3GPP TSG-SA WG3 Meeting #36
Shenzhen, China, November 23-26, 2004

Tdoc ⌘S3-041027

CR-Form-v7.1

CHANGE REQUEST

⌘ **33.220** **CR** **041** ⌘ **rev** **-** ⌘ Current version: **6.2.0** ⌘

*For **HELP** on using this form, see bottom of this page or look at the pop-up text over the ⌘ symbols.*

Proposed change affects: | UICC apps⌘ [X] ME [X] Radio Access Network [ ] Core Network [X]

Title:	⌘	Key derivation function
Source:	⌘	Nokia
Work item code:	⌘	SEC1-SC
Date:	⌘	16/11/2004
Category:	⌘	**B**
Release:	⌘	Rel-6

Use one of the following categories:
F *(correction)*
A *(corresponds to a correction in an earlier release)*
B *(addition of feature),*
C *(functional modification of feature)*
D *(editorial modification)*
Detailed explanations of the above categories can be found in 3GPP TR 21.900.

Use one of the following releases:
Ph2 (GSM Phase 2)
R96 (Release 1996)
R97 (Release 1997)
R98 (Release 1998)
R99 (Release 1999)
Rel-4 (Release 4)
Rel-5 (Release 5)
Rel-6 (Release 6)
Rel-7 (Release 7)

Reason for change:	⌘	SAGE has defined a key derivation function (KDF) in their LS to SA3 (S3-040914 updated in S3-040937). This CR implements the KDF to TS 33.220.
Summary of change:	⌘	Annex B contains the KDF as specified by SAGE with following difference: - the FC field is changed to variable length (i.e., to add the possibility for extending the range FC parameter values, and thus to have a possibilty for more than 256 instances of the algorithm). Subclauses 4.2.1 and 4.5.2 refer to Annex B. **NOTE:** As it is expected that the there will be substancial changes in 5.3.2 regarding the GBA_U key derivation, this section is not implemented regarding KDF implementation. Only the existing editor's notes are replaced by new ones.
Consequences if not approved:	⌘	The KDF is not specified.

Clauses affected:	⌘	2, 4.2.1, 4.5.2, 5.3.2, Annex B

Other specs affected:	⌘	Y	N		⌘
			X	Other core specifications	
			X	Test specifications	
			X	O&M Specifications	

Other comments:	⌘	

===== **BEGIN CHANGE** =====

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 TS 31.102: "3rd Generation Partnership Project; Technical Specification Group Terminals; Characteristics of the USIM application".

[2] 3GPP TS 33.102: "3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; 3G Security; Security architecture".

[3] Franks J., et al,: "HTTP Authentication: Basic and Digest Access Authentication", RFC 2617, June 1999.

[4] A. Niemi, et al,: "Hypertext Transfer Protocol (HTTP) Digest Authentication Using Authentication and Key Agreement (AKA)", RFC 3310, September 2002.

[5] 3GPP TS 33.221: "3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Generic Authentication Architecture (GAA); Support for Subscriber Certificates".

[6] T. Dierks, et al,: "The TLS Protocol Version 1.0", RFC 2246, January 1999.

[7] OMA: "Provisioning Content Version 1.1", Version 13-Aug-2003. Open Mobile Alliance.

[8] 3GPP TS 23.228: "3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; IP Multimedia Subsystem (IMS); Stage 2 (Release 6)".

[9] IETF RFC 3546 (2003): "Transport Layer Security (TLS) Extensions".

[10] 3GPP TS 31.103: "3rd Generation Partnership Project; Technical Specification Group Terminals; Characteristics of the IP Multimedia Services Identity Module (ISIM) application".

[11] 3GPP TS 23.003: "3rd Generation Partnership Project; Technical Specification Group Core Network; Numbering, addressing and identification".

[12] IETF RFC 3548 (2003): "The Base16, Base32, and Base64 Data Encodings".

[13] 3GPP TS 33.210: "3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; 3G Security; Network domain security; IP network layer security".

[14] IETF RFC 3588 (2003): "Diameter Base Protocol".

[15] IETF RFC 2104 (1997): "HMAC: Keyed-Hashing for Message Authentication".

[16] FIPS PUB 180-2 (2002): "Secure Hash Standard".

