**3GPP TSG-SA3 Meeting #101-e *draft\_S3-203039-r4***

**e-meeting, 9 - 20 November 2020**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *CR-Form-v12.0* | | | | | | | | |
| **CHANGE REQUEST** | | | | | | | | |
|  | | | | | | | | |
|  | **33.220** | **CR** | **0207** | **rev** |  | **Current version:** | **16.2.0** |  |
|  | | | | | | | | |
| *For* [***HE******LP***](http://www.3gpp.org/3G_Specs/CRs.htm#_blank)*on using this form: comprehensive instructions can be found at* [*http://www.3gpp.org/Change-Requests*](http://www.3gpp.org/Change-Requests)*.* | | | | | | | | |
|  | | | | | | | | |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ***Proposed change affects:*** | UICC apps | **X** | ME | **X** | Radio Access Network |  | Core Network | **X** |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | | | | | |
| ***Title:*** | SHA-1 deprecation in GBA | | | | | | | | | |
|  |  | | | | | | | | | |
| ***Source to WG:*** | Huawei, Hisilicon | | | | | | | | | |
| ***Source to TSG:*** | S3 | | | | | | | | | |
|  |  | | | | | | | | | |
| ***Work item code:*** | GAB-ext | | | | |  | ***Date:*** | | | 2020-10-22 |
|  |  | | | |  | |  | | |  |
| ***Category:*** | **B** |  | | | | | ***Release:*** | | | TEI17 |
|  | *Use one of the following categories:* ***F*** *(correction)* ***A*** *(mirror corresponding to a change 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](http://www.3gpp.org/ftp/Specs/html-info/21900.htm). | | | | | | | | *Use one of the following releases: Rel-8 (Release 8) Rel-9 (Release 9) Rel-10 (Release 10) Rel-11 (Release 11) Rel-12 (Release 12)* *Rel-13 (Release 13) Rel-14 (Release 14) Rel-15 (Release 15) Rel-16 (Release 16)* | |
|  |  | | | | | | | | | |
| ***Reason for change:*** | | SHA-1 is known to be insecure and has been deprecated. 33.220 has deprecated the usage of SHA-1 in BSF certificate in 2010 (via contribution SP-100361).  Current 33.220 still uses SHA-1 to compuate MAC\* and MAC. This contribution proposes to introduce the usage of SHA-256 for MAC\* and MAC calculation, and proposes that SHA-1 shall only be used for backward compatibility. | | | | | | | | |
|  | |  | | | | | | | | |
| ***Summary of change:*** | | Adding the usage of SHA-256 for MAC\* and MAC computation. The use of SHA-1 is only for backward compatibility. | | | | | | | | |
|  | | 。 | | | | | | | | |
| ***Consequences if not approved:*** | | Usage of insecure algorithm | | | | | | | | |
|  | |  | | | | | | | | |
| ***Clauses affected:*** | | 5.3.2 | | | | | | | | |
|  | |  | | | | | | | | |
|  | | **Y** | **N** |  | | | |  | | |
| ***Other specs*** | |  | **X** | Other core specifications | | | | TS/TR ... CR ... | | |
| ***affected:*** | |  | **X** | Test specifications | | | | TS/TR ... CR ... | | |
| ***(show related CRs)*** | |  | **X** | O&M Specifications | | | | TS/TR ... CR ... | | |
|  | |  | | | | | | | | |
| ***Other comments:*** | |  | | | | | | | | |
|  | |  | | | | | | | | |
| ***This CR's revision history:*** | |  | | | | | | | | |

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Start of 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.



Figure 5.1: The bootstrapping procedure with UICC-based enhancements

A UE shall always include the product token "3gpp-gba-tmpi" in the user agent request-header field when communicating over Ub. A BSF shall always include the product token "3gpp-gba-tmpi" in the server response-header field when communicating over Ub.

NOTE a: According to the HTTP specification RFC 2616 [33], the product tokens may contain any text. They are ignored when unknown by a UE or a BSF.

1. The ME sends an HTTP request towards the BSF. When a TMPI associated with the IMPI in use is available on the UE, the UE includes this TMPI in the "username" parameter, otherwise the UE includes the IMPI.

