3GPP TSG SA WG3 Security — SA3#37 **S3-050056**
February 21-25, 2005
Sophia Antipolis, France

Source:	Nokia
Title:	GBA User Security Settings (GUSS) transfer optimisation
Agenda item:	6.9.2 (GBA)
Document for:	Discussion/Decision

1 Introduction

This paper discusses the methods to transfer GUSS from the HSS to the BSF, and how to minimize the traffic load related to the GUSS transfer procedure.

2 Discussion

In the last SA3 meeting it was agreed that the BSF is able to fetch only one authentication vector (AV) at a time from the HSS over Zh reference point. This forces the BSF to update the GUSS that it has in its memory from the HSS when the UE bootstraps with the BSF. Also, if the NAF wants to get an updated USS (or multiple USSs) from the BSF, all it needs to do it to request the UE to bootstrapping with the BSF (i.e., renegotiate) and the next time the NAF request a particular USS from the BSF it knows that the USS is up to date.

But this procedure also forces the HSS to send the GUSS every time a new AV is requested - regardless of whether the GUSS has been updated in the HSS since the last time the BSF fetched the GUSS. Since the service usage is likely to be more frequent than subscriber profile updates, the frequent sending of GUSS may be unnecessary. And as GUSS may be quite large in size the number of times the GUSS is transferred from the HSS to the BSF should be minimized.

We propose to add a timestamp to the GUSS by the HSS to indicate the last time the GUSS was updated. If the BSF has a subscriber's GUSS in the memory and the BSF is fetching a new AV from the HSS for that subscriber, it would also include the timestamp of its GUSS to the request. Upon receiving the GUSS timestamp from the BSF, the HSS would compare it to the GUSS timestamp in its database. The HSS will include GUSS in the reply message to the BSF, only if the timestamps are different. If the timestamps are equal, then there is no need to include GUSS in the reply message.

If the BSF does not have a subscriber's GUSS in its memory and the BSF is fetching a new AV from the HSS for that subscriber, then there will be no timestamp in BSF's requiest and the HSS will include GUSS in the reply.

3 Conclusion & Proposal

We propose to add a GUSS timestamp to each GUSS indicating when the GUSS was last changed by the HSS. This timestamp is used to optimise the GUSS transfer policy between the HSS and the BSF: If the BSF has subscriber's GUSS in memory when it needs to fetch a new authentication vector (AV) for the subscriber, it will also include the GUSS timestamp to the request. Upon receiving the GUSS timestamp, the HSS will compare it to the timestamp of the GUSS it has in its database. If the timestamps are equal, then the HSS does not send the GUSS to the BSF as the BSF already has a copy of the GUSS. If the timestamps are not equal, then the HSS sends the GUSS as the BSF has an invalid GUSS, which needs to be updated.

The accompanying CR implements the necessary changes to TS 33.220.

3GPP TSG-SA WG3 Meeting #37
Sophia Antipolis, France, February 21-25, 2005

Tdoc ⌘att_S3-050056

CR-Form-v7.1

CHANGE REQUEST

⌘	**33.220**	**CR**	**046**	⌘**rev**	**-**	⌘	Current version:	**6.3.0** ⌘

*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:	⌘	GBA User Security Settings (GUSS) transfer optimisation		
Source:	⌘	Nokia		
Work item code:	⌘	SEC1-SC	***Date:*** ⌘	14/02/2005
Category:	⌘	**C**	***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:	⌘	A GUSS timestamp is added to each GUSS indicating when the GUSS was last changed by the HSS. This timestamp is used to optimise the GUSS transfer policy between the HSS and the BSF: If the BSF has subscriber's GUSS in memory when it needs to fetch a new authentication vector (AV) for the subscriber, it will also include the GUSS timestamp to the request. Upon receiving the GUSS timestamp, the HSS will compare it to the time of the GUSS it has in its databases. If the timestamps are equals, the HSS does not send the GUSS to the BSF as the BSF already has a copy of the GUSS. If the timestamp are not equal, the HSS sends the GUSS as the BSF has an invalid GUSS which needs to be updated.
Summary of change:	⌘	GUSS timestamp is added to the GUSS, and logic for handling the GUSS timestamp in both the BSF and the HSS is added.
Consequences if not approved:	⌘	The GUSS is needlessly sent over Zh reference point even though the BSF may already have a valid copy of the GUSS.

Clauses affected:	⌘	3.1, 4.4.5, 4.5.2, 5.3.2				
		Y	**N**			
Other specs affected:	⌘	X		Other core specifications	⌘	TS 29.109
			X	Test specifications		
			X	O&M Specifications		
Other comments:	⌘					

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

3.1 Definitions

For the purposes of the present document, the following terms and definitions apply:

Application: In all places in this document where the term application is used to refer to a service offered by the MNO or a third party to the mobile subscriber, then it always denotes the type of application and not the actual instance of an application installed on an application server.

