CR-Form-v7

CHANGE REQUEST

| ⌘ | **33.220** CR | **035** | ⌘ **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 ☐ Radio Access Network ☐ Core Network **X**

Title:	⌘	Complete the MAC modification for GBA_U
Source:	⌘	Siemens
Work item code:	⌘ SEC1-SC	**Date:** ⌘ 15/11/2004

Category:	⌘ **F**	**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:
 2 (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)

Reason for change:	⌘	- Correction of wrong symbols - Removal of editor's Note by proposing a hash truncation
Summary of change:	⌘	Complete the MAC modification for GBA_U
Consequences if not approved:	⌘	Incomplete specification

Clauses affected:	⌘	2, 5.3.2, New clauses 3.3 and 3.4

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

Other comments:	⌘	

** FIRST 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]   FIPS PUB 180-2 (2002): "Secure Hash Standard".

** END OF CHANGE ***

** NEXT CHANGE ***

3.3 Symbols

For the purposes of the present document, the following symbols apply:

|| Concatenation

⊕   Exclusive or

3.4 Conventions

All data variables in this specification are presented with the most significant substring on the left hand side and the least significant substring on the right hand side. A substring may be a bit, byte or other arbitrary length bitstring. Where a variable is broken down into a number of substrings, the leftmost (most significant) substring is numbered 0, the next most significant is numbered 1, and so on through to the least significant.
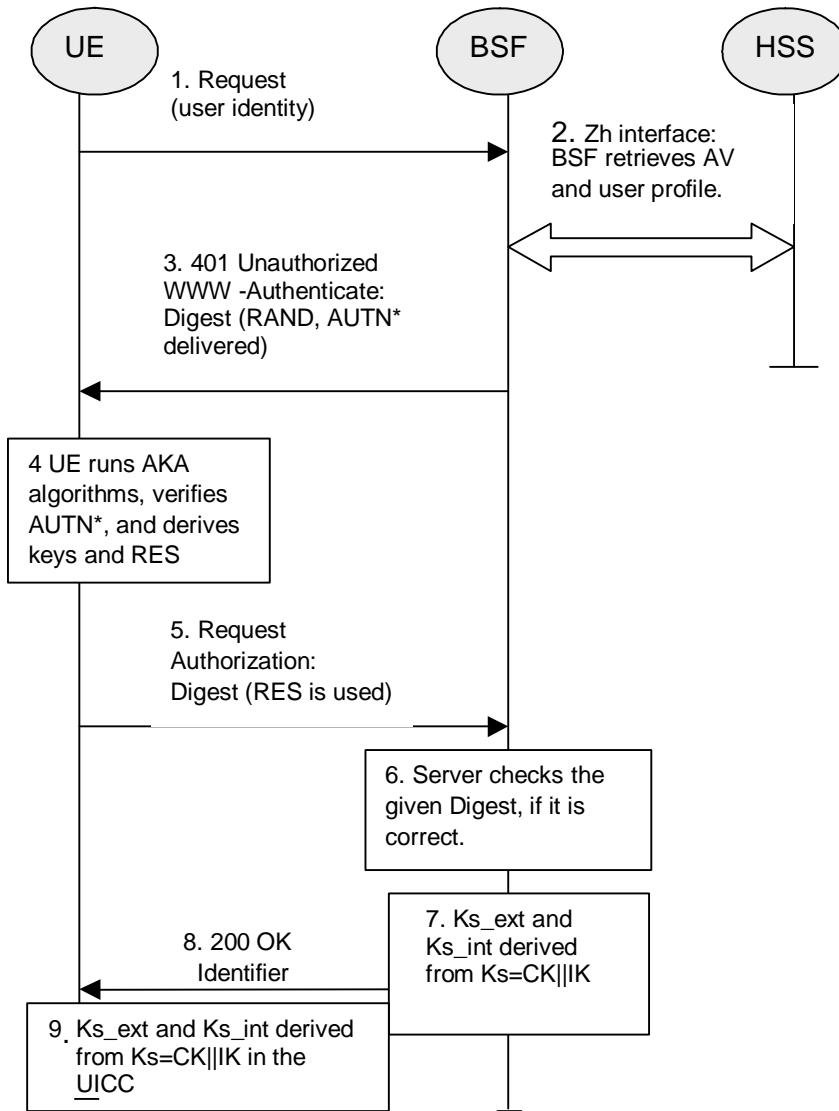
** END OF CHANGE ***

