3GPP TSG-SA WG3 Meeting S3#34
Acapulco, Mexico, July 6-9, 2004

Tdoc ⌘S3-040536

CR-Form-v7

CHANGE REQUEST

⌘	**33.220**	**CR**	**CRNum**	⌘ **rev**	**-**	⌘	Current version:	**6.1.0** ⌘

*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:	⌘	GBA_U: key derivation procedure modified		
Source:	⌘	Nokia, Siemens		
Work item code:	⌘	GBA	***Date:*** ⌘	29/06/2004
Category:	⌘	**C**	***Release:*** ⌘	Rel-6
		Use <u>one</u> 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 <u>one</u> 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:	⌘	GBA_U key derivation is modified to better address the needs of GBA_U unaware ME. GBA_U key lengths are unified with GBA_ME.
Summary of change:	⌘	- Modified key derivation procedure is defined to be used with GBA_U. WIth this approach, the different usage scenarios between GBA_U aware and unaware components are clarified. - GBA_U key lengths are defined to be 256 bits
Consequences if not approved:	⌘	GBA_U key derivation in cases where GBA_U aware and unaware components (i.e., ME and UICC) is not clarified, and GBA_U has different key lengths than GBA_ME.

Clauses affected:	⌘	5.2.1, 5.3.2				
		Y	**N**			
Other specs affected:	⌘		X	Other core specifications	⌘	
			X	Test specifications		
			X	O&M Specifications		
Other comments:	⌘					

===== BEGIN CHANGE =====

5.2.1 Requirements on UE

The 3G AKA keys CK and IK resulting from a run of the protocol over the Ub reference point shall not leave the UICC.

The UICC shall be able to distinguish between authentication requests for GBA_U, and authentication requests for other 3G authentication domains.

Upon an authentication request from the ME, which the UICC recognises as related to GBA_U, the UICC shall derive ~~two~~four keys from CK and IK called CK', IK', CK" and IK". The length of CK', IK', CK" and IK" shall be 128 bits. All 3G MEs are capable of such a request.

Editor's note: The definition of exact derivation of CK', IK', CK" and IK" is left to ETSI SAGE.

The UICC shall deliver CK' and IK' to the ME in response to an AUTHENTICATE command as defined in TS 31.102 [1]. The ME shall derive the Ks_ext by concatenating CK' and IK'.

The UICC shall derive the Ks_int by concatenating CK" and IK", and store it to the UICC.

Upon request from the ME, the UICC shall be able to derive further NAF-specific keys (Ks_int_NAF) from the derived key stored on the UICC (Ks_int). Only GBA_U-aware 3G MEs are capable of such a request.

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

===== 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 generation of the Authentication Vector in the HSS and the local handling of keys 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.

