about the authentication result (see sub-clause 6.1.4 of TS 33.501 [10] for details on linking authentication confirmation).
9. The AUSF and the UE may exchange EAP-Request/AKA'-Notification and EAP-Response /AKA'-Notification messages via the AAuF and SEAF. The AAuF and SEAF shall transparently forward these messages.
10. The AUSF derives EMSK from CK' and IK' as described in RFC 5448 [12] and Annex F of TS 33.501 [10]. The AUSF uses the most significant 256 bits of EMSK as the KAUSF and then calculates KSEAF from KAUSF as described in clause A.6 of TS 33.501 [10]. The AUSF shall also calculate KAKMA from KAUSF. The AUSF shall send an EAP Success message to the AAuF inside Nausf\_UEAuthentication\_Authenticate Response, which shall forward it transparently to the SEAF. Nausf\_UEAuthentication\_Authenticate Response message contains the KSEAF and KAKMA. If the AUSF received a SUCI from the SEAF when the authentication was initiated (see sub-clause 6.1.2 of TS 33.501 [10]), then the AUSF shall also include the SUPI in the Nausf\_UEAuthentication\_Authenticate Response message.

NOTE 1: The use of the KAKMA for deriving further keys is explained later in this solution.

1. The SEAF shall transparently forward the EAP Success message to the SEAF in a Naauf\_UEAuthentication\_Authenticate Response message. The Naauf\_UEAuthentication\_Authenticate Response message contains the KSEAF. The Naauf\_UEAuthentication\_Authenticate Response message shall contain the SUPI if the SUPI was received in the Nausf\_UEAuthentication\_Authenticate Response message. The Nausf\_UEAuthentication\_Authenticate Response message shall also contain a temporary user identifier Temp ID. The Temp ID is used by the UE in the communication with AApF.

14. The SEAF shall send the EAP Success message to the UE in the N1 message. This message shall also include the ngKSI and the ABBA parameter. The SEAF shall set the ABBA parameter as defined in Annex A.7.1 of TS 33.501 [10]. The message shall also contain Temp ID parameter received from the AAuF.

The key received in the Naauf\_UEAuthentication\_Authenticate Response message shall become the anchor key, KSEAF in the sense of the key hierarchy in sub-clause 6.2 of TS 33.501 [10]. The SEAF shall then derive the KAMF from the KSEAF, the ABBA parameter and the SUPI according to Annex A.7 of TS 33.501 [10] and send it to the AMF. On receiving the EAP-Success message, the UE derives EMSK from CK' and IK' as described in RFC 5448 [12] and Annex F of TS 33.501 [10]. The ME uses the most significant 256 bits of the EMSK as the KAUSF and then calculates KSEAF in the same way as the AUSF. The UE shall derive the KAMF from the KSEAF, the ABBA parameter and the SUPI according to Annex A.7 of TS 33.501 [10]. The UE shall also calculate the KAKMA from the KAUSF in the same way as the AUSF.

NOTE 2: The use of the KAKMA for deriving further keys is explained later in this solution.



Figure 5.17.2.2.1-3: Authentication procedure for 5G AKA with AKMA additions

The authentication procedure for 5G AKA with AKMA additions works as follows, cf. Figure 5.17.2.2.1-3:

1. For each Nudm\_Authenticate\_Get Request, the UDM/ARPF shall create a 5G HE AV. The UDM/ARPF does this by generating an AV with the Authentication Management Field (AMF) separation bit set to "1" as defined in TS 33.102 [11]. The UDM/ARPF shall then derive KAUSF (as per Annex A.2 of TS 33.501 [10]) and calculate XRES\* (as per Annex A.4 of TS 33.501 [10]). Finally, the UDM/ARPF shall create a 5G HE AV from RAND, AUTN, XRES\*, and KAUSF.

2. The UDM shall then return the 5G HE AV to the AUSF together with an indication that the 5G HE AV is to be used for 5G-AKA in a Nudm\_UEAuthentication\_Get Response. In case SUCI was included in the Nudm\_UEAuthentication\_Get Request, UDM will include the SUPI in the Nudm\_UEAuthentication\_Get Response.

3. The AUSF shall store the XRES\* temporarily. The AUSF shall compute the HXRES\* from XRES\* and KAnchor Function from KAUSF.

The AUSF shall then generate the 5G AV from the 5G HE AV received from the UDM/ARPF by computing the HXRES\* from XRES\* (according to Annex A.5 of TS 33.501 [10]) and KSEAF from KAUSF (according to Annex A.6 of TS 33.501 [10]), and replacing the XRES\* with the HXRES\* and KAUSF with KSEAF in the 5G HE AV. The AUSF shall also calculate KAKMA from KAUSF.

NOTE 3: The use of the KAKMA for deriving further keys is explained later in this solution.

4. The AUSF shall then create a 5G SE AV by removing the KSEAF and return the 5G SE AV (RAND, AUTN, HXRES\*) to the AAuF in a Nausf\_UEAuthentication\_Authenticate Response message.

5. The AAuF shall transparently forward the 5G SE AV to the SEAF in a Naauf\_UEAuthentication\_Authenticate Response message

6. The SEAF shall send RAND, AUTN to the UE in a NAS message Authentication Request. This message shall also include the ngKSI that will be used by the UE and AMF to identify the KAMF and the partial native security context that is created if the authentication is successful. This message shall also include the ABBA parameter. The SEAF shall set the ABBA parameter as defined in Annex A.7.1 of TS 33.501 [10]. The ME shall forward the RAND and AUTN received in NAS message Authentication Request to the USIM.