===== **BEGIN NEXT CHANGE** =====

4.2.1 Bootstrapping server function (BSF)

A generic Bootstrapping Server Function (BSF) and the UE shall mutually authenticate using the AKA protocol, and agree on session keys that are afterwards applied between UE and an operator-controlled Network Application Function

(NAF). The BSF shall restrict the applicability of the key material to a specific NAF by using ~~a suitable~~the key derivation procedure as specified in Annex B. The key derivation procedure may be used with multiple NAFs during the lifetime of the key material. The lifetime of the key material is set according to the local policy of the BSF. The generation of key material is specified in clause 4.5.2.

The BSF shall be able to acquire the GBA user security settings from the HSS.

===== BEGIN NEXT CHANGE =====

4.5.2 Bootstrapping procedures

When a UE wants to interact with a NAF, and it knows that the bootstrapping procedure is needed, it shall first perform a bootstrapping authentication (see figure 4.3). Otherwise, the UE shall perform a bootstrapping authentication only when it has received bootstrapping initiation required message or a bootstrapping negotiation indication from the NAF, or when the lifetime of the key in UE has expired (cf. subclause 4.5.3).

NOTE 1: The main steps from the specifications of the AKA protocol in TS 33.102 [2] and the HTTP digest AKA protocol in RFC 3310 [4] are repeated in figure 3 for the convenience of the reader. In case of any potential conflict, the specifications in TS 33.102 [2] and RFC 3310 [4] take precedence.