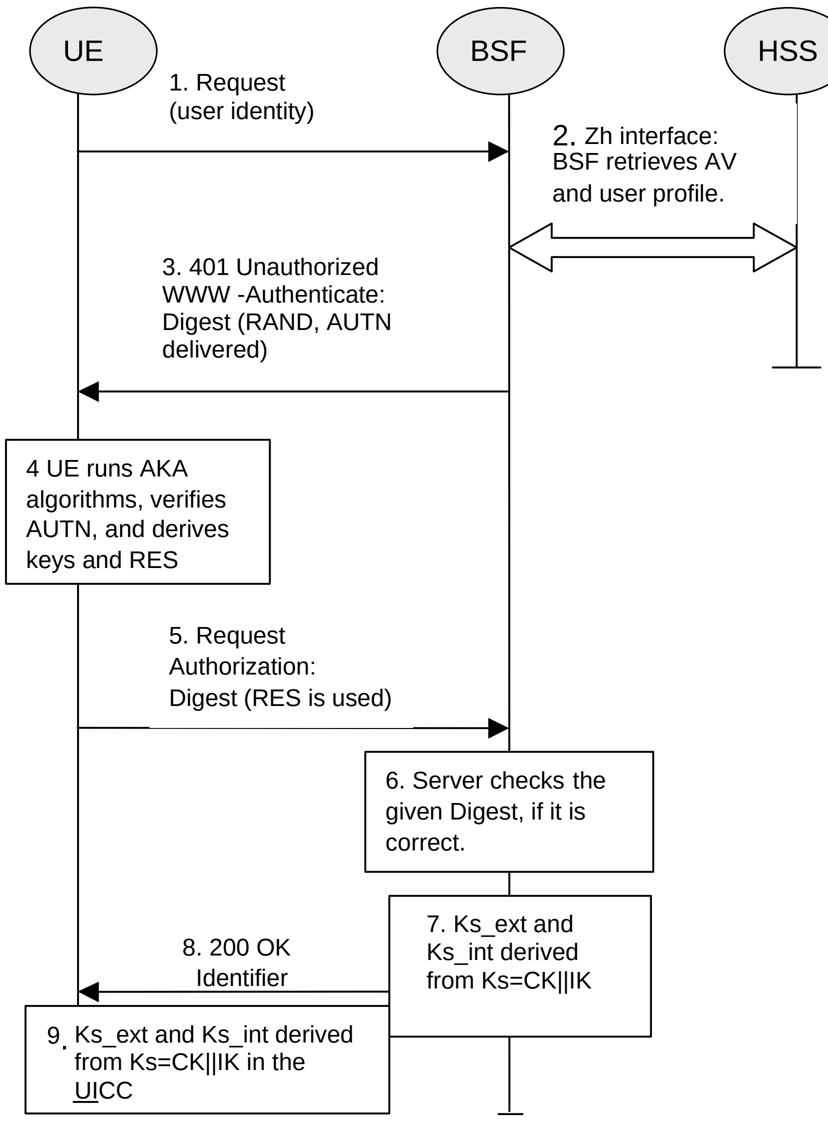

UE
BSF
HSS
1. Request (user identity)
2. Zh interface: BSF retrieves AV and user profile.
3. 401 Unauthorized WWW -Authenticate: Digest (RAND, AUTN delivered)
4 UE runs AKA algorithms, verifies AUTN, and derives keys and RES
5. Request Authorization: Digest (RES is used)
6. Server checks the given Digest, 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 user profile 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 HSS recognises that the UICC is GBA_U aware and that the request for AVs came from a GBA_U aware BSF, and generates a GBA_U-AV. If the BSF received GBA_U-AVs then it stores the XRES after flipping the least significant bit.

Editors' Note: The GBA_U-AV will be described within Annex D of this specification.

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

5. The UICC checks if a GBA_U-AV was received as specified in step 2 of this clause. If this is not the case, the UICC transfers RES, CK and IK to the ME, and the ME proceeds according to the procedures specified in section 4 of this document, without involving the UICC any further. If a GBA_U-AV was received, the UICC then derives four keys CK', IK', CK'', and IK'' by applying a suitable key derivation function h1 to CK, IK, and other possible key derivation parameters. ~~applies a suitable key derivation function h1 to Ks, which is the concatenation of~~ The UICC generates the Ks_int by concatenting 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), CK', and ~~Ks_ext~~IK' to the ME and stores Ks_int~~/ks_ext~~ on the UICC. The ME generates Ks_ext by concatenating CK' and IK'. See also figure 5.2.

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. The BSF checks if the AV was a GBA_U- AV as specified in step 2 of this clause. If this is not the case, the BSF applies the procedures specified in clause 4 of this document. If the GBA_U-AV was recognized then the BSF applies the key derivation function h1 to ~~Ks~~CK and IK as the UICC did in step 5 to derive CK', IK', CK", and IK"~~and possibly further h1-key derivation parameters to obtain two keys,~~. Ks_ext and Ks_int~~,~~ are generated ~~-~~in the same way as the UICC did in step 5. The Transaction Identifier value shall be also generated in format of NAI by taking the RAND value from step 3, and the BSF server name, i.e. RAND@BSF_servers_domain_name.

9. The BSF shall send a 200 OK message, including the Transaction Identifier, 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.