7. At receipt of the RAND and AUTN, the USIM shall verify the freshness of the 5G AV by checking whether AUTN can be accepted as described in TS 33.102 [11]. If so, the USIM computes a response RES. The USIM shall return RES, CK, IK to the ME. If the USIM computes a Kc (i.e. GPRS Kc) from CK and IK using conversion function c3 as described in TS 33.102 [11], and sends it to the ME, then the ME shall ignore such GPRS Kc and not store the GPRS Kc on USIM or in ME. The ME then shall compute RES\* from RES according to Annex A.4 of TS 33.501 [10]. The ME shall calculate KAUSF from CK||IK according to clause A.2 of TS 33.501 [10]. The ME shall calculate KSEAF from KAUSF according to clause A.6 of TS 33.501 [10]. An ME accessing 5G shall check during authentication that the "separation bit" in the AMF field of AUTN is set to 1. The "separation bit" is bit 0 of the AMF field of AUTN. The ME shall also calculate the KAKMA from KAUSF in the same way as the AUSF.

NOTE 4: The use of the KAKMA for deriving further keys is explained later in this solution.

8. The UE shall return RES\* to the SEAF in a NAS message Authentication Response.

9. The SEAF shall then compute HRES\* from RES\* according to Annex A.5 of TS 33.501 [10], and the SEAF shall compare HRES\* and HXRES\*. If they coincide, the SEAF shall consider the authentication successful from the serving network point of view. If not, the SEAF proceed as described in sub-clause 6.1.3.2.2 of TS 33.501 [10]. If the UE is not reached, and the RES\* is never received by the SEAF, the SEAF shall consider authentication as failed, and indicate a failure to the AUSF via the AAuF.

10. The SEAF shall send RES\*, as received from the UE, in a Naauf\_UEAuthentication\_Authenticate Request message to the AAuF.

11. The AAuF shall transparently forward the RES\*, as received from SEAF, to the AUSF in a Nausf\_UEAuthentication\_Authenticate Request message.

12. When the AUSF receives as authentication confirmation the Nausf\_UEAuthentication\_Authenticate Request message including a RES\* it may verify whether the AV has expired. If the AV has expired, the AUSF may consider the authentication as unsuccessful from the home network point of view. AUSF shall compare the received RES\* with the stored XRES\*. If the RES\* and XRES\* are equal, the AUSF shall consider the authentication as successful from the home network point of view. AUSF shall inform UDM about the authentication result (see sub-clause 6.1.4 of TS 33.501 [10] for linking with the authentication confirmation).

13. The AUSF shall indicate to the AAuF in the Nausf\_UEAuthentication\_Authenticate Response whether the authentication was successful or not from the home network point of view. If the authentication was successful, the KSEAF shall be sent to the SEAF in the Nausf\_UEAuthentication\_Authenticate Response. In case the AUSF received a SUCI from the SEAF in the authentication request (see sub-clause 6.1.2 of TS 33.501 [10]), and if the authentication was successful, then the AUSF shall also include the SUPI in the Nausf\_UEAuthentication\_Authenticate Response message. The AUSF shall also include the KAKMA in the Nausf\_UEAuthentication\_Authenticate Response.

14. The AAuF shall forward the Nausf\_UEAuthentication\_Authenticate Request message, as received from SEAF, to the SEAF in a Naauf\_UEAuthentication\_Authenticate Request message. The Naauf\_UEAuthentication\_Authenticate Request message shall not contain the KAKMA and it shall contain a temporary user identifier Temp ID. The Temp ID is used by the UE in the communication with AApF.

15. The SEAF shall send the Temp ID in a NAS message to the UE.

NOTE 5: Step 15 could be NAS Security Mode Command.

The authentication procedure for EAP-AKA' and 5G AKA are similar to the primary authentication and key agreement procedures described in TS 33.501 [10], with the following differences:

* An AKMA Authentication Function (AAuF) is used between SEAF and AUSF.
* The UE includes an AKMA Application Function ID (AApF ID) in the initial request, cf. Figure 5.17.2.2.1-1. The AApF ID is used on the one hand to indicate to the SEAF that an AAuF needs to be involved in the authentication, and on the other hand it provides the AAuF information about the AApF for which the KAF needs to be derived.
* The AUSF shall derive/calculate a KAKMA and include this in the communication to the AAuF.
* The SEAF shall forward the Temp ID, that it receives from the AAuF, to the UE.



Figure 5.17.2.2.1-4: Calculation of the end-to-end encryption and integrity keys

The calculation of the end-to-end encryption and integrity keys works as follows, cf. Figure 5.17.2.2.1-4:

1. The UE calculates the KAF from KAKMA depending on the AApF ID used in the authentication described before. The UE associates the KAF with the Temp ID received during the authentication.

2. The AAuF calculates the KAF from KAKMA depending on the AApF ID received in the authentication described before. The AAuF is associated the KAF with the Temp ID used in the authentication.

3. The AAuF sends the KAF and associated Temp ID to the appropriate AApF (corresponding to the AApF ID received during the authentication).

4. The UE calculates the KE2Eint and KE2Eenc from the corresponding KAF and the pre-shared KEnterprise according to the key hierarchy described in clause 5.17.2.2.2. These keys are associated with the Temp ID used in the authentication.

5. The AApF calculates the KE2Eint and KE2Eenc from the corresponding KAF and the pre-shared KEnterprise according to the key hierarchy described in clause 5.17.2.2.2. These keys are associated with the Temp ID used in the authentication.

6. The AApF may optionally send a response to the received message from the AAuF.

##### 5.17.2.2.2 Key hierarchy

The key hierarchy explaining the dependency of the various keys is depicted in Figure 5.17.2.2.2-1.



Figure 5.17.2.2.2-1: key hierarchy for battery efficient AKMA

The above key hierarchy is similar to the one described in TS 33.163 [3].

NOTE: Exact derivations of the keys KAKMA, KAF, KE2Eenc, and KE2Eint are not detailed in this solution and could be designed during the normative work if this solution is recommended.