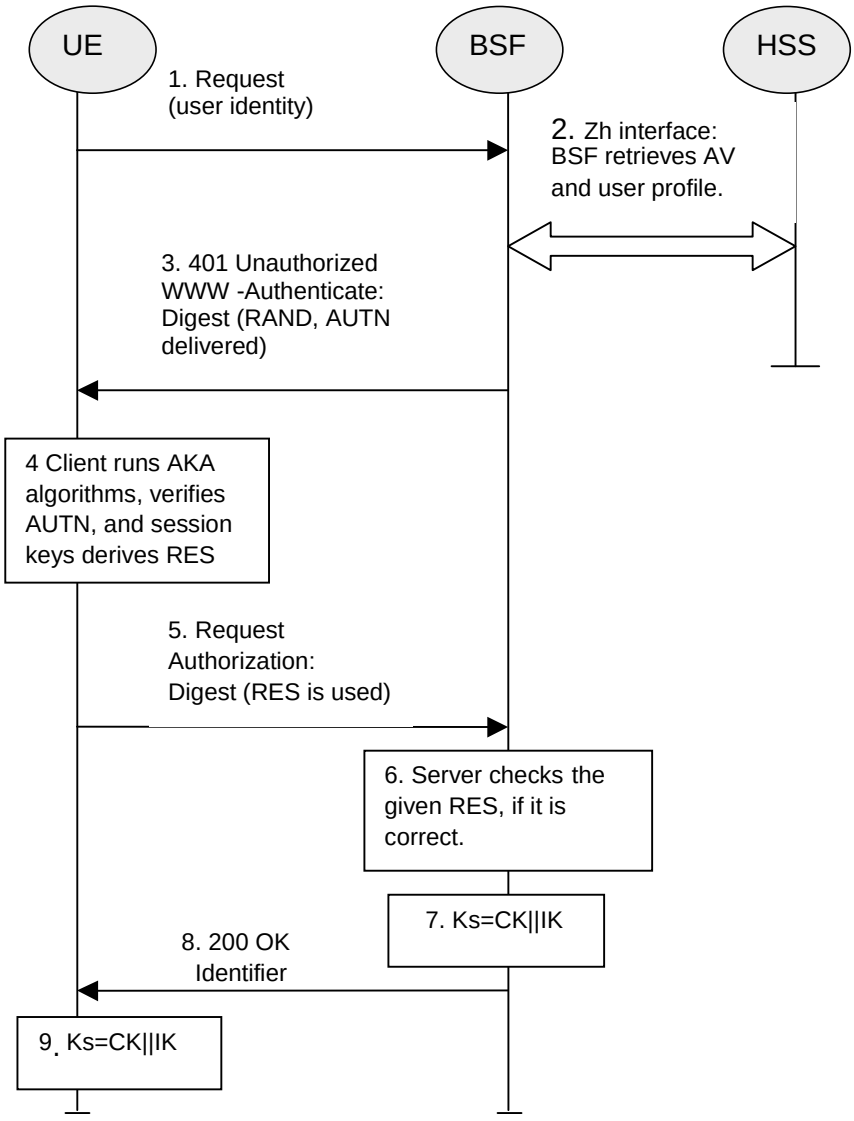

Figure 4.3: The bootstrapping procedure

1. The UE sends an HTTP request towards the BSF.

2. BSF retrieves the complete set of GBA user security settings and one or a whole batch of Authentication Vectors (AV, AV = RAND||AUTN||XRES||CK||IK) over the reference point Zh from the HSS.

3. Then BSF forwards the RAND and AUTN to the UE in the 401 message (without the CK, IK and XRES). This is to demand the UE to authenticate itself.

4. The UE checks AUTN to verify that the challenge is from an authorised network; the UE also calculates CK, IK and RES. This will result in session keys IK and CK in both BSF and UE.

5. The UE sends another HTTP request, containing the Digest AKA response (calculated using RES), to the BSF.

6. The BSF authenticates the UE by verifying the Digest AKA response.

7. The BSF generates key material Ks by concatenating CK and IK. The B-TID value shall be also generated in format of NAI by taking the base64 encoded [12] RAND value from step 3, and the BSF server name, i.e. base64encode(RAND)@BSF_servers_domain_name.

8. The BSF shall send a 200 OK message, including a B-TID, to the UE to indicate the success of the authentication. In addition, in the 200 OK message, the BSF shall supply the lifetime of the key Ks. The key material Ks is generated in UE by concatenating CK and IK.

9. Both the UE and the BSF shall use the Ks to derive the key material Ks_NAF during the procedures as specified in clause 4.5.3. Ks_NAF shall be used for securing the reference point Ua.

   Ks_NAF is computed as Ks_NAF = KDF (Ks, "gba-me" || RAND || IMPI || NAF_Id~~key derivation parameters~~), where KDF is ~~a suitable~~the key derivation function as specified in Annex B, and the key derivation parameters consist of the user's IMPI, the NAF_Id and RAND. The NAF_Id consists of the full DNS name of the NAF. KDF shall be implemented in the ME.

NOTE 2: To allow consistent key derivation based on NAF name in UE and BSF, at least one of the three following prerequisites shall be fulfilled:

(1) The NAF is known in DNS under one domain name (FQDN) only, i.e. no two different domain names point to the IP address of the NAF. This has to be achieved by administrative means.
This prerequisite is not specific to 3GPP, as it is necessary also under other circumstances, e.g. for TLS V1.0 without use of wildcard or multiple-name certificates.

(2) Each DNS entry of the NAF points to a different IP address. The NAF responds to all these IP addresses. Each IP address is tied to the corresponding FQDN by NAF configuration. The NAF can see from the IP address, which FQDN to use for key derivation.

(3) Ua uses a protocol which transfers the host name (FQDN of NAF as used by UE) to NAF (e.g. HTTP/1.1 with mandatory Host request header field). This requires the NAF to check the validity of the host name, to use this name in all communication with UE where appropriate, and to transfer this name to BSF to allow for correct derivation of Ks_NAF.
In case of a TLS tunnel this requires either multiple-identities certificates or the deployment of RFC 3546 [9] or other protocol means with similar purpose.

~~Editor's note: The definition of the KDF is left to ETSI SAGE and is to be included in the Annex B of the present specification.~~

The UE and the BSF shall store the key Ks with the associated B-TID for further use, until the lifetime of Ks has expired, or until the key Ks is updated.

===== **BEGIN NEXT CHANGE** =====

5.3.2 Bootstrapping procedure

The procedure specified in this clause differs from the procedure specified clause 4.5.2 in the local handling of keys and Authentication Vectors in the UE and the BSF. The messages exchanged over the Ub reference point are identical for both procedures.

When a UE wants to interact with a NAF, and it knows that the bootstrapping procedure is needed, it shall first perform a bootstrapping authentication (see figure 5.1). Otherwise, the UE shall perform a bootstrapping authentication only

when it has received bootstrapping initiation required message or a bootstrapping renegotiation indication from the NAF, or when the lifetime of the key in UE has expired (see clause 5.3.3).