Bootstrapping Server Function: BSF is hosted in a network element under the control of an MNO. BSF, HSS, and UEs participate in GBA in which a shared secret is established between the network and a UE by running the bootstrapping procedure. The shared secret can be used between NAFs and UEs, for example, for authentication purposes.

Bootstrapping Usage Procedure: A procedure using bootstrapped security association over Ua reference point.

GBA Function: A function on the ME executing the bootstrapping procedure with BSF (i.e. supporting the Ub reference point) and providing Ua applications with security association to run bootstrapping usage procedure. GBA function is called by a Ua application when a Ua application wants to use bootstrapped security association.

ME-based GBA: in GBA_ME, all GBA-specific functions are carried out in the ME. The UICC is GBA-unaware. If the term GBA is used in this document without any further qualification then always GBA_ME is meant, see clause 4 of this specification.

UICC-based GBA: this is a GBA with UICC-based enhancement. In GBA_U, the GBA-specific functions are split between ME and UICC, see clause 5 of this specification.

Network Application Function: NAF is hosted in a network element. GBA may be used between NAFs and UEs for authentication purposes, and for securing the communication path between the UE and the NAF.

Bootstrapping Transaction Identifier: the bootstrapping transaction identifier (B-TID) is used to bind the subscriber identity to the keying material in reference points Ua, Ub and Zn.

GBA User Security Settings: GUSS contains the BSF specific information element and the set of all application-specific USSs.

GUSS timestamp: the timestamp of the GUSS is set by the HSS. It changes whenever the HSS has modified the GUSS.

NAF Group: A grouping of NAFs to allow assignment of different USSs to NAFs representing the same application. This grouping is done in each home network separately, i.e. one NAF contacting BSFs in different home networks belongs to different groups in every home network.

Ua Application: An application on the ME intended to run bootstrapping usage procedure with a NAF.

User Security Setting: A USS is an application and subscriber specific parameter set that defines two parts, an authentication part, which contains the list of identities of the user needed for the application (e.g. IMPI, IMPUs, MSISDN, pseudonyms), and an authorisation part, which contains the user permission flags (e.g. access to application allowed, type of certificates which may be issued). Sometimes also called application-specific user security setting. The USS is delivered to the BSF as a part of GUSS from the HSS, and from the BSF to the NAF if requested by the NAF.

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

4.4.5 Requirements on reference point Zh

The requirements for reference point Zh are:

- mutual authentication, confidentiality and integrity shall be provided;

NOTE 1: This requirement may be fulfilled by physical or proprietary security measures if BSF and HSS are located within the same operator's network.

- the BSF shall be able to send bootstrapping information request concerning a subscriber;
- the BSF shall be able to send GBA user security settings timestamp to the HSS;
- the HSS shall be able to send 3GPP AKA vectors to the BSF in batches;
- the HSS shall be able to send the complete set of subscriber's GBA user security settings needed for security purposes to the BSF;

NOTE 2: If subscriber's GUSS is updated in HSS, this is not propagated to the BSF. The GUSS in the BSF is updated when the BSF next time fetches the authentication vectors and GUSS from the HSS over Zh reference point as part of the bootstrapping procedure.

NOTE 3: If the BSF has sent GUSS timestamp to the HSS, the HSS will compare that timestamp with the timestamp of the GUSS that is stored in the HSS. If the timestamps are equal, then the HSS will not send the GUSS to the BSF as the BSF already has a copy of the GUSS. If the timestamp are not equal, the HSS will send the GUSS to the BSF as the BSF's copy of the GUSS needs to be updated.

- no state information concerning bootstrapping shall be required in the HSS;
- all procedures over reference point Zh shall be initiated by the BSF;
- the number of different interfaces to HSS should be minimized.

===== 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.