** LAST 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.
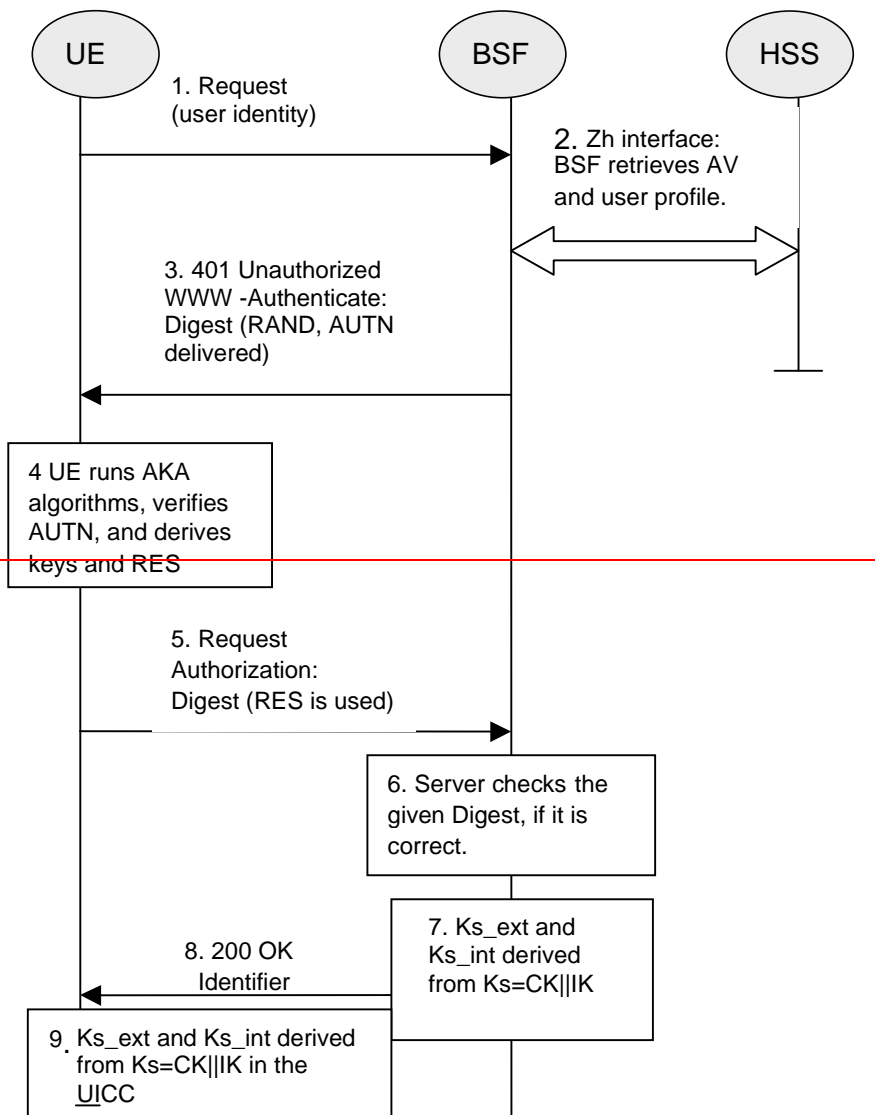
```
        ┌──────┐              ┌──────┐         ┌──────┐
        │  UE  │              │ BSF  │         │ HSS  │
        └──────┘              └──────┘         └──────┘
```

1. Request
(user identity)

2. Zh interface:
BSF retrieves AV
and user profile.

3. 401 Unauthorized
WWW -Authenticate:
Digest (RAND, AUTN*
delivered)

4 UE runs AKA
algorithms, verifies
AUTN*, and derives
keys and RES

5. Request
Authorization:
Digest (RES is used)

6. Server checks the
given Digest, if it is
correct.

7. Ks_ext and
Ks_int derived
from Ks=CK||IK

8. 200 OK
Identifier

9. Ks_ext and Ks_int derived
from Ks=CK||IK in the
UICC

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$\oplus$ Trunc(─SHA-1(IK1)$)$ (where IK= IK1‖ IK2 ~~and * is a exclusive or as described in TS 33.102 [2]~~)

NOTE: Trunc denotes that from the 160 bit output of SHA-1[15], the 64 bits numbered as [0] to [63] are used within the $\oplus$ operation to MAC.