NOTE: The main steps from the specifications of the AKA protocol in TS 33.102 [2] and the HTTP digest AKA protocol in RFC 3310 [4] are repeated in figure 5.1 for the convenience of the reader. In case of any potential conflict, the specifications in TS 33.102 [2] and RFC 3310 [4] take precedence.

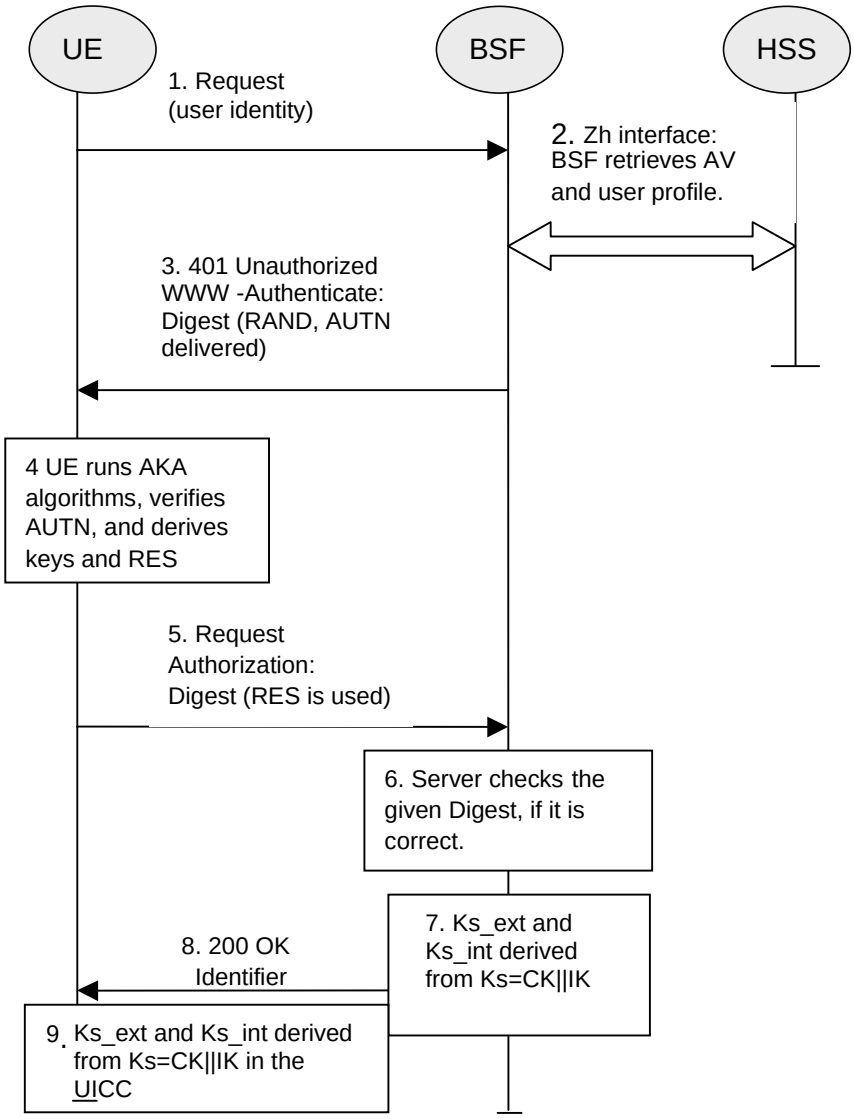

Figure 5.1: The bootstrapping procedure with UICC-based enhancements

1. The ME sends an HTTP request towards the BSF.

2. The BSF retrieves the complete set of GBA user security settings and one or a whole batch of Authentication Vectors
(AV, AV = RAND||AUTN||XRES||CK||IK) over the Zh reference point from the HSS. The BSF can then decide to perform GBA_U, based on the user security settings (USSs). In this case, the BSF proceeds in the following way:

- BSF computes MAC* = MAC SHA-1(IK1) (where IK= IK1|| IK2 and * is a exclusive or as described in TS 33.102 [2])

Editor's note: The exact format of the MAC modification function is to be reviewed. The output of SHA-1 needs to be truncated to exact amount of bits needed (64 bits).