### 5.17.3 Evaluation

This solution provides an efficient AKMA authentication method based a few additions to the standard primary authentication and key agreement procedures (both for EAP-AKA' and for 5G AKA).

This solution provides a key agreement method involving the use of pre-shared keys in the UE and the AKMA Application Function.

This solution does not address the aspect of AKMA key life times, key refresh and key revocation.

## 5.18 Solution #18: Key separation for AKMA AFs using counters

### 5.18.1 Introduction

This solution addresses KI#9 (Key separation for AKMA AFs). The assumption is that the UE has been successfully authenticated by the AAuF as described in the solution 3 or by the AUSF itself.

### 5.18.2 Solution details

Once the UE has been successfully authenticated, the UE has necessary keying material to establish secure communication with any AKMA application function. The key separation for a UE between any AKMA application function is supported using a 16-bit AKMA Application Function Counter (AF Counter). The UE and the AKMA authentication function (either AAuF or AUSF) initialize the AF Counter to ‘0' whenever an AKMA anchor key (KAKMA) is generated for a UE based on a 3GPP credential in 5G. The AF counter can be monotonically incremented for every new application key (KAF) generation and used as an input during KAF generation from the same KAKMA. To derive the application key the UE increments the locally stored AF counter and verify if it matches with the AF counter received from the AKMA application function. If the verification is successful, the UE generates the application key using the received AF Counter and the other AKMA parameters. The AF counter specific to the UE is managed by both the UE and the AKMA authentication function. The AKMA application key is derived as follows.

KAF = KDF (KAKMA, AApF ID, AF Counter)

where AApF ID is the AKMA Application Function Identifier.

It is recommended that application functions receiving the key do not use the key directly, but rather derive a further, e.g. protocol specific, key for separate sessions. Direct usage of the key in a weak security protocol may expose the key and compromise further use of the same key. For example, protocol specific key may be derived as follows:

KAF\_Prot = KDF (KAF, Protocol Identity)

where Protocol Identity is an identity that indicates the specific protocol for different applications.

Figure 5.18.2-1 illustrates the proposed key separation mechanism.



Figure 5.18.2-1: Key separation procedure

### 5.18.3 Evaluation

This solution covers the requirement of key issue #9, namely "The AKMA architecture shall support key separation for different AKMA AFs." It does so by introducing an application key (KAF) derived from KAKMA by including the AApF ID for separation and a counter for freshness as input parameters to guarantee key separation and freshness.

The context of this solution is limited to those cases where a pre-shared key is available in the UE and the AAuF. In this solution, it is assumed that this pre-shared key is KAKMA.

The advantage of this solution is that in addition to key separation, freshness is also provided by using a unique counter. A drawback is that the AAuF will have to maintain a counter. Therefore, the system impact is that the AAuF will be required to maintain a counter in order to make this solution work. Also, counter synchronization errors may occur, which are not detailed in this solution.

## 5.19 Solution #19: Reusing KAUSF for AKMA

### 5.19.1 Introduction

This solution addresses KI#1, KI#2, KI#3, KI#4 and KI#5.

### 5.19.2 Solution details

This solution introduces two new functions to 5GC:

- AKMA Authentication Function (AAuF)

- AKMA Application Function (AApF)



Figure 5.19.2-1: AKMA Architecture that reuses KAUSF

In this solution, no separate authentication is performed to support AKMA functionality. Instead, it reuses the 5G primary authentication for AKMA purposes. Therefore, it is assumed that the UE had successfully registered to the 5G core before invoking AKMA services. A successful 5G primary authentication results in KAUSF being stored at the AUSF and the UE.

The KAUSF is used for the following AKMA purposes:

- Deriving a KAUSF key identifier from KAUSF at the UE and the AUSF. The KAUSF key identifier is stored by the AUSF along with the KAUSF. The derived key identifier is transported from the UE in NAI format to the AApF where the “username" part of the NAI includes the UE's KAUSF key identifier and the “realm" part is set to home network identifier identifying the AUSF in the home network that holds KAUSF. If the AApF does not have context associated with the key identifier, then the AApF sends a request to AAuF with the key identifier to request AKMA keys for the UE. The KAUSF key identifier is equivalent to the B-TID in GBA and identifies the KAUSF key of the UE from which other AKMA keys are derived.

- Deriving a key KAKMA at the UE and the AUSF. The AUSF sends KAKMA to the AAuF. KAKMA is equivalent to key Ks for GBA in TS 33.220)

Both the AAuF and the UE use KAKMA to derive application specific keys needed for AKMA Application Functions (AApFs) in similar manner as for NAFs in GBA. This implies that existing GBA-based Ua protocols can be mostly reused (with necessary adaptations) by the UE and the AApF with AKMA and is denoted as Ua\* interface.

NOTE: Derivation of AKMA specific keys and the key identifier is not detailed in this solution and could be which could be designed during the normative work if this solution is recommended.

In this solution, the AKMA keys can only be refreshed by running a fresh primary authentication. This means that the AKMA key lifetime(s) cannot be shorter than the time interval between primary authentications.

### 5.19.3 Evaluation

This solution reuses primary authentication and the key KAUSF for AKMA, thus avoiding the need to perform a separate authentication for AKMA.

This solution supports user privacy as SUPI is never sent by the UE to the network. The derived KAUSF key identifier is also used by the AApF to identify the UE.

## 5.20 Solution #20: Key identification when implicit bootstrapping is used

### 5.20.1 Introduction

This solution addresses key issue #15 (Synchronization of keys when using established keys).

This solution introduces a key identifier in order to identify the key used for implicit bootstrapping. The key to be identified depends on the solution and can be an established key from the 5G key hierarchy or a key derived from this key hierarchy such as the KAKMA. This solution refers to the KAUSF and the AUSF, however, it can easily be generalized to also work for other keys and other network functions.