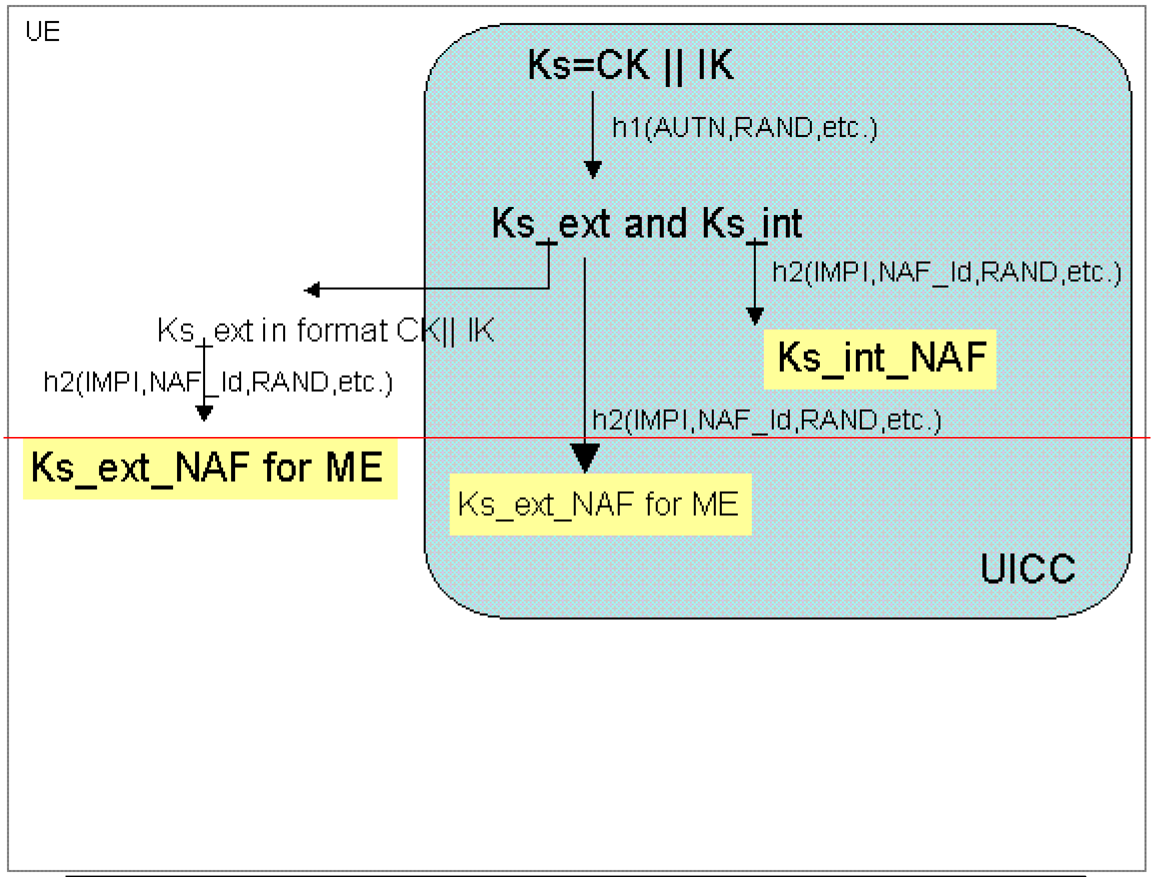
10. The BSF shall use the keys Ks_ext and Ks_int to derive the NAF-specific keys Ks_ext_NAF and Ks_int_NAF, each of length 256 bits, if requested by a NAF over the Zn reference point. Ks_ext_NAF and Ks_int_NAF are used for securing the Ua reference point. The UE shall use the key Ks_ext to derive the 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 Transaction Identifier 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.