The BSF stores the XRES after flipping the least significant bit.

3. Then BSF forwards the RAND and AUTN* (where AUTN* = SQN ⊕ AK || AMF || MAC*) to the UE in the 401 message (without the CK, IK and XRES). This is to demand the UE to authenticate itself.

4. The ME sends RAND and AUTN* to the UICC. The UICCcalculates IK and MAC (by performing MAC= MAC* ⊕ SHA-1(IK1)). Then the UICC checks AUTN(i.e. SQN ⊕ AK || AMF || MAC) to verify that the challenge is from an authorised network; the UICC also calculates CK and RES. This will result in session keys CK and IK in both BSF and UICC.

5. The UICC then applies a suitable key derivation function h1 to Ks, which is the concatenation of CK and IK, and possibly further h1-key derivation parameters to obtain two keys, Ks_ext and Ks_int, each of length 128 bit, i.e. h1(Ks, h1 key derivation parameters) = Ks_ext || Ks_int (see also figure 5.2). The UICC then transfers RES (after flipping the least significant bit) and Ks_ext to the ME and stores Ks_int/ks_ext on the UICC.

~~Editors' Note: The definition of the h1 is left to ETSI SAGE and is to be included in the Annex B of the present specification.~~

Editors' Note: The key derivation details for Ks_int and Ks_ext need to be finalized before the key derivation function in Annex B is taken into use.

Editors' Note: The location (whether in the UICC or in the ME) of the storage of Ks_ext is ffs.

6. The ME sends another HTTP request, containing the Digest AKA response (calculated using RES), to the BSF.

7. The BSF authenticates the UE by verifying the Digest AKA response.

8. The BSF generates the key Ks by concatenating CK and IK. Then the BSF applies the key derivation function h1 to Ks and possibly further h1-key derivation parameters to obtain two keys, Ks_ext and Ks_int, in the same way as the UICC did in step 5. The B-TID value shall be also generated in format of NAI by taking the base64 encoded [12] RAND value from step 3, and the BSF server name, i.e. base64encode(RAND)@BSF_servers_domain_name.

9. The BSF shall send a 200 OK message, including the B-TID, to the UE to indicate the success of the authentication. In addition, in the 200 OK message, the BSF shall supply the lifetime of the keys Ks_ext and Ks_int, The lifetimes of the keys Ks_ext and Ks_int shall be the same.

10. The BSF shall use the keys Ks_ext and Ks_int to derive the NAF-specific keys Ks_ext_NAF and Ks_int_NAF, if requested by a NAF over the Zn reference point. Ks_ext_NAF and Ks_int_NAF are used for securing the Ua reference point. The UE shall use the key Ks_ext to derive the NAF-specific key Ks_ext_NAF, if applicable. The UICC shall use the key Ks_int to derive the NAF-specific key Ks_int_NAF, if applicable.

    Ks_ext_NAF is computed as Ks_ext_NAF = h2 (Ks_ext, h2-key derivation parameters), and Ks_int_NAF is computed in the UICC as Ks_int_NAF = h2 (Ks_int, h2-key derivation parameters), where h2 is a suitable key derivation function, and the h2-key derivation parameters include the user's IMPI, the NAF_Id and RAND. The NAF_Id consists of the full DNS name of the NAF.

~~Editors' Note: The definition of the h2 is left to ETSI SAGE and is to be included in the Annex B of the present specification.~~

Editors' Note: The key derivation details for Ks_int_NAF and Ks_ext_NAF need to be finalized before the key derivation function in Annex B is taken into use.

NOTE: The NOTE 2 of clause 4.5.2 also applies here.

The ME, the UICC and the BSF store the keys Ks_ext and Ks_int together with the associated B-TID for further use, until the lifetime of Ks_ext and Ks_int has expired, or until the keys Ks_ext and Ks_int are updated.