The solution has two options:

1) The key identifier is calculated from the keys;

2) The ngKSI is reused.

### 5.20.2 Solution details

#### 5.20.2.1 Option 1 – Key Identifier calculated from the keys

In this option, an AKMA KAUSF identifier (A-KI) is calculated from the KAUSF as follows:

A-KI = KDF (KAUSF, "AKMA").

The UE and the AUSF will store the KAUSF together with this identifier. The UE and AUSF may store more than one A-KI and KAUSF pair in order to address the desynchronization error situation.

In order to use the key, the procedure is as follows:

1) Whenever the UE starts an initiation procedure for AKMA, the UE will retrieve the A-KI corresponding to the latest KAUSF from memory. The UE will then send a service request according to solution 2 to the AKMA server including the A-KI of the KAUSF.

2) The AKMA server / AUSF looks up the key based on the A-KI received (and UE identity if included) and if found, uses this key for further procedures with the UE. If no key was found, the AUSF will either:

- Fall back to solution #2 and run an authentication; or

- Return an error message with another A-KI that the AUSF has in memory for the UE.

3) Upon reception of the response, the UE will either:

- Perform the authentication according to solution #2; or

- Retrieve the KAUSF that corresponds to the A-KI received, or if not found, return an error message.

#### 5.20.2.2 Option 2 – Reuse of ngKSI

In this option, the existing ngKSI is reused. In order to do so, the AUSF has to receive the ngKSI that is communicated to the UE. This can be achieved as follows:

**EAP AKA'**

After the SEAF has received the RES from the UE, the SEAF forwards the RES in a Nausf\_Authentication Authenticate Request message to the AUSF. In this message, the SEAF also includes the ngKSI.

The AUSF then stores the ngKSI together with the KAUSF.

**5G AKA**

After the SEAF has received the RES\* from the UE, the SEAF forwards the RES\* in a Nausf\_Authentication Authenticate Request message to the AUSF. In this message, the SEAF also includes the ngKSI.

The AUSF then stores the ngKSI together with the KAUSF.

**Binding of the ngKSI to KSEAF**

In order to make sure that both the UE and the AUSF have the same ngKSI, the calculation of the KSEAF is changed as follows to also include the ngKSI.:

KSEAF = KDF (KAUSF, Serving network name, ngKSI)

**Using the key**

In order to use the key, the procedure is as follows:

1) Whenever the UE starts an initiation procedure for AKMA, the UE will retrieve the ngKSI corresponding to the latest KAUSF from memory. The UE will then send a service request according to solution 2 to the AKMA server including the ngKSI of the KAUSF.

2) The AKMA server / AUSF looks up the key based on the ngKSI received (and UE identity if included) and if found, uses this key for further procedures with the UE. If no key was found, the AUSF will either:

- Fall back to solution #2 and run an authentication; or

- Return an error message with another ngKSI that the AUSF has in memory for the UE.

3) Upon reception of the response the UE will either:

- Perform the authentication according to solution #2; or

- Retrieve the KAUSF that corresponds to the ngKSI received or if not found, return an error message.

### 5.20.3 Evaluation

This solution addresses key issue #15 (Established Key Synchronization). For this key issue, the following requirements are met:

KI #15 - Requirement: If established keys are used for AKMA, the keys shall be identifiable.

The above requirement is met by calculating a key identifier from KAUSF and using this key identifier in subsequent procedures.

KI #15 – Requirement: Potential AKMA use of established keys shall not lead to a denial of service.

The above requirement refers to the possibility that the keys used to derive KAKMA may be out-of-sync. In this solution, a denial of service is mitigated by the use of the key identifier and by the AUSF storing multiple pairs of key and its key identifier.

This solution's context is limited to only the problem of key synchronization if implicit bootstrapping is used. If no implicit bootstrapping is used, this solution does not apply. However, the problem of key synchronization applies to a wider set of services defined in TS 33.501 [10]. As such, this solution may be superseded by a solution within the scope of TS 33.501 [10], if one is developed.

Two options discussed in the solution detail subclause have different system impacts. For option 1, the system impact is limited to the AUSF, which will have to store key identifiers along with KAUSF (and potentially store multiple pairs). The AUSF is already impacted if implicit bootstrapping is used. This option therefore has a limited additional system impact.

For option 2, the system impact is larger. Backwards compatibility is difficult to achieve because all three of the UE, the serving network and the home network have to be compatible with the mechanism. Furthermore, this option impacts existing signalling between the SEAF and AUSF during primary authentication. As such, this option is considered to have a large system impact. Option 2 is therefore not preferred.

The advantages of option 1 are that this solution is relatively straightforward. The disadvantage is that the AUSF has to store a key identifier in addition to storing KAUSF. Also, the solution works better if the AUSF stores multiple pairs of key identifier and KAUSF for each UE. This is a drawback because it requires additional storage in the AUSF and the UE.

## 5.21 Solution #21: Combining implicit bootstrapping solutions for usage of KAUSF or KSEAF as AKMA root key

### 5.21.1 Introduction

This solution addresses key issue #10 by introducing a procedure for deciding which key to use for AKMA root key depending on the deployment and local configuration.

This solution combines parts of solutions #15 and #16. With respect to the presence of an AAuF it has three options:

1) Home Network Option: In this option, there is no AAuF in the serving network that plays a role in this solution;

2) Serving Network Option: In this option, there is only an AAuF in the serving network;

3) Combined option: In this option, there is an AAuF in both the serving network and the home network.

Solution #15 and solution #16 are similar in that they introduce a new element AKAF and AKRS respectively that functions roughly similar between the two solutions. In this solution, the term AKRS is used.

### 5.21.2 Solution details

#### 5.21.2.1 Generic procedure

This generic procedure for solution works as follows:

1) The UE initiates an initiation procedure by contacting an AAuP according to clause 5.2.2.2.1. The AAuP responds signalling AKMA compatibility. Upon reception of the trigger, the UE initiates an AKMA Established Key Use Procedure by sending a service request to the AAuF including the UE Identity and the UE's preference for which key to use.