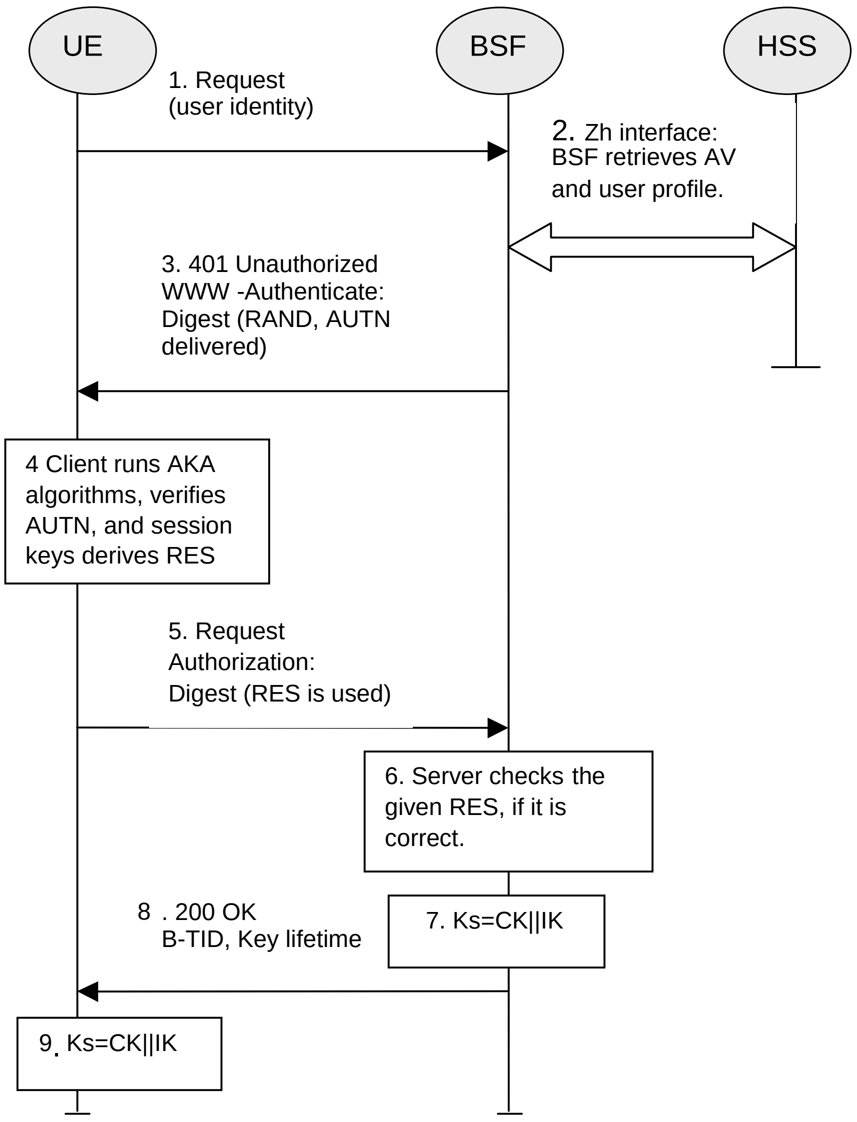
UE
BSF
HSS
1. Request (user identity)
2. Zh interface: BSF retrieves AV and user profile.
3. 401 Unauthorized WWW -Authenticate: Digest (RAND, AUTN delivered)
4 Client runs AKA algorithms, verifies AUTN, and session keys derives RES
5. Request Authorization: Digest (RES is used)
6. Server checks the given RES, if it is correct.
7. Ks=CK||IK
8 . 200 OK B-TID, Key lifetime
9. Ks=CK||IK

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 Authentication Vector (AV, AV = RAND||AUTN||XRES||CK||IK) over the reference point Zh from the HSS.

   If the BSF has a GUSS for the subscriber that has been fetched from the HSS during a previous bootstrapping procedure, the BSF shall include the GUSS timestamp in the request message. Upon receiving that timestamp, the HSS shall compare it with the timestamp of the GUSS stored in the HSS. If the timestamps are equal, then the HSS shall not send the GUSS to the BSF, if they are not equal, or if there is no timestamp in the request message, then the HSS shall send the GUSS to the BSF.

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), where KDF is 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.

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.