2. The BSF recognises from the structure of the "username" parameter (cf. Annex B.4) whether a TMPI or an IMPI was sent. If a TMPI was sent the BSF looks up the corresponding IMPI in its local database. If the BSF does not find an IMPI corresponding to the received TMPI it returns an appropriate error message to the UE. The UE then deletes the TMPI and retries the request using the IMPI.

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.

The HSS shall also send an indication that the UICC supports SHA-256 to the BSF if the UICC supports SHA-256.

If the BSF implements the timestamp option and has a local copy of the GUSS for the subscriber that has been fetched from the HSS during a previous bootstrapping procedure, and this GUSS includes a timestamp, the BSF may include the GUSS timestamp in the request message. Upon receiving that timestamp, if the HSS implements the timestamp option, the HSS may compare it with the timestamp of the GUSS stored in the HSS. In this case, if and only if the HSS has done the comparison and the timestamps are equal, then the HSS shall send "GUSS TIMESTAMP EQUAL" indication to the BSF. In any other case, the HSS shall send the GUSS (if available) to the BSF. If the BSF receives "GUSS TIMESTAMP EQUAL" indication, it shall keep the local copy of the GUSS. In any other case, the BSF shall delete the local copy of the GUSS, and store the received GUSS (if sent).

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:

- The BSF computes MAC\*. If an indication that the UICC supports SHA-256 is received from the HSS, the MAC\* is computed as MAC\* = MACÅ Trunc(SHA-256(IK)); otherwise, MAC\* = MACÅ Trunc(SHA-1(IK)) .

NOTE 1: Trunc denotes that from the output of SHA-256 or SHA‑1 [23], 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.

NOTE 2: In a multiple HSS environment, the BSF may have to obtain the address of the HSS where the subscription of the user is stored by querying the SLF, prior to step 2.

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 UICC calculates IK and MAC (by performing MAC= MAC\* ⊕ Trunc(SHA-256(IK)) if the UICC supports SHA-256, otherwise 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.

NOTE X: The usage of SHA-256 for MAC\* computation at BSF and MAC calculation at UICC is recommended. The usage of SHA-1 is only for backward compatibility.

NOTE Y: BSF and HSS need to be upgraded to use SHA-256 before a UICC supporting SHA-256 can be used.

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.

NOTE 3: The password in "AKAv1" HTTP Digest AKA is in binary format.

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.

NOTE 3a: If the HSS/AuC uses a good random number generator, then the chance of a B-TID collision is practically zero. If such a collision occurs, then the key retrieved by the NAF can have a mismatch with the UE generated NAF key. This will result in a Ua authentication failure which will cause the NAF to once again request the UE to bootstrap which will create a new Ks and a new B-TID.

If the request included the product token "3gpp-gba-tmpi" in the user agent request-header field the BSF shall compute a new TMPI as specified in Annex B.4 and store it together with the IMPI, overwriting a previous TMPI related to this IMPI, if any.

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, "gba-me", RAND, IMPI, NAF\_Id), and Ks\_int\_NAF is computed in the UICC as Ks\_int\_NAF = KDF(Ks, "gba-u, RAND, IMPI, NAF\_Id), 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 is constructed as follows: NAF\_Id = FQDN of the NAF || Ua security protocol identifier. The Ua security protocol identifier is specified in Annex H. 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 4: If a NAF hosts two or more applications which use the same FQDN and Ua security protocol identifier, they will share the same NAF specific keys. This causes a risk of so called two-time pad which may lead to the situation that the security of these applications is compromised. This can be avoided by running bootstrapping separately to each application or by application specific means, which are however out of the scope of the current specification.

To allow consistent key derivation based on NAF name in UE and BSF, at least one of the prerequisites which are specified in clause 4.5.2 shall be met.

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 or until the deletion conditions are satisfied (see 4.4.11).

If the response included the product token "3gpp-gba-tmpi" in the server response-header field the UE shall compute the TMPI as specified in Annex B.4 and store it together with the IMPI, overwriting a previous TMPI related to this IMPI, if any.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* End of Change \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*