2) Upon reception of the service request, the AAuF decides whether the KAKMA should be derived from KSEAF or KAUSF and sends a request for a KAKMA based on a particular key to the AKRS.

3) The AKRS generates the KAKMA and sends the key together with a random and an XRES to the AAuF.

4) The AAuF authenticates the UE, and if successful sends the UE the necessary information for AKMA (which key was used as a base for KAKMA, the temporary identity, and the validity timer).

The procedure is shown in the figure 5.21.2.1-1 below:

UE

AAuF

AKRS

Service request (SUCI/SUPI, Key Flag)

Established Key Use request (UE ID, Key Flag)

Decide whether key reusage is allowed

Fetch key from storage, derive KAKMA

Key Use response (KAKMA, RAND, XRES)

Authentication  
Temporary ID and validity timer exchange

Figure 5.21.2.1-1: Generic procedure

Each of the options differs in how the decision on which key to use is taken. Depending on the decisions and access to keys, KAKMA will be derived in one of the following ways:

1) KAKMA = KDF(KAUSF, …)

2) KAKMA = KDF( KDF(KAUSF, Serving Network Name), …)

3) KAKMA = KDF(KSEAF, …)

#### 5.21.2.2 Home Network Option

In this option, it is assumed that the AAuF is connected to the AKRS in the home network and that this AKRS has access to (the relevant keys derived from) KAUSF. For this solution it is not relevant whether this is implemented using a push mechanism according to solution #15 or a pull mechanism according to solution #16.

In this case, the AAuF can choose between using a KAKMA based on KAUSF directly or based on KSEAF. The AAuF can decide based on criteria like:

1) Whether the UE is roaming and where;

2) Whether the service is located in the country where the UE is roaming / serving network;

3) Whether there is a network element in the serving network that can receive the derived key;

4) Local configuration.

#### 5.21.2.3 Serving Network Option

In this option, it is assumed that the AAuF is connected to the AKRS in the serving network and that this AKRS has access to (the relevant keys derived from) KSEAF. For this solution it is not relevant whether this is implemented using a push mechanism according to solution #15 or a pull mechanism according to solution #16.

In this case, the AAuF cannot choose and will instruct the AKRS to derive a key from KSEAF.

#### 5.21.2.4 Combined Option

In this option, it is assumed that there is an AAuF is connected to the AKRS in the home network and one connected to the AKRS in the serving network. For this solution it is not relevant whether this is implemented using a push mechanism according to solution #15 or a pull mechanism according to solution #16. It is assumed that the AAuF in the serving network can take the role of a proxy for the AAuF in the home network.

In this case, the Home AAuF can choose between using a KAKMA based on KAUSF directly or based on KSEAF. The AAuF can decide based on criteria like:

1) Whether the UE is roaming and where;

2) Whether the service is located in the country where the UE is roaming / serving network;

3) Whether there is a network element in the serving network that can receive the derived key;

4) Local configuration.

### 5.21.3 Evaluation

In the serving network option, the authentication is performed between the UE and the serving network without involving the home network. Consequently, the home network has no control over the authentication, which may pose problems with charging and liability. Therefore, this option is not preferred.

In the home network option, an authentication is performed between the UE and the home network based on KSEAF or KAUSF. If the KSEAF is used, the serving network could pose as the AUSF because the serving network knows the KSEAF. Also, this condition leads to a situation of loss of home network control over the authentication. The solution lacks a mitigating measure for this attack. Therefore, this option is not preferred either.

## 5.22 Solution #22: Key freshness in AKMA

### 5.22.1 Introduction

This solution addresses key issue #16 Key freshness in AKMA.

It is assumed that the derived sub-keys (i.e., application key KAF) is not exceed the lifetime of the anchor key KAKMA. When the key lifetime of KAF is expired, the application key KAF is renegotiated.

### 5.22.2 Solution details

Once the UE has been successfully authenticated by the Anchor Function, the UE has the necessary keying material to establish secure communication with any AKMA AF. Once the UE and the NAF have established that they want to use AKMA then every time the UE wants to interact with an NAF the following steps are executed as depicted in Figure 5.22.2-1.



Figure 5.22.2-1: Usage procedure

1. The UE starts communication with AKMA AF. The UE supplies the temporary identifier to the AKMA AF. The temporary identifier is generated in the procedure of bootstrapping authentication of AKMA and used to bind the subscriber identity to the keying material.

2. The AKMA AF checks if the KAF lifetime is expired, if so the AKMA AF requests key material corresponding to the temporary identifier supplied by the UE. The AKMA AF supplies the temporary identifier and AKMA AF identifier to the Anchor Function.

3. The Anchor Function checks the lifetime of KAKMA. If the KAKMA is expired or the remaining lifetime of KAKMA shorter than the lifetime of KAF going to generated, the Anchor Function shall trigger to renegotiate a new KAKMA. Otherwise, Anchor Function derives the key KAF from the key KAKMA. KAF is computed as KAF = KDF (KAKMA, UE identity, AKMA AF identifier, Nonce), where KDF is the key derivation function and Nonce is a random number generated by Anchor Function.

4. The Anchor Function supplies to AKMA AF the requested key KAF, as well as the Nonce and the lifetime of KAF.

5. The Anchor Function stores the key KAF and lifetime of KAF.

6. The AKMA AF supplies the Nonce and lifetime of KAF to the UE. The AKMA AF calculates a MAC using KAF to protect the integrity of the message.

7. The UE derives the key KAF from the key KAKMA and the key derivation parameters according to step 3. Then the UE checks the MAC using KAF.

### 5.22.3 Evaluation

This solution fulfils the security requirement of KI #16.

