3GPP TSG SA WG3 Security#32 **S3-040161**
9 – 13 February, 2004
Edinburgh, UK

Source: **Siemens**

Title: **Multiple key derivation in a Generic Bootstrapping Architecture - Pseudo-CR**

Agenda Item: **6.9.2 (GBA)**

Document for: **Discussion and decision**

Abstract

In the current version of the Generic Bootstrapping Architecture specification (TS 33.220 v100), precisely one key Ks_NAF is derived from one key Ks. This contribution proposes to allow the derivation of multiple keys Ks_NAF for different NAFs from one Ks. It is shown that the adoption of this proposal would significantly reduce the number of requests on the HSS over the Zh interface, while the increase in complexity would be small. Any potential security risks are addressed by introducing a key life-time parameter. A pseudo-CR to TS 33.220 v100 is also provided in this contribution.

1. Reason for proposed change to TS 33.220 v100

Current situation: In TS 33.220 v100, a NAF-specific key Ks_NAF for user over the Ua interface is derived from a key Ks established between the UE and the BSF. This key derivation was introduced in order thwart an attack described in contribution S3-030552, where one NAF could impersonate another. TS 33.220 v100 further specifies that only one Ks_NAF shall be derived from one Ks.

Problem: There is a potentially significant performance disadvantage incurred by using no key derivation at all, or by allowing the derivation of only one key Ks_NAF from one key Ks, as is shown by the following consideration:
Browsing: a user may access a larger number of application servers in rapid succession when looking for information or technical support;
Gaming: a user may try one game with a few clicks, not like it and continue to the next game server within a relatively short time.
More examples of the kind would be possible.
But with the current specification, for each new contact with a NAF, a new run of http digest aka over the Ub interface is required, and a new authentication vector needs to be generated by the HSS. This leads to unnecessary load on the HSS and the BSF. Our proposal tries to avoid this disadvantage.

2. Proposal

In order to overcome the performance disadvantage described in section 1, the following is proposed:

- When the UE accesses the first NAF1, the procedure is as described in TS 33.220 v100. However, the UE and the BSF store the key Ks with the associated transaction identifier TID for further use, even after Ks_NAF1 was derived.

- When the UE accesses a second NAF2, the UE sends the stored TID to the NAF2, and the UE and the BSF use the stored Ks to derive Ks_NAF2. There is no need for a new run of the protocol over the Ub interface.

- The UE continues to use the stored key Ks for further derivations of keys Ks_NAF with further NAFs until the key Ks is required to be updated.

- The key Ks is required to be updated when its lifetime has expired, or when a NAF requests a key update (according to TS 33.220 v100, section 4.3.3)
- The key Ks is updated in a new run of the protocol over the Ub interface with the BSF. When the protocol run is complete, the old Ks is replaced by the new Ks in both, UE and BSF. The keys Ks_NAF stored in the UE and in the NAFs are not affected by this update of Ks (cf. also companion contribution on key handling).
- In order for the proposed procedure to be efficient the lifetime of Ks in the UE shall be less or equal the lifetime of Ks in the BSF. In order to ensure this it is proposed that the BSF communicates the lifetime of Ks to the UE in the 200 OK message over the Ub interface, together with the TID. In addition the BSF shall indicate to the UE whether multiple key derivation is allowed to be used. The transport format used for the TID can also be used for the key lifetime and this indication, see the XML schema provided in Nokia's CN1 contribution N1-040086.

3. Evaluation of the proposal

Security: the key Ks would normally be available in the UE and in the BSF for a longer period than in the current version of the specification. However, the security risk introduced by a possible compromise of the key Ks is mitigated by the following factors:

- The lifetime of Ks can be controlled by both, BSF an UE;
- Ks is deleted when the UE is powered down, or when the UICC is removed(cf. companion contribution on key handling);
- Ks is not directly used over the Ua interface;
- Ks is not known to any NAF;

Please also note that a Trojan horse on the UE could also be used for a successful attack in the current version of the specification.
Therefore the additional security risk seems limited, provided the lifetime of Ks is reasonably limited.

Number of protocol runs: when the UE accesses m different NAFs within the lifetime of the key Ks then the number of requests from the UE to the BSF and the consumption of authentication vectors can be reduced by the factor m. This seems quite significant.

Storage: both, the UE and the BSF, have to store Ks until the end of its specified lifetime. This seems not a major problem.