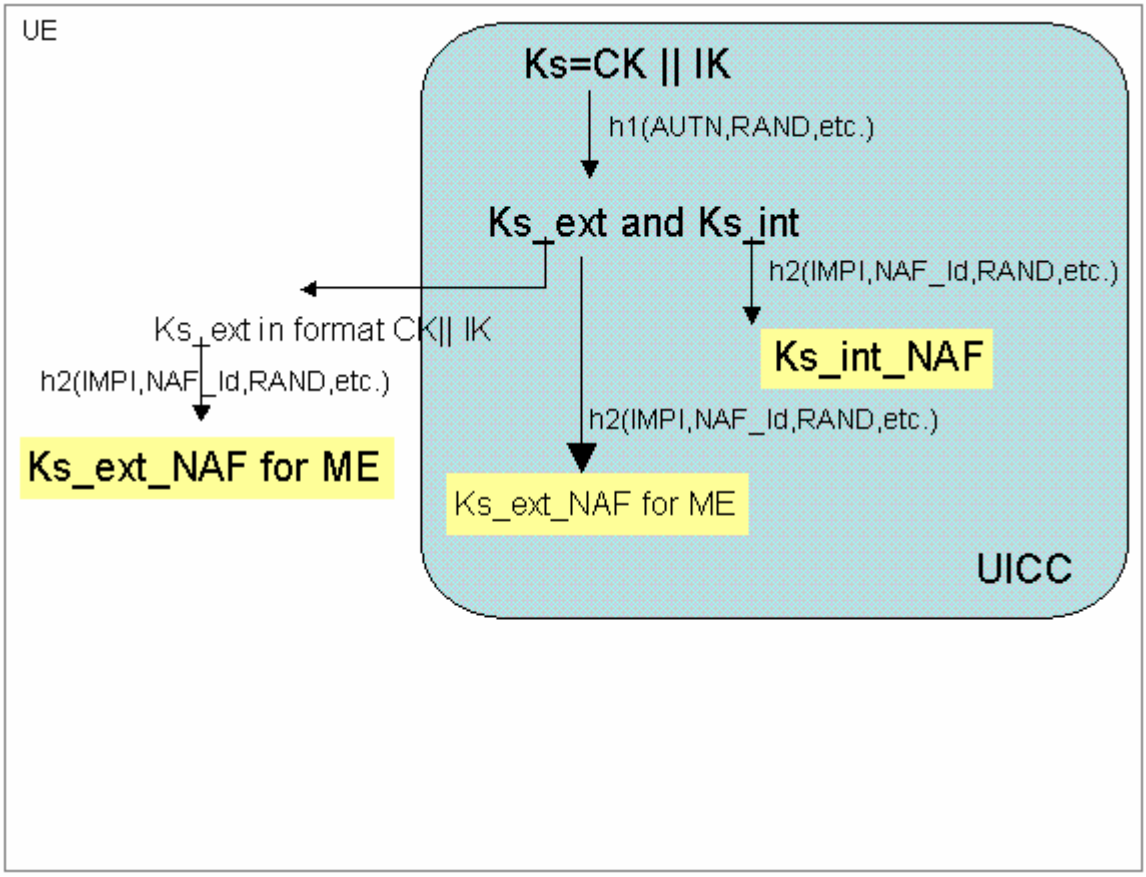

Figure 5.2: Key derivation for GBA-aware UICC when GBA-run was triggered

===== BEGIN NEXT CHANGE =====

Annex B (normative): Specification of the key derivation function KDF

~~Editor's note: The definition of the KDF and the possible inclusion of further key derivation parameters is left to ETSI SAGE.~~

B.1 Introduction

This annex specifies the key derivation function (KDF) that is used in the NAF specific key derivation in both GBA (i.e., GBA_ME) and GBA_U. The key derivation function defined in the annex takes the following assumptions:

1. the input parameters to the key derivation functions are octet strings - not bit strings of arbitrary length:
2. a single input parameter will have lengths no greater than 65535 octets.

B.2 Generic key derivation function

The input parameters and their lengths shall be concatenated into a string S as follows:

1. The length of each input parameter in octets shall be encoded into two-octet string:

   a) express the number of octets in input parameter Pi as a number l in the range $0 \leq l \leq 65535$.

   b) Li is then a two-octet representation of the number l, with the most significant bit of the first octet of Li equal to the most significant bit of l, and the least significant bit of the second octet of Li equal to the least significant bit of l,

   Example: If Pi contains 258 octets then Li will be the two-octet string 0x01 0x02.