This solution binds the KAKMA and KAF together by checking the lifetime of KAKMA when a new KAF is derived.

The advantage of this solution is the Anchor function gurantees the lifetime of KAKMA is longer than the KAF when a new KAF is derived by checking the expiry time of the KAKMA.

The disadvantage is that the check binds the lifetime of KAKMA and KAF together, and the value of binding the lifetime is not clear.

## 5.23 Solution #23: Implicit bootstrapping using NEF as the AKMA Anchor Functions

### 5.23.1 Introduction

This solution addresses Key Issues #1, #2, #3, #4, #5.

### 5.23.2 Solution details

#### 5.23.2.1 Architecture

This solution introduces NEF as the AKMA Anchor Function (AKAF). The NEF is the anchor that provides key management services to 3rd party AKMA Application functions (AApF).



Figure 5.23.2.1-1 NEF as the AKMA Anchor Function

The NEF provides Northbound API(s) for 3rd party applications that intend to use PLMN's AKMA service to setup application layer security between the application and the UE. These APIs are used by an AApF to obtain AF-specific keying material to be used between the UE and the AApF.

There is no separate authentication of the UE to support AKMA functionality. Instead, it reuses the 5G primary authentication for AKMA purposes. The AKMA anchor key, KAKMA, is derived by the AUSF and the UE as part of the 5G primary authentication run.

AApF (AKMA Application Function) interacts with NEF for AKMA application specific keys. The NEF obtains the AKMA anchor key KAKMA from the AUSF.

#### 5.23.2.2 Procedures



Figure 5.23.2.2-1: Implicit bootstrapping of AApF using NEF as the AKMA Anchor Function

1. At the end of the primary authentication run, the UE and the AUSF generate the AKMA Anchor Key (KAKMA) and the associated Key Identifier from KAUSF.

2. The UE starts communication with the AApF with a session establishment request.

The UE derives AApF specific key from KAKMA before it begins communicating with the application function. It includes AKMA Key Identifier, AApF Identifier. in the request.

NOTE 1: AApF Id may be its public hostname that the UE has used to access AApF.

3-4. The AApF checks if it has the necessary pre-shared key for the requesting UE. If not, it'll invoke the Key Request API to obtain AApF specific key from the NEF.

The AApF includes the key identifier, and its own Id in the API Request.

5-6. The NEF generates AApF specific key from KAKMA and responds back to AApF with the AApF key and its lifetime.

NOTE 2: Derivation of KAKMA and associated Key Identifier, and AApF specific key (including whether additional information regarding the AF is required for deriving AApF specific key) are not detailed in this solution, which could be designed during the normative work.

Since AKMA keys are based on KAUSF from primary authentication run, the AKMA keys can only be refreshed by running a fresh primary authentication. This implies that the AKMA key lifetime(s) are higher than the time interval between primary authentication runs.

### 5.23.3 Evaluation

This solution addresses key issues #1, #2, #3, #4, #5.

The solution reuses NEF, which is the standard mechanism to expose network services in 5G, as the anchor for providing PLMN anchored key management service to 3rd party applications. One of the advantages of this solution is the using of existing network function, as well as the interface between NEF and application functions, which avoids introducing a new 5GC NF, and helps operators to manage and secure AKMA service exposure. However, this means NEF has to handle key derivation, thus new functionalities of NEF have to be defined.

The solution reuses KAUSF, generated from primary authentication of the UE, to derive AKMA specific keys. There is no separate authentication of the UE for AKMA key management services.

There is another benefit of this solution. During AKMA procedures, UE only need to interact with AKMA Application Function in order to use AKMA service, without the necessity to interact with any of network functions in 5GC. Thus, this solution is applicable to all kinds of IoT devices whatever application protocols (HTTP, MQTT, CoAP, etc.) they are using.

## 5.24 Solution #24: AKMA push

### 5.24.1 Introduction

This solution addresses key issue #17 AKMA push, and proposes a new mechanism allowing AApF to push an information securely to the UE.

The GBA push feature specified in TS 33.223 [13] is a mechanism to bootstrap the security between the NAF and the UE, without forcing the UE to contact the BSF to initiate the bootstrapping, the security of which is based on the GBA AKA mechanism.

For the AKMA mechanism, different authentication procedures will be defined, e.g., EAP AKA', 5G AKA, etc. On the other hand, the existing 5G security context (i.e. KAUSF) can also be used for AKMA security. Hence, AKMA push takes the above scenarios into consideration. Therefore, three ways for AKMA key generation during the AKMA push are required, such as the existing KAUSF, or authentication vector of the EAP AKA' or 5G AKA.

### 5.24.2 Solution details

#### 5.24.2.1 Architecture and reference points

The AKMA push architecture includes Network Functions:

- The AKMA Authentication Function (AAuF), and

- The AKMA Application Function (AApF).

The AAuF is the authentication anchor that performs the UE authentication service. In the AKMA push, AAuF is responsible for retrieving the KAUSF from the AUSF, and AKMA push information (AKMA-PI) generation.

The AAuF interacts with the AUSF and the AApF using Service-Based Interfaces.

The AApF is the application function that provides service for the UE. The AApF interacts with the AAuF to retrieve the push information from the AAuF, and establishes the security association with the UE based on the AKMA push.

The AApF interacts with the UE using the interface Ap1, which is dedicated for the push information and message transmission.



Figure 5.24.2.1-1: AKMA reference architecture

#### 5.24.2.2 Potential Procedures

##### 5.24.2.2.1 Initiation

The high level of this solution is proposed as follows:

- AApF sends an AKMA push request to the AAuF for the AKMA-PI

- AAuF asks the AUSF for the AAuF key, then generates and sends the AKMA-PI to the AAPF

- AApF sends the AKMA push message to the UE, to establish the AKMA SA between them.



