3GPP TSG SA WG3 Security — SA3#35 **S3-040825**
October 5-8, 2004
St Paul's Bay, Malta

3GPP TSG SA WG3 Security — S3#35 **S3-040773**
5 – 8 October 2004
Malta

Source: ~~**Gemplus, Axalto, Oberthur**~~ **Nokia, Siemens, Ericsson**

Title: **GBA_U: finalisation of GBA_U procedure commented by Nokia, Siemens, Ericsson**

Document for: **Discussion and decision**

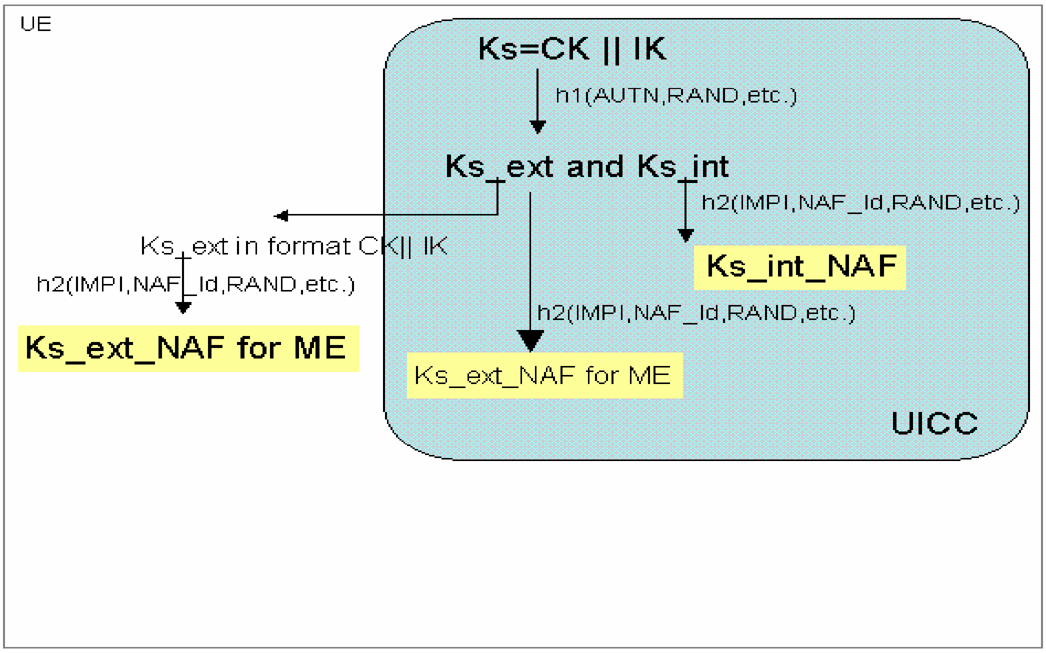
Agenda Item:

1. Introduction

The current version of TS 33.220 "Generic Bootstrapping Architecture" mentions that the location (whether in the UICC or in the ME) of the storage of Ks_ext is for further study. This contribution proposes some elements of comparison in order to finalize the GBA_U procedure.

2. Current status of GBA-U key derivation

The current TS 33.220 [1] proposes the following key derivations for GBA-U:



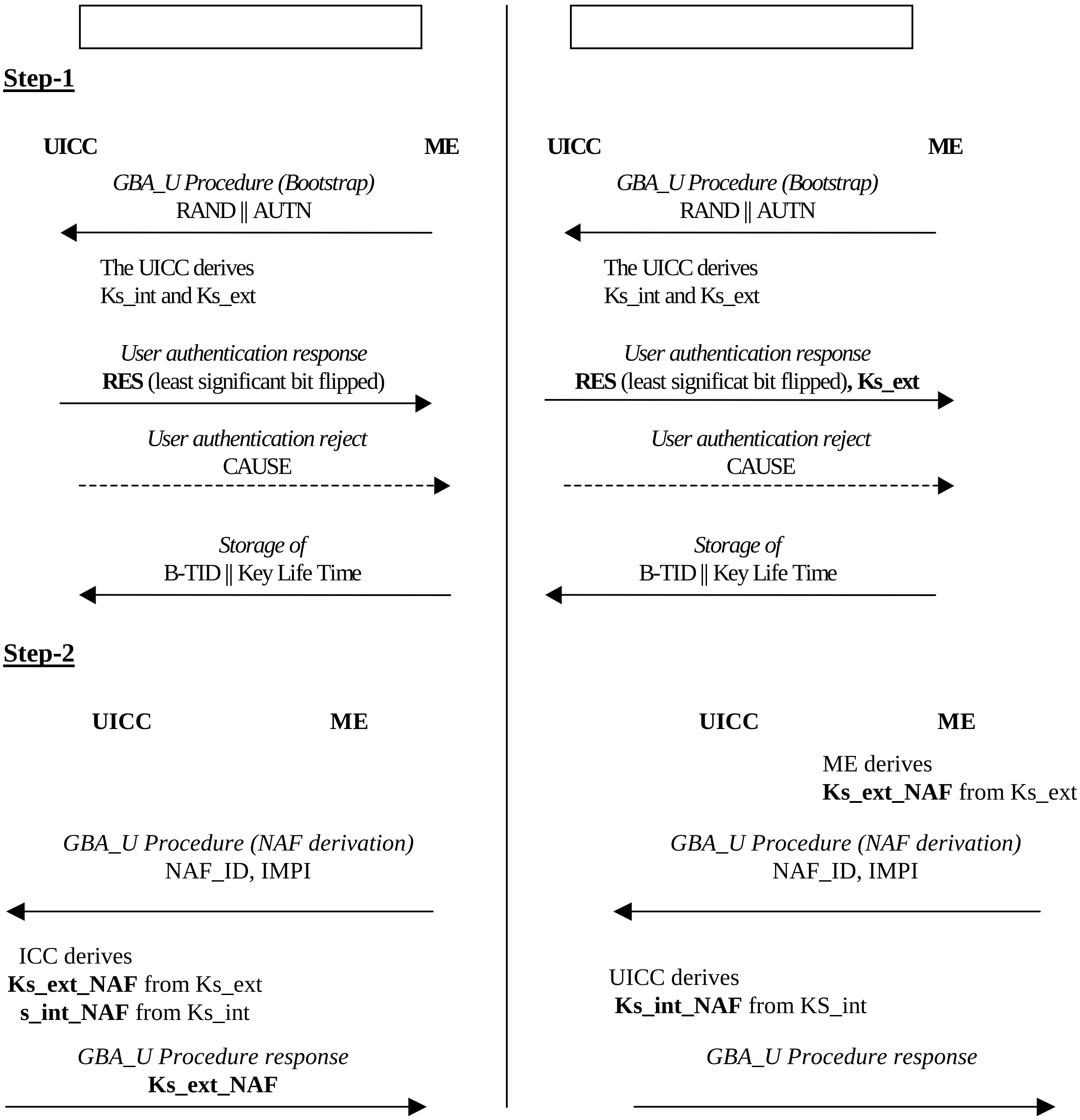
The TS states that:

- The location (whether in the UICC or in the ME) of the storage of Ks_ext is for further study.

- When the UE is powered down, or when the UICC is removed, any GBA_U keys shall be deleted from storage in the ME. There is no need to delete the keys Ks_int and Ks_int_NAF from storage in the UICC.

<u>GBA_U steps</u>

The difference of processing between Ks_ext stored on the UICC and Ks_ext stored on the ME are the following:



In case of Ks_ext stored on the UICC, another alternative to derive Ks_xx_NAF keys was proposed at SA3#34. This solution is described in section 3.6 of the present document. Besides an analysis of the key derivation alternatives is provided in another SA3#35 contribution [2].

The choice of the derivation method has no impact on the elements of comparison excepting the processing time and the implementation complexity.

3. Elements of comparison

3.1. Security

The storage of bootstrapping keys on the UICC provides a higher level of security; this is demonstrated in the following attack description.

The attack consists in sending authenticated application requests from a device not hosting the UICC to new NAFs.

Context:
Ks_ext_NAF = h2 (Ks_ext, h2-key derivation parameters)
where h2-key derivation parameters include IMPI, NAF_ID and RAND.

- Scenario 1: Ks_ext stored on the ME

An attacker accessing the ME [To access the ME without user noticing requires quite some effort depending on the type of attack].can retrieve [Even, if an attacker is able to access the ME, that does not imply that he is also able to retrieve this data, since the ME is not such an open platform as the PC. The ME is protected by the operator and might have in addition internal methods of protection. Hence, attacks known from the Internet can not be "translated" into the mobile environment.] Ks_ext, IMPI, RAND, B-TID and current Ks_ext_NAF values.
The knowledge of Ks_ext, IMPI and RAND allows the attacker to compute Ks_ext_NAF of any NAF during the key lifetime of Ks_ext [If the Ks_ext is stored in the ME, the de-facto lifetime of the key, will not be that long.] since the NAF_specific value is the NAF_ID (public value sent by the NAF [Hence a potential attacker had also to obtain the NAF_ID e.g. by intercepting the Ua communication (what might be quite an effort).]).

So, in case of Ks_ext stored on the ME, an attacker needs only **one connection** with the ME to allow a device not hosting the UICC to perform **authenticated** application requests to **any NAF** during the key lifetime of Ks_ext.