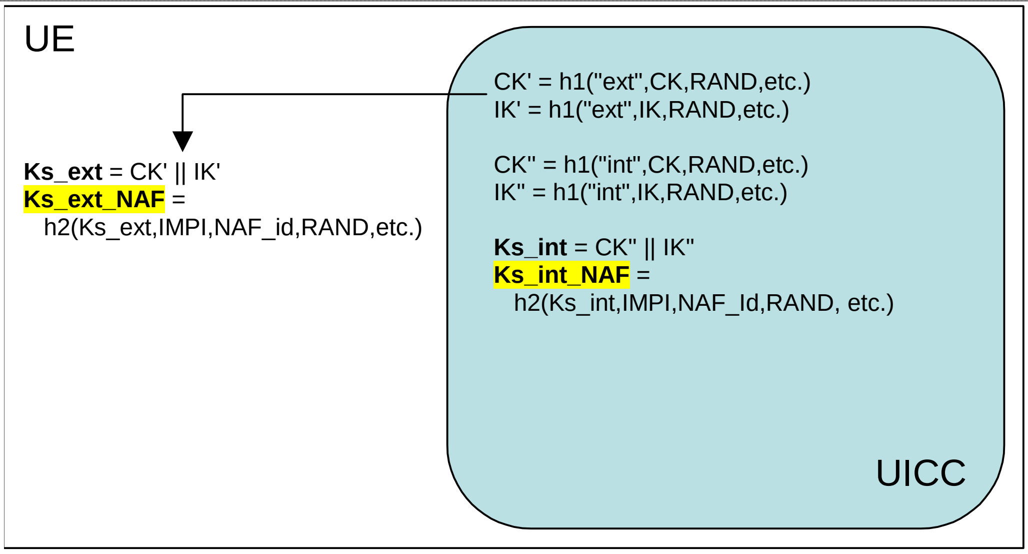

UE
CK' = h1("ext",CK,RAND,etc.)
IK' = h1("ext",IK,RAND,etc.)
CK" = h1("int",CK,RAND,etc.)
IK" = h1("int",IK,RAND,etc.)
Ks_int = CK" || IK"
Ks_int_NAF = h2(Ks_int,IMPI,NAF_Id,RAND, etc.)
Ks_ext = CK' || IK'
Ks_ext_NAF = h2(Ks_ext,IMPI,NAF_id,RAND,etc.)
UICC

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

Editor's note: Figure 5.2 needs to be update after ETSI SAGE has defined the key derivation functions for GBA_U.

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

GBA_ME/GBA_U scenarios in UE

June 29, 2004

Discussion

<u>Reasoning</u>:

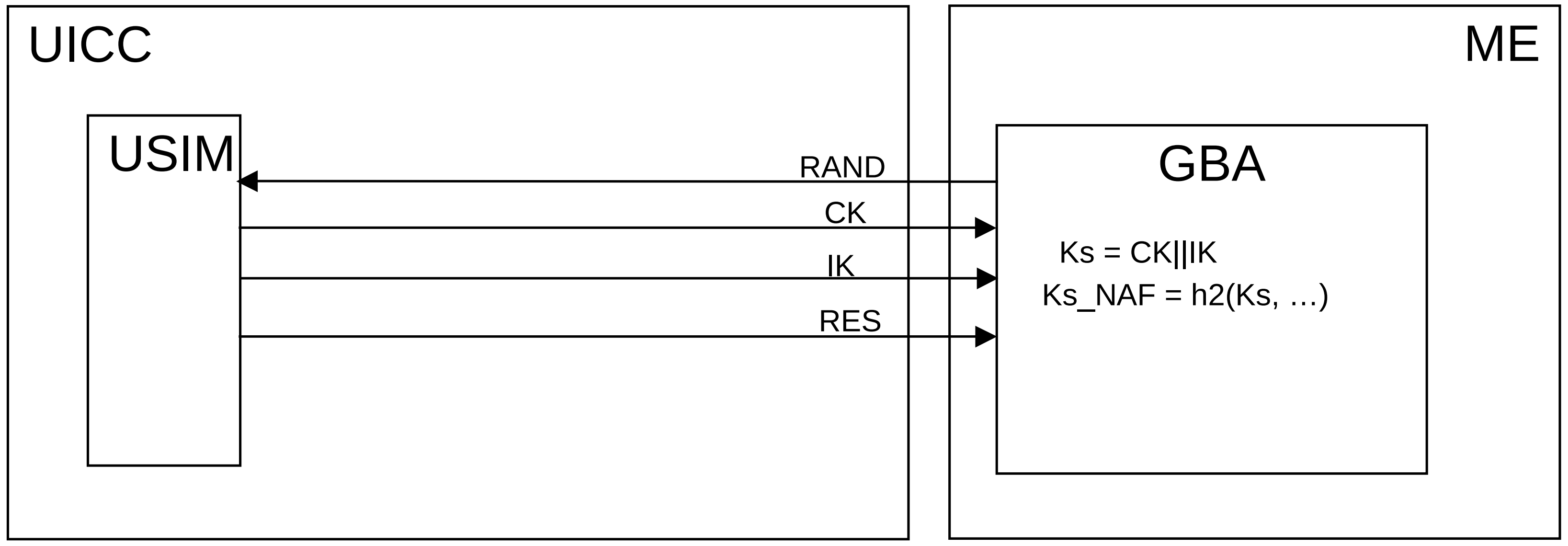
- Clarify key derivation
- Harmonize key derivation between GBA_ME and GBA_U
- All scenarios between GBA_U aware and unaware ME and UICC in UE are possible (see following slides)