2. String S shall be constructed from n input parameters as follows:

   S = FC || P0 || L0 || P1 || L1 || P2 || L2 || P3 || L3 ||... || Pn || Ln

   where

   FC is single octet used to to distinguish between different instances of the algorithm,

   P0 is a static ASCII-encoded string,

   L0 is the two octet representation of the length of the P0,

   P1 ... Pn are the n input parameters, and

   L1 ... Ln are the two-octet representations of the corresponding input parameters.

3. The final output, i.e., the derived key is equal to HMAC-SHA-256 (as specified in [15] and [16]) computed on the string S using the key Key:

   derived key = HMAC-SHA-256 (Key , S)

B.3 NAF specific key derivation in GBA and GBA_U

In GBA and GBA_U, the input parameters for the key derivation function shall be the following:

- FC = 0x01,
- P1 = RAND,

- L1 = length of RAND is 16 octets (i.e., 0x00 0x10),
- P2 = IMPI,
- L2 = length of IMPI is variable (not greater that 65535),
- P3 = NAF_ID, and
- L3 = length of NAF_ID is variable (not greater that 65535).

In the key derivation of Ks_NAF as specified in clause 4 and Ks_ext_NAF as specified in clause 5,

- P0 = "gba-me" (i.e., 0x67 0x62 0x61 0x2d 0x6d 0x65), and
- L0 = length of P0 is 6 octets (i.e., 0x00 0x06).

In the key derivation of Ks_int_NAF as specified in clause 5,

- P0 = "gba-u" (i.e., 0x67 0x62 0x61 0x2d 0x75), and
- L0 = length of P0 is 5 octets (i.e., 0x00 0x05).

The key Key to be used in key derivation shall be:

- Ks (i.e., CK || IK concatenated) in normal GBA as specified in clause 4,
- Ks_ext in GBA_ME as specified in subclause B.4, and
- Ks_int in GBA_U as specified in subclause B.4.

NOTE: In the specification this function is denoted as:
Ks_NAF = KDF (Ks, "gba-me" || RAND || IMPI || NAF_Id),
Ks_ext_NAF = KDF (Ks_ext , "gba-me" || RAND || IMPI || NAF_Id), and
Ks_int_NAF = KDF (Ks_int , "gba-u" || RAND || IMPI || NAF_Id).

Editor's note: If the key derivation optimisations related to GBA_U that are described in S3-040776 are adapted then Ks_ext and Ks_int keys are not used and the Ks key (i.e., CK || IK) is used as the key in the Ks_ext_NAF and Ks_int_NAF derivation:
Ks_ext_NAF = KDF (Ks , "gba-me" || RAND || IMPI || NAF_Id), and
Ks_int_NAF = KDF (Ks , "gba-u" || RAND || IMPI || NAF_Id).

B.4 Middle key derivation in GBA_U

Editor's note: This section is not needed if the key derivation optimisations as described in S3-040776 are adapted.

Editor's note: RAND and IMPI as input parameters to middle key (i.e., Ks_int/Ks_ext) derivations are given as an example. The decision needs to be made by SA3 what are the suitable key derivation parameters.

In GBA_U, the middle result keys Ks_int and Ks_ext keys shall be derived using the key derivation function as follows:

- FC = 0x02,
- P1 = RAND,
- L1 = length of RAND is 16 octets (i.e., 0x00 0x10),
- P2 = IMPI, and
- L2 = length of IMPI is variable (not greater that 65535),

In the key derivation of Ks_ext as specified in clause 5,

- P0 = "ks-ext" (i.e., 0x6b 0x73 0x2d 0x65 0x78 0x74), and
- L0 = length P0 is 6 octets (i.e, 0x00 0x06).

In the key derivation of Ks_int,

- P0 = "ks-int" (i.e., 0x6b 0x73 0x2d 0x69 0x6e 0x74), and

- L0 = length of P0 is 6 octets (i.e., 0x00 0x06).

The key Key shall be Ks (i.e., CK and IK concatenated).

NOTE: In the specification this function is denoted as:
Ks_int = KDF (Ks , "ks-int" || RAND || IMPI), and
Ks_ext = KDF (Ks , "ks-ext" || RAND || IMPI).

===== END CHANGE =====