3GPP TSG SA WG3 Security — SA3#36 **S3-041049**
November 23-26, 2004
Shenzhen, China

Source:	**Siemens**
Title:	**Comments on S3-040940, 941, 942 on "Key freshness in GBA" (all by "3") and on S3-040982 "Key lifetime clarifications" (by Nokia)**
Document for:	**Discussion and decision**
Agenda Item:	**6.9.2 GBA**

S3-040940 is a discussion paper on the potential need to enhance the specification of GBA in TS 33.220 to include guarantees of freshness of the NAF-specific keys Ks_NAF. (In the case of GBA_U, the corresponding keys are Ks_int_NAF and Ks_ext_NAF respectively. We limit ourselves to the case of GBA_ME in these comments, as the considerations for GBA_U are similar.) S3-040941 and S3-040942 implements alternative solutions in CRs. S3-040941 proposes an exchange of additional nonces to be included in the key derivation process, while S3-040942 simply proposes to add a note to TS 33.220 informing about the guarantees given by GBA regarding key freshness. S3-041024 proposes yet another solution, namely to inform the NAF also about the key generation time. We comment these contributions below.

Key freshness vs. replay protection:

These two notions are different, they may, however, be related. Guarantee of key freshness is a property of a key establishment protocol, the guarantee may be that the established key has not been used in a previous protocol run (as e.g. in UMTS AKA), or else, that it was generated during the current protocol run, or else that it was generated or will expire at a certain time.

Replay protection is a property of an arbitrary security protocol, which ensures that a message cannot be replayed without detection. For the two examples of Ua protocols, which are currently considered for Release 6:

- TLS ensures replay protection by computing fresh Diffie-Hellman keys in a full handshake, and keeping state on sequence numbers during the lifetime of the established session key;
- http digest ensures replay protection by the use of nonces and sequence numbers (where the use of cnonce and nonce-count is optional).

Is key freshness a property required of key distribution protocols, and, in particular, of GBA?

There is no clear-cut answer to this, as the answer depends on the overall design goals to be achieved with GBA. It appears to the author of these comments that it is a reasonable design goal to conceive of the key Ks_NAF as of a (semi-)permanent shared key for the Ua protocol. It is then the Ua protocol, which has to take care of replay attacks.

It should be noted in this context that

- the Ua protocols currently specified for use with GBA, http digest and pskTLS, are not vulnerable to replay attacks;

- a very popular key distribution protocol, Kerberos, does not provide key freshness in the sense required by S3-0940, as tickets containing session keys may be re-used during the session key lifetime.

Are there currently means for the UE and the NAF to ensure freshness of the key Ks_NAF?

Yes, there are. If a UE or a NAF require, for a particular Ua protocol, a guarantee that a key Ks_NAF was not used in a previous protocol run, then the UE or the NAF can store the keys Ks_NAF or the corresponding key identifiers, i.e. the

B-TID values, during the key lifetime, and then check for a previous key use in a new Ua protocol run. It may be useful anyhow to store keys at the NAF to improve performance and reduce protocol runs over Zn.

Is there are straightforward solution to implement the changes proposed in S3-040941?

No, it is said in S3-0940 about the issue of transferring the NAF-generated additional nonce to the UE for the two options mentioned that "The exact details of doing either of these are specific to each Ua protocol." This means that the specification of the proposed key freshness mechanism cannot be achieved in a generic way as part of TS 33.220, which seems to be one of the aims of the proposal, but needs additional specification work for every Ua protocol under consideration. For some Ua protocols, it may even be unfeasible to do this.

The proposed solution has potential problems in other areas as well. E.g. it is proposed that the B-TID be extended to include additional nonces generated by UE and NAF, and that the extended B-TID going from UE to NAF may be different from the extended B-TID going into the other direction from the NAF to the UE. But when used with http digest (RFC 2617), the B-TID is the user name which shall be the same in both directions. To include the transfer of nonces in http digest in different ways may also be non-conforming to RFC 2617.

Are there risks in accepting the changes proposed in S3-040941?

Key establishment protocols are notoriously tricky to design, and errors in protocol design are notorious. TS 33.220 has been around for a quite a while, and there is at least some hope, that people used the opportunity to thoroughly study it. To introduce quite complex changes at the last minute, considerably increases the risk to introduce protocol errors.

Does the introduction of a key generation time help?

This is proposed in S3-040982. It does not really help, as the key lifetime and key generation are associated with Ks, and not Ks_NAF, and thus do not give sufficient information about the freshness of Ks_NAF. Also, the specifications allow the run of the Ub protocol to take place at any time before a run of the Ua protocol. So, the key generation time would only be a weak indicator of key freshness.

Which way forward?

The way forward preferred by the author of these comments is to leave GBA as it currently is and let the Ua protocol or NAF (by storing used keys) take care of the replay protection. Therefore, S3-040941 and S3-040982 are recommended to be rejected. But we have no objection to further study this issue in Release 7. We would also support the inclusion of a note, as proposed in S3-040942. We suggest to somewhat re-formulate the note as follows:

~~NOTE: Without a NAF requesting a run of AKA, GBA on it own does not guarantee the freshness of the key, Ks(_int/ext)_NAF. A NAF that support a Ua protocol that does not provide replay protection over unconnected runs of the protocol, will need to take some action to avoid replay attacks if desired. A possible ways to achieve this is to store all used Ks(_int/ext)_NAF to ensure that the key has not been used. If the key has been used then the NAF should request a new run of AKA to generate a new Ks(_int/ext).~~

NOTE: Without additional measures, GBA does not guarantee the freshness of the key, Ks(_int/ext)_NAF in the sense that it does not guarantee that the key was not used in a previous run of the Ua protocol. The additional measures which may be taken by the UE and the NAF to ensure key freshness in GBA are:
1) enforce a new run of the Ub protocol (thus generating a new Ks) before deriving a new Ks_NAF.
2) store previously used keys Ks(_int/ext)_NAF, or the corresponding key identifiers B-TID, until the end of their lifetime.
A UE and a NAF that support a Ua protocol that does not provide replay protection over unconnected runs of the protocol, will need to take corresponding action to avoid replay attacks if desired.