Figure 5.24.2.2.1-1: Initiation procedure

Aprecondition for use of AKMA Push is that the UE is registered with the AApF for the intended service. The AApF knows the identity of the subscriber.

Processing and message flow:

1. The AApF sends the AKMA push request to the AAuF, including the AApF ID, identity of the subscriber, e.g., GPSI.

2. Upon receiving the request from the AApF, the AAuF send the AKMA security context request to the UDM, including the GPSI.

3. The UDM returns back the SUPI. If the UE is already registered into the network, the UDM also sends back the AUSF address, where the AUSF address indicate the serving AUSF for the UE.

4. AAuF forwards the SUPI and AAuF ID to the AUSF according to the AUSF address. If there is no AUSF address, the AAuF select an AUSF based on the local policy.

5. Upon receiving the AKMA security request from the AAuF, AUSF generates the AKMA key Kaauf with the following cases:

Case A: If the identity of the subscriber is the SUPI, AUSF obtains the key Kausf based the SUPI, then generates the Kaauf based on the KAUSF and AAuF ID.

Case B: If the identity of the subscriber is the SUPI, and there is no Kausf related with the SUPI, AUSF sends the SUPI to the UDM. UDM/ARPF shall choose the authentication method and generate the authentication vector, based on the subscription data of the SUPI. Then AUSF obtains the KAUSF, then generates the Kaauf based on the KAUSF and AAuF ID.

6. For Case A, the AUSF sends the Kaauf to the AAuF. For Case B, the AUSF sends the RAND and AUTN of EAP-AKA' AV or 5G HE AV, the authentication method indicator and Kaauf to the AAuF. The detail of EAP-AKA' AV and 5G HE AV refers to TS 33.501 clause 6.1. The authentication method indicator here indicates which authentication method is chosen for AKMA-PI.

7. The AAuF generates the requested AApF key Kaapf according to the provided AApF ID and Kaauf. AAuF generates the AKMA-PI with the following cases. Here AKMA-PI also includes a context indicator indicating whether the existing KAUSF is used or not, and includes an authentication method indicator.

Case A: AAuF randomly select a random number RAND, and generates the MAC using the RAND and KAUSF. Then AAuF generates the AKMA-PI similar as the GBA PI generation specified in TS 33.223 [13], using the RAND and MAC, where the MAC is here used instead of the AUTN.

Case B: AAuF generates the AKMA-PI similar as the GBA PI generation specified in TS 33.223 [13].

8. The AAuF sends the AKMA-PI, and AApF key to the AApF.

9. The AApF stores the received information together with other user information in an AKMA SA. The AKMA SA include the AApF key, identifier of the user, etc. details of AKMA SA is the same as NAF SA defined in TS 33.223 [13].

10. The AApF then forwards the AKMA-PI to the UE over Ap1 using the selected transport mechanism and the given transport address.

11. The UE receives the message containing the AKMA-PI.

If the context indicator indicates the existing the KAUSF is used for AKMA-PI, the UE first finds out the KAUSF, then generates the KAUSF, and verifies the MAC.

Otherwise, the UE verifies the RAND and AUTN based on the authentication method indicated by the authentication method indicator.

If the MAC, or the RAND and AUTN is successfully verified, the following procedure is the same as TS 33.223 [13]. The UE stores the AKMA SA.

The UE and NAF are now ready to use the established AKMA SA.

### 5.24.3 Evaluation

The above solution addresses the requirements of key issue #17 on the AKMA push.

The solution reuses the existing security context (i.e. KAUSF), or primary authentication for AKMA push to securely generate the AKMA push information for the UE, rather than performing a separate authentication for AKMA.

From the privacy protection aspect, the solution uses the GPSI for UE identification, which is sent from AApF to AAuF, to assure that SUPI will be never sent out of the 5G network.

NOTE: The necessity of AKMA push is not clarified in this solution, thus this feature will not progress to conclusions in the present document.

## 5.25 Solution #25: Key lifetimes

### 5.25.1 Introduction

This solution addresses the Key issue #12.

### 5.25.2 Solution details

#### 5.25.2.1 KAKMA lifetime

As concluded in clause 7.1 of the present document, the KAKMA will be derived from KAUSF (implicit bootstrapping). Since KAUSF does not have a lifetime, the KAKMA will not automatically inherit a lifetime.

There are some options for the anchor key (KAKMA) lifetimes:

**1) Implicit lifetime:** The KAKMA will be valid until the next primary authentication is performed, in which case the KAKMA is replaced after a successful new authentication or removed after an unsuccessful one.

a) A revocation procedure might be needed, that revokes the current KAKMA key and its derived application keys upon an unsuccessful re-authentication.

b) An explicit key refresh procedure is not needed in this case.

**2) Explicit lifetime:** A life time is specified for the KAKMA based on some configuration parameter. The following needs consideration:

a) A refresh procedure is needed in order to get a fresh KAKMA based on the same KAUSF in case the KAKMA lifetime runs out before a new primary authentication is run.

b) A policy is needed to handle the case where a new primary authentication is performed within that lifetime of a KAKMA. In such a case, shall the current KAKMA be replaced or shall it remain until its lifetime expires (provided that the new authentication is successful)?

c) A revocation procedure might be needed, that revokes the current KAKMA key and its derived application keys upon an unsuccessful re-authentication.

To avoid the need for a refresh procedure for anchor keys, it is proposed that the implicit lifetime is used for KAKMA. This is also in line with how the other anchor key based on KAUSF, KSEAF, is handled.

#### 5.25.2.2 Application key lifetime

For application keys, the same options for key lifetimes are available. The consequences are however different:

**- Implicit lifetime:** The KAF will be valid until a new anchor key (KAKMA) is derived in which case all application keys are replaced.

a) Since the generation of application keys in not initialized by the anchor function, this implies some signalling is needed from the Anchor function to the Application server and/or UE (or within the UE) when there is a new anchor key.