<u>GBA_U functionality in the UICC can be (this is T3 issue):</u>

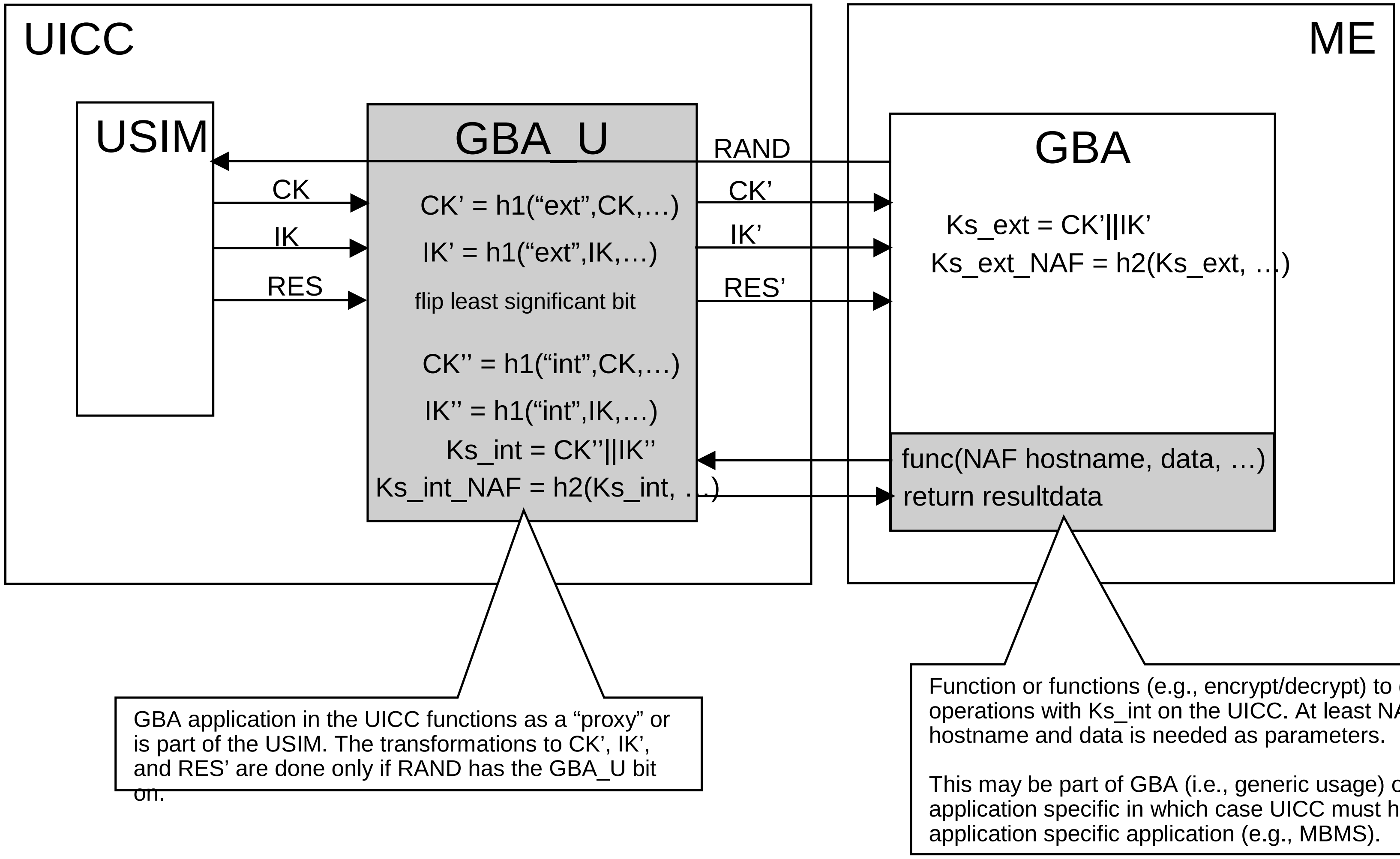
- "proxy" application between USIM and ME (see following slides)
- part of USIM implementation