- Scenario 2: bootstrapping keys (e.g. Ks_ext) stored on the UICC

An attacker accessing the ME can retrieve IMPI, RAND, B-TID and current Ks_ext_NAF values.
This attacker can send authenticated application requests to the corresponding NAF by using the retrieved Ks_ext_NAF keys.
To send authenticated application requests to a new NAF, the attacker needs to establish a new connextion [A new connection is not needed if the attacker makes the ME send the "Ks_ext_NAF retrieve" command to the UICC at the same time IMPI, RAND, B-TID and current Ks_ext_NAF values are retrieved. More over, the attacker may send the "Ks_ext_NAF retrieve" command to the UICC multiple times to retrieve all the Ks_ext_NAF values the attacker wants.] to the ME in order to make the ME send the "Ks_ext_NAF retrieve" command to the UICC and, after the execution of the commande, retrieve the Ks_ext_NAF value on the ME.

So, in case of Ks_ext stored on the UICC [An attacker might also install a Trojan horse and attack the card by launching a man-in-the-middle attack and acting as a proxy, the risk is the same for both scenarios.], if an attacker wants to send authenticated application requests to new NAFs from a device not hosting the UICC, then he needs to establish **one connection** with the ME for **each NAF** to contact [Not true, see previous comment.].

Conclusion: the storage of bootstrapping keys on the UICC offers a higher security level [The arguments, that the security risk of storing the Ks_ext in the ME is higher than storing it in UICC are not convincing.].

3.2. Bootstrapping key lifetime

- Scenario 1: Ks_ext stored on the ME
  Ks_ext shall be deleted when the UE is powered down or when the UICC is removed [A long living key might have to be revoked, updated and might be easier compromised (e.g. NAF is hacked). For these cases, the ME would be required to support additional functionalities on the UICC-ME to request key revocation, update etc.].

- Scenario 2: bootstrapping keys stored on the UICC
  There is no need to delete bootstrapping keys (e.g. Ks_int and Ks_ext) from the UICC in case of UE power down or UICC removal. Only, the Ks_ext_NAF key stored on the ME shall be deleted.

  So, the bootstrapping key lifetime is longer when it is only stored on the UICC, this leads to:
    - Decrease the frequency of bootstrapping procedures
    - Decrease the consumption of authentication vectors [This is the same argument as above, but just with different wording (frequency vs. consumption)]

3.3. Processing time

In case of application requiring the use of Ks_ext_NAF only, the storage of the Ks_ext on the UICC implies that the ME sends a second call to the ME to retrieve Ks_ext_NAF. So, compared to the storage of Ks_ext on the ME, there is a processing delay due to the call to the UICC but this delay represents only some few ten ms [Depending on the application type, the time that is needed for the communication between UICC and non-terminal entity might be too long e.g. fast payments, ticketing, streaming.]. Moreover, in case of Ks_ext stored on the UICC, it was previously mentioned that an alternative for Ks_xx_NAF key derivation could be proposed; Ks_ext and Ks_int could be replaced with a single Ks key Cf [2]. In this case the processing time of the first step (corresponding to the Bootstrapping mode) would be decreased then the difference of the GBA_U processing time between the storage on the UICC and the storage on the ME would be also decreased.

In conclusion, the processing delay in case of storage on the UICC would be very small.

3.4. Key derivation complexity

At SA3#34, it was proposed to optimise the GBA_U bootstrapping procedure by removing at least one key derivation procedure. In fact, the current version of TS 33.220 requires the UICC and the BSF to perform four key derivation procedures to calculate Ks_int_NAF and Ks_ext_NAF keys. First the UICC performs Ks derivation, then performs Ks_int and Ks_ext derivation, and finally performs Ks_int_NAF and Ks_ext_NAF derivation. This GBA_U bootstrapping procedure can be optimised by removing Ks_int and Ks_ext derivation and by directly deriving Ks_int_NAF and Ks_ext_NAF from Ks [This is only possible, if Ks_ext is stored in UICC.].

In addition to all the benefits of a bootstrapping procedure where associated keys are stored in the UICC, this solution presents the following advantages:

- It reduces the bootstrapping time, as the UICC would have to perform one key derivation for the bootstrapping instead of two. This will lead to better performance in the UICC and BSF (less network resources consumption/less booting time).
- It reduces the implementation complexity in the BSF, as the GBA_U procedures will be similar to the GBA_ME procedures at least for the Bootstrapping mode (Step-1).