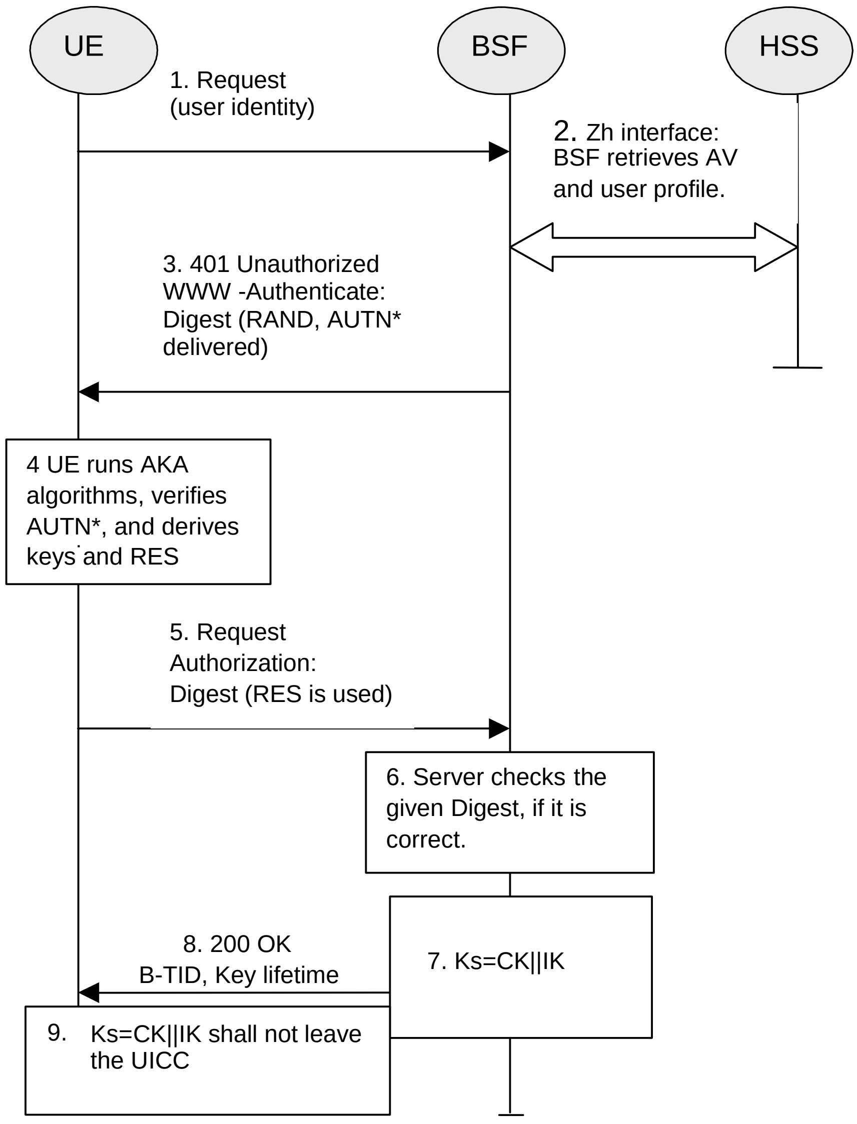
UE
BSF
HSS
1. Request (user identity)
2. Zh interface: BSF retrieves AV and user profile.
3. 401 Unauthorized WWW -Authenticate: Digest (RAND, AUTN* delivered)
4 UE runs AKA algorithms, verifies AUTN*, and derives keys and RES
5. Request Authorization: Digest (RES is used)
6. Server checks the given Digest, if it is correct.
7. Ks=CK||IK
8. 200 OK B-TID, Key lifetime
9. Ks=CK||IK shall not leave 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 Authentication Vector (AV, AV = RAND||AUTN||XRES||CK||IK) over the Zh reference point from the HSS.

   If the BSF has a GUSS for the subscriber that has been fetched from the HSS during a previous bootstrapping procedure, the BSF shall include the GUSS timestamp in the request message. Upon receiving that timestamp, the HSS shall compare it with the timestamp of the GUSS stored in the HSS. If the timestamps are equal, then the HSS shall not send the GUSS to the BSF, if they are not equal, or if there is no timestamp in the request message, then the HSS shall send the GUSS to the BSF.

   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⊕ Trunc(SHA-1(IK))

NOTE: Trunc denotes that from the 160 bit output of SHA-1 [21], the 64 bits numbered as [0] to [63] are used within the * operation to MAC.

   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* ⊕ Trunc(SHA-1(IK))). 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. The UICC then transfers RES (after flipping the least significant bit) to the ME and stores Ks, which is the concatenation of CK and IK, on the UICC.

5. The ME 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 the key 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 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 key Ks.

9. Both the UICC and the BSF shall use the Ks to derive NAF-specific keys Ks_ext_NAF and Ks_int_NAF during the procedures as specified in clause 5.3.3, if applicable. Ks_ext_NAF and Ks_int_NAF are used for securing the Ua reference point.

   Ks_ext_NAF is computed in the UICC as Ks_ext_NAF = KDF(Ks, h1-key derivation parameters), and Ks_int_NAF is computed in the UICC as Ks_int_NAF = KDF(Ks, h1-key derivation parameters), where KDF is the key derivation function as specified in Annex B, and the 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. The key derivation parameters used for Ks_ext_NAF derivation must be different from those used for Ks_int_NAF derivation. This is done by adding a static string "gba-me" in Ks_ext_NAF and "gba-u" in Ks_int_NAF as an input parameter to the key derivation function.

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

The UICC and the BSF 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.

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