Following slides describe the procedures between different components when they are either GBA_U aware or not.

GBA_ME



GBA_U

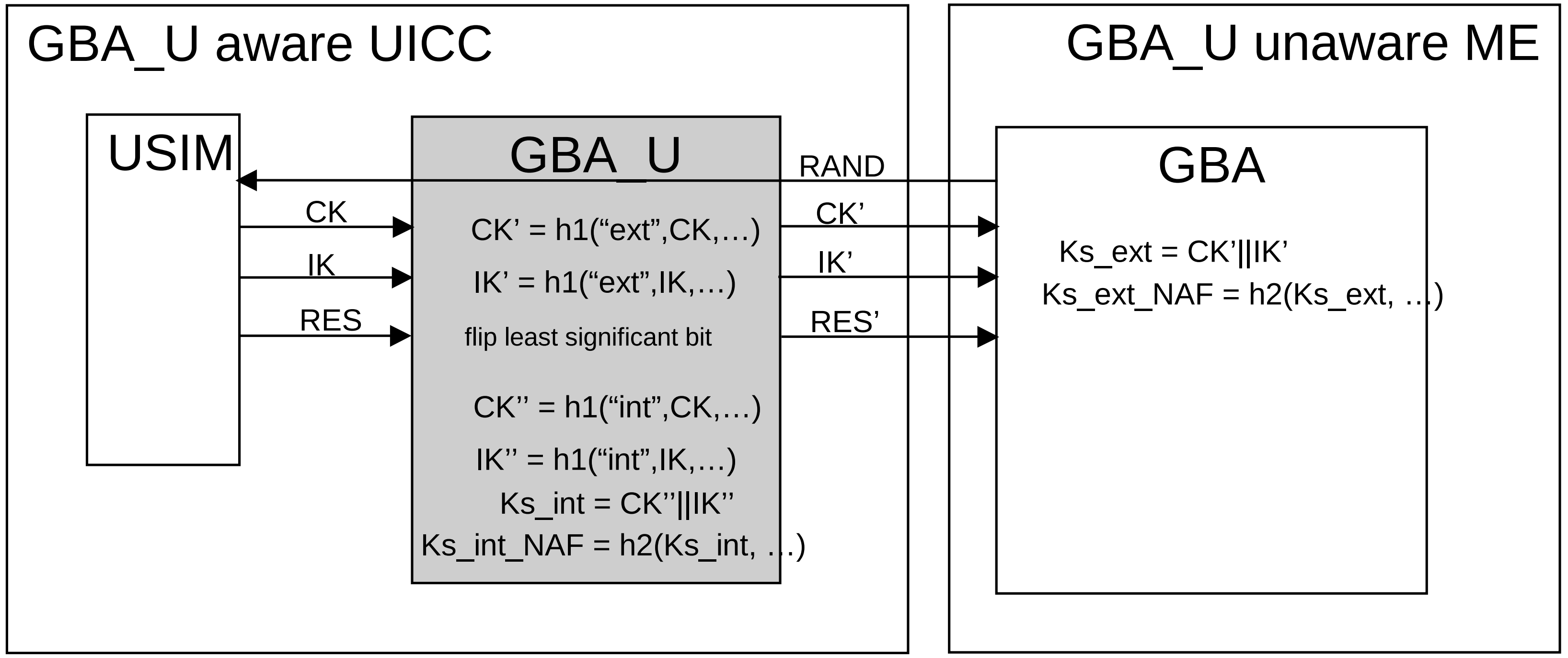


GBA application in the UICC functions as a "proxy" or is part of the USIM. The transformations to CK', IK', and RES' are done only if RAND has the GBA_U bit on.