3.5. Implementation cost

As mentioned in the previous clause the storage of Ks_ext on the UICC means that the ME shall send a second call to the UICC to retrieve Ks_ext_NAF but this second call requires no new command [It would

require from the ME to handle different kind of UICC differently, including recognition of UICC capabilities, message flows, error messages, and future extensions.]. T3 agreed at T3#32 meeting the creation of a GBA Security Context in the AUTHENTICATE command with two specific modes: Bootstrapping mode and NAF Derivation mode, cf T3-040540 CR [xx [A ME mandatory has to support GBA_U would then also be forced to implement this further functionalities, like this one.]].
For GBA_U the ME shall implement the AUTHENTICATE command with GBA security context - Bootstrapping mode anyway to retrieve the RES value and perform the http digest authentication. The difference between the Bootstrapping mode and the NAF Derivation mode only consists in the AUTHENTICATE command data.

So, in case of application only requiring the use of Ks_ext_NAF, there is no cost of implementation for the second call to the UICC. From the ME side, the difference between a GBA_ME and a GBA_U [GBA_ME is more stable than GBA_U, so if the ME would be mandate to support GBA_U, then the changes in GBA_U will also affect the ME implementation (even the ones, that do not use GBA_U)!] will be in the implementation and handling of the UICC AUTHENTICATE command. The support of GBA_U in the ME requires only the extension of the currently defined AUTHENTICATE command.

3.6. GBA_ME not supporting GBA_U

Some companies objected the storage of Ks_ext on the UICC because of the existence of GBA-aware ME not supporting GBA_U capabilities [This might be the case for ME that support Presence and do not support MBMS.]. In case of GBA_U aware UICC inserted in GBA-aware ME not supporting GBA_U capabilities, the ME does not receive Ks_ext and cannot derive Ks_ext_NAF [This argument only applies, if the Ks_ext is only stored in UICC.].

This argument "GBA-capable ME not supporting GBA_U" is not relevant [It is relevant, since a GBA capable ME device should not be forced to support a functionality that might not be used at all.] to prevent the storage of Ks_ext on the UICC because SA3 did not yet decided to allow [See SA3#33 (Beijing) meeting report: "It was agreed that the GBA method will be used for MBMS Security (GBA_U + GBA_ME + MIKEY). Based on this, it was noted that if a Terminal is to support MBMS, then it will need to support GBA_U." Hence, the working assumption was always, that a terminal not having MBMS is not required to support GBA_U.] the scenario of "GBA-capable ME not supporting GBA-U". Moreover, there are several arguments [There is a commenting paper on this contribution as well.] to require that GBA-aware ME shall support both GBA-ME and GBA-U, Cf SA3#35 contribution [3].

So, it is the security level [We believe that a longer key lifetime would actually increase the risk of key compromise. Also difference in security between the Ks_ext being stored in the UICC or in the ME is not that much higher that is implied in this paper because the Ks_ext_NAF values will be present in the ME in both cases in the end.] due to the storage of Ks_ext that should influence the decision on the support of GBA features within a Rel-6 ME. The argument related to "GBA-aware ME not supporting GBA_U" should not be taken into consideration to decide on the storage of bootstrapping keys [The argument is important to manufacturers, hence consideration seems to be advisable.].

4. Conclusion

The storage of bootstrapping keys on the UICC offers a higher security level [The security level is different, longer key life-time, might even decrease the security and force additional burden upon operators and NAFs to revoke and update keys. Also difference in security between the Ks_ext being stored in the UICC or in the ME is not that much higher that is implied in this paper because the Ks_ext_NAF values will be present in the ME in both cases in the end.], extends the bootstrapping key lifetime and decreases the frequency of the bootstrapping procedures and the consumption of authentication vectors. An optimization of the GBA_U bootstrapping procedure could make the implementation easier in the BSF and UICC, and could decrease significantly the bootstrapping time.

So, we kindly recommend SA3 to require the storage of bootstrapping keys on the UICC for GBA_U and to optimize the GBA_U bootstrapping procedures.

Two CRs ([4] and [5]) implement this proposal.

We believe that the arguments given in this document are not convincing and thus do not warrant the requirement that the Ks_ext should be stored in the UICC only and that the subsequent Ks_ext_NAF key derivation should be done in the UICC. CRs S3-040783 and S3-040784 implements alternative proposal.

5. References

[1] TS 33.220, v6.1.0 “Generic Authentication Architecture, Generic bootstrapping architecture” Rel-6

[2] TD S3-040xxx, “Alternatives for GBA_U derivation”, Gemplus, Axalto, Oberthur, SA3#35

[3] TD S3-040xxx, “Support of GBA_U capabilities for Rel-6 MEs”, Gemplus, Axalto, Oberthur, SA3#35

[4] TD S3-040xxx, “CR: Storage of Ks_ext on the UICC”, Gemplus, Axalto, Oberthur, SA3#35

[5] TD S3-040xxx, “CR: Optimization of the GBA_U key derivation procedure”, Gemplus, Axalto, Oberthur, SA3#35