4. Pseudo-CR

4.2.2.1 Bootstrapping server function (BSF)

A generic bootstrapping server function (BSF) and the UE shall mutually authenticate using the AKA protocol, and agree on session keys that are afterwards applied between UE and an operator-controlled network application function (NAF). ~~The key material must be generated specifically for each NAF independently, that is, for each key uniquely identified by a transaction identifier and that is shared between a UE and a NAF there is a new run of HTTP Digest AKA over the Ub interface.~~ The BSF can restrict the applicability of the key material to a defined set of NAFs by using a suitable key derivation procedure. The generation of key material is specified in section 4.3.2.

Editor's note: Key generation for NAF is ffs. Potential solutions may include:
- Separate run of HTTP Digest AKA over Ub interface for each request of key material from a NAF
- Issues with key lifetime are ffs.

4.3.2 Bootstrapping procedures

When a UE wants to interact with an NAF, and it knows that bootstrapping procedure is needed, it shall first perform a bootstrapping authentication (see figure 4)

Editor's note: Zh interface related procedure will be added here in future development. It may re-use Cx interface that is specified in TS 29.228.

Otherwise, the UE shall perform a bootstrapping authentication only when it has received bootstrapping initiation required message or a key update indication from the NAF, or when the lifetime of the key has expired ~~-~~(cf. subclause 4.3.3).

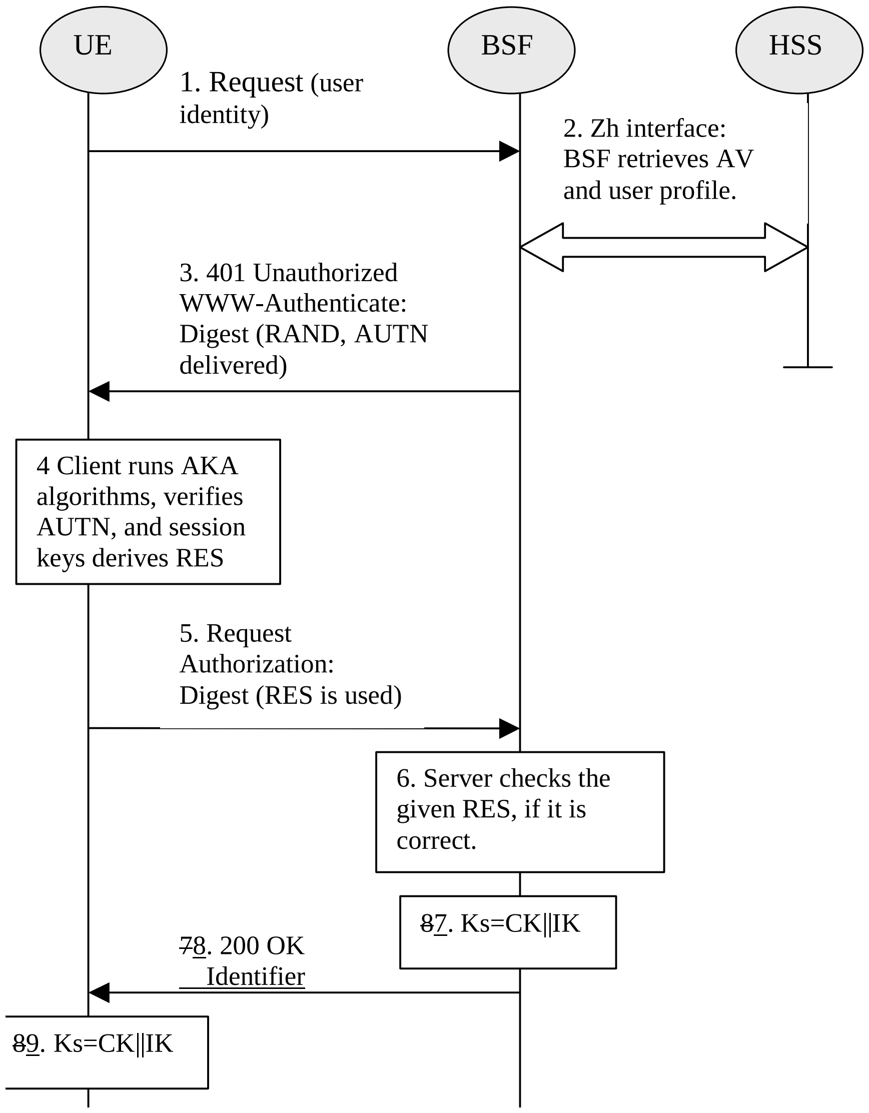
UE
BSF
HSS
1. Request (user identity)
2. Zh interface: BSF retrieves AV and user profile.
3. 401 Unauthorized WWW-Authenticate: Digest (RAND, AUTN delivered)
4 Client runs AKA algorithms, verifies AUTN, and session keys derives RES
5. Request Authorization: Digest (RES is used)
6. Server checks the given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

Figure 4: The bootstrapping procedure

1. The UE sends an HTTP request towards the BSF.
2. BSF retrieves the user profile and a challenge, i.e. the Authentication Vector (AV, AV = RAND||AUTN||XRES||CK||IK) over Zh interface from the HSS.
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 calculates the message authentication code (MAC) so as to verify the challenge from authenticated 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 request again, with the Digest AKA RES as the response to the BSF.
6. If the RES equals to the XRES that is in the AV, the UE is authenticated.