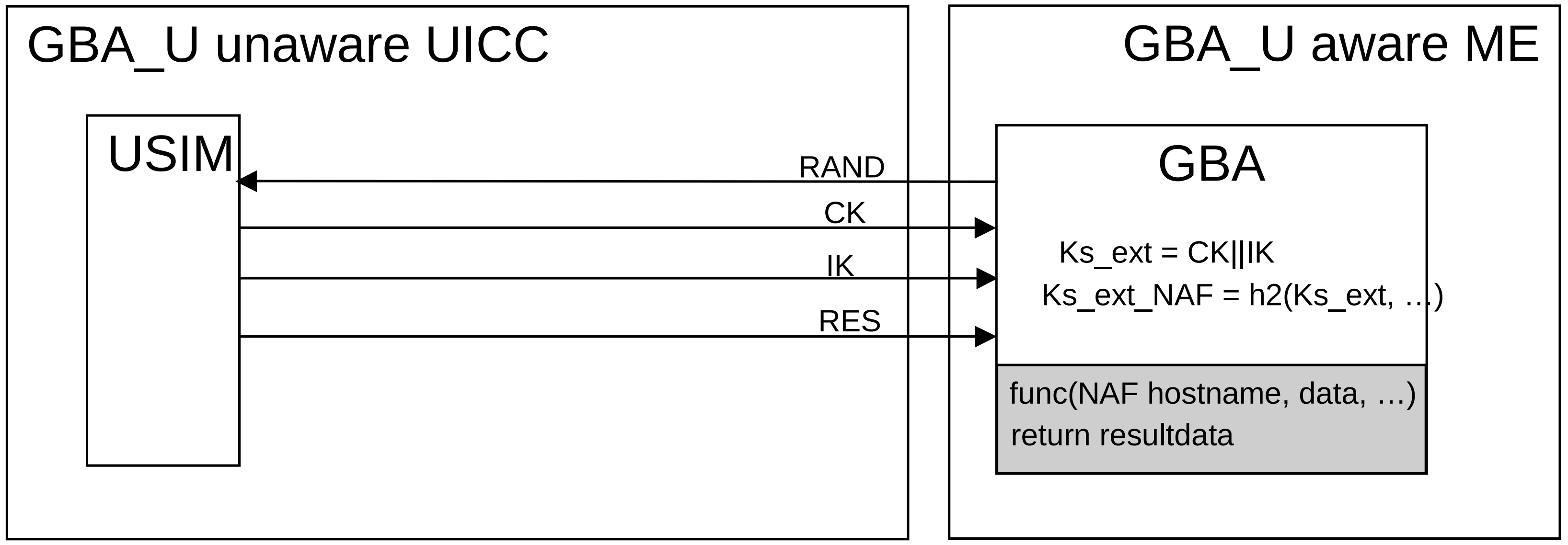
Function or functions (e.g., encrypt/decrypt) to do operations with Ks_int on the UICC. At least NAF hostname and data is needed as parameters.

This may be part of GBA (i.e., generic usage) or be application specific in which case UICC must have application specific application (e.g., MBMS).

Mixture 1


GBA_U aware UICC
USIM
GBA_U
CK
IK
RES
CK' = h1("ext",CK,…)
IK' = h1("ext",IK,…)
flip least significant bit
CK'' = h1("int",CK,…)
IK'' = h1("int",IK,…)
Ks_int = CK''||IK''
Ks_int_NAF = h2(Ks_int, …)
RAND
CK'
IK'
RES'
GBA_U unaware ME
GBA
Ks_ext = CK'||IK'
Ks_ext_NAF = h2(Ks_ext, …)

Mixture 2


GBA_U unaware UICC
USIM
RAND
CK
IK
RES
GBA_U aware ME
GBA
Ks_ext = CK||IK
Ks_ext_NAF = h2(Ks_ext, …)
func(NAF hostname, data, …)
return resultdata