~~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 $\oplus$ 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 UICC calculates IK and MAC (by performing MAC= MAC* ⊕ Trunc(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 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.

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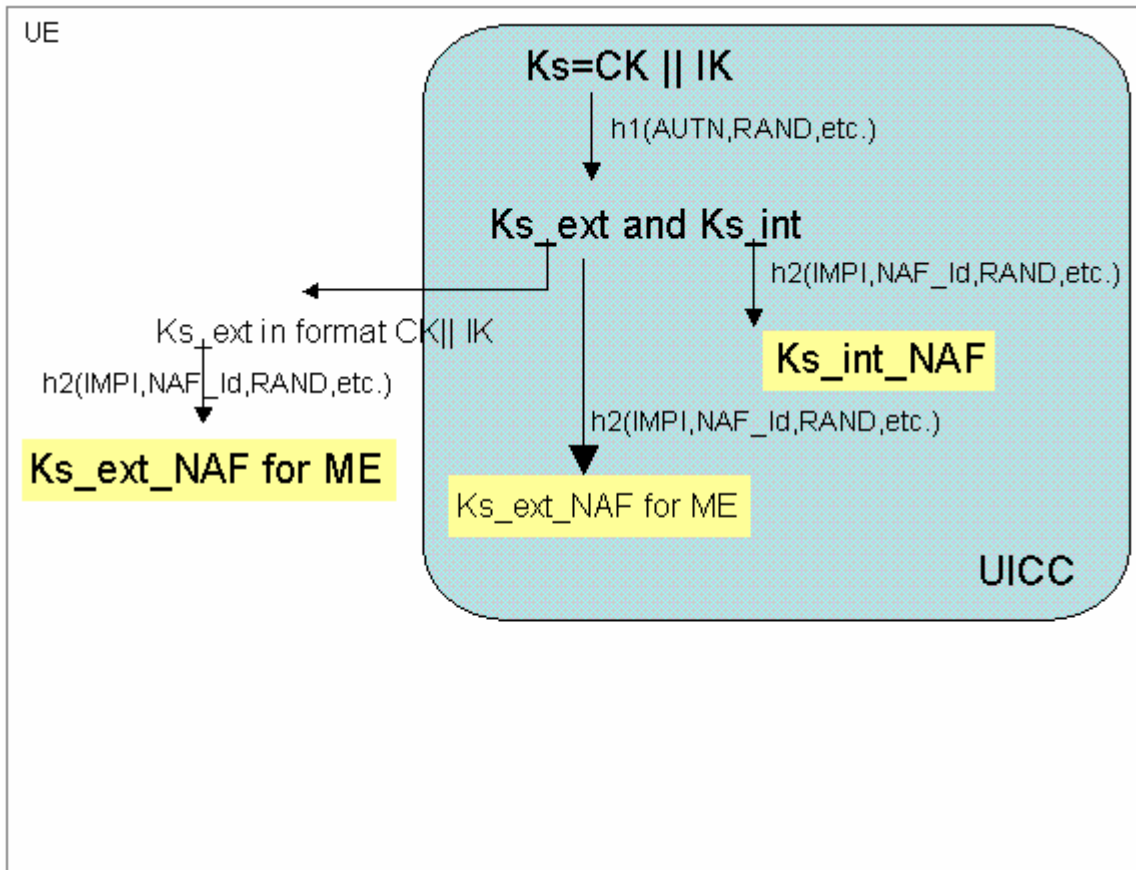
Figure 5.2: Key derivation for GBA-aware UICC when GBA-run was triggered

** END OF CHANGE ***