**- Explicit lifetime:** A lifetime is specified for the KAF based on some configuration parameter or operator policy. The following needs consideration:

a) A refresh procedure is needed in order to get a fresh KAF based on the same KAKMA in case the KAF lifetime runs out before a new KAKMA is derived (unless they both have explicit and equal lifetimes)

b) There is a requirement stating that the lifetime of an application key shall not exceed the lifetime of the anchor key.

i) But if the anchor key has implicit lifetime, there is need for some signalling from the Anchor function to the Application function to avoid that a child key is used after the parent key is has been replaced.

ii) The signalling above can be avoided if the application key continues to be used until its lifetime runs out. This would contradict the requirement above but provide a smoother solution.

It is proposed that explicit lifetimes are used for application keys to avoid the extra signalling mentioned above for implicit lifetimes.

It is proposed that application keys can continue to be used until their lifetimes expire even when there is a new anchor key established. When the application key lifetime expires, a new application key is established using the new anchor key. As mentioned above, this property does not meet the requirement that states that the lifetime of the derived sub-keys shall not exceed the lifetime of the anchor key. However, since there is no explicit lifetime of the anchor key there is no strict security need to change the application key when the anchor key is changed.

A separate application key refresh procedure is needed but not provided by this solution since the study includes several potential solutions (e.g. solution #22) for application key refresh.

If a new primary authentication is performed that does not result in a new anchor key, the application keys might need to be revoked. A potential solution for key revocation that could be used for this solution is Solution #14.

### 5.25.3 Evaluation

This solution addresses key issue #12 which has 3 requirements.

Requirement #1 is fulfilled by letting the anchor key having an implicit lifetime and the application keys an explicit lifetime.

To avoid extra signalling when an anchor key is replaced, application key can still be used until the explicit lifetime expires. Thus, the requirement #2 is not addressed. However, since there is no explicit lifetime of the anchor key there is no strict security need to change the application key when the anchor key is changed

Requirement #3 is not applicable for anchor keys with implicit lifetimes, only for application keys. A solution for application key refresh is provided in solution #22.

A procedure for revocation of anchor key and application keys is needed in case a renewed primary authentication fails. A solution for key revocation is provided in solution #14.

# 6 Evaluation and conclusion

## 6.1 Evaluation and conclusion on architecture and authentication procedures

1) It is recommended to use the idea of solution #15, #19 and #23 of implicit bootstrapping as the basis of AKMA authentication procedure. In details, reusing KAUSF（as described in solution #19, #23 and the child option in solution #15）is recommended to be the basis of normative work.

2) It is recommended to use the AKMA architecture introduced in solution #19 and solution #23. The decision of the realization of the AKMA anchor function (AUSF or NEF or a new NF) will be done in the normative work.

## 6.2 Evaluation and conclusion on key management

### 6.2.1 Evaluation and conclusion on Key lifetimes (Key issue #12)

Solution #25 is recommended for normative work for key lifetimes. This solution recommends using implicit lifetime for the anchor key and explicit lifetimes for the application keys.

It is proposed that application keys can continue to be used until their lifetimes expire even when there is a new anchor key established. When the application key lifetime expires, a new application key is established using the new anchor key. This property does not meet the requirement in the key issue that states that the lifetime of the derived sub-keys shall not exceed the lifetime of the anchor key. However, since there is no explicit lifetime of the anchor key there is no strict security need to change the application key when the anchor key is changed.

Because of implicit lifetimes for anchor keys, the requirement in the key issue about key renegotiation is only applicable for application keys, not anchor keys.

A notification procedure is needed that enables the AUSF to notify the Anchor function that the KAUSF is replaced or is no longer valid. The anchor function can then choose to delete the anchor key and/or notify the application functions.

Details on the interface between AUSF and the Anchor function is left for normative work.

Details on the notification between the Anchor function and the Application functions, and any subsequent key management is left for normative work.

Details on notification and subsequent key management within the UE is left for normative work.

### 6.2.2 Evaluation and conclusion on Secure communication between UE and application server (Key issue #6)

Secure communication between UE and application server can be achieved in multiple ways. For operators wanting to provide security services to verticals, it is recommended to provide informative text with example profiles describing methods for obtaining secure communication in addition to the AKMA normative text.

## 6.3 Evaluation and conclusion on interfaces and protocols

For key issue #2, it is recommended to use the idea of solution #15, #19, #23 (i.e. to only have an AKMA related user plane connection between UE and Application Functions) to address the transport independent protocol issue.

Note 1: The AKMA Application Function is replaced by Application Function here.

NOTE 2: Protection of AKMA interfaces is left for normative work, which depends on the specific AKMA architecture specified in the normative work.

## 6.4 Evaluation and conclusion on privacy

For key issue#5, it is recommended to use the idea of solution #19 and solution #23 of generating an identifier to bind the user identity to the keying material as the basis for normative work. Detailed design on how this identifier is generated and to which key it is associated will be done according to the AKMA procedures in normative work.

## 6.5 Evaluation and conclusion on API of AKMA in the UE

It is concluded that no normative work will be done regarding the API between AKMA bootstrapping client and AKMA app, but the potential changes to the UICC-ME interface achieving AKMA procedures need to be clarified in the normative work.

Annex A:  
Change history

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Change history** | | | | | | | |
| **Date** | **Meeting** | **TDoc** | **CR** | **Rev** | **Cat** | **Subject/Comment** | **New version** |
| 2019-12 | SA#86 | SP-191142 |  |  |  | EditHelp review and presented for approval | 2.0.0 |
| 2019-12 | SA#86 |  |  |  |  | Change control version | 16.0.0 |
| 2020-07 | SA#88E | SP-200375 | 0001 | - | F | Clean up of TR 33.835 | 16.1.0 |