7. BSF generates key material Ks by concatenating CK and IK. Ks is used to derive the key material Ks_NAF. Ks_NAF is used for securing the Ua interface.

8. The BSF shall send 200 OK message and shall supply a transaction identifier to the UE to indicate the success of the authentication. The BSF may also supply the parameter *n* used to determine the NAF_Id_n (cf. previous bullet) to the UE over the Ub interface. If the parameter *n* is not supplied then no key derivation is performed, i.e. Ks = Ks_NAF. If key derivation is performed it is to be applied uniformly to all keys shared between any UE and any NAF. In addition, in the 200 OK message, the BSF shall supply the lifetime of the key Ks, and an indication whether multiple key derivation ~~may~~shall be used.

9. The key material Ks is generated in UE by concatenating CK and IK. The Ks is used to derive the key material Ks_NAF. Ks_NAF is used for securing the Ua interface.

Ks_NAF is computed as Ks_NAF = KDF (Ks, key derivation parameters), where KDF is a suitable key derivation function, and the key derivation parameters include the user's IMSI, the NAF_Id_n and RAND. The NAF_Id_n consists of the n rightmost domain labels in the DNS name of the NAF, separated by dots (n= 1, ..., 7). For n = 0, NAF_Id_n equals the full DNS name of the NAF. The next bullet specifies how the UE obtains n.

NOTE: This note gives an example how to obtain the NAF_Id_n: if the DNS name of the NAF is "server1.presence.bootstrap.operator.com", and n = 3, then NAF_Id_n = " bootstrap.operator.com".

Editor's note: The definition of the KDF and the possible inclusion of further key derivation parameters is left to ETSI SAGE.

If multiple key derivation is used then t~~T~~he UE and the BSF store the key Ks with the associated transaction identifier TID for further use, until the lifetime of Ks has expired, or until the key Ks is updated. Otherwise, the key Ks and the TID may be deleted in the UE and in the BSF after the key Ks_NAF has been derived.

4.3.3 Procedures using bootstrapped Security Association

After UE is authenticated with the BSF, every time the UE wants to interact with an NAF the following steps are executed as depicted in figure 5.

UE starts communication over Ua interface with the NAF

- In general, UE and NAF will not yet share the key(s) required to protect Ua interface. If they already do (i.e. if a key Ks_NAF for the corresponding key derivation parameter NAF_Id_n is already available), ~~there is no need for NAF to retrieve the key(s) over Zn interface~~the UE and the NAF can start to securely communicate right away. If the UE and the NAF do not yet share a key, the UE proceeds as follows:
  if a key Ks is available in the UE, the UE derives the key Ks_NAF from Ks, as specified in clause 4.3.2:
  if no key Ks is available in the UE, the UE first ~~fetches~~agrees on a new key Ks ~~from~~with the BSF over the Ub interface, ~~it~~and then proceeds to derive Ks_NAF.

- If the NAF shares a key with the UE, but an update of that key is required, it sends a suitable key update request to the UE and terminates the protocol used over Ua interface. The form of this indication may depend on the particular protocol used over Ua interface and is ffs.

- It is assumed that UE supplies sufficient information to NAF, e.g. a transaction identifier, to allow the NAF to retrieve specific key material from BSF.

- The UE derives the keys required to protect the protocol used over Ua interface from the key material, as specified in clause 4.3.2.

NOTE 1: The UE may adapt the key material Ks_NAF to the specific needs of the Ua interface. This adaptation is outside the scope of this specification.

NAF starts communication over Zn interface with BSF

- The NAF requests key material corresponding to the information supplied by the UE to the NAF (e.g. a transaction identifier) in the start of the protocol used over Ua interface.

- The BSF derives the keys required to protect the protocol used over Ua interface from the key material and the key derivation parameters, as specified in clause 4.3.2, and supplies to NAF the requested key material. If the key identified by the transaction identifier supplied by the NAF is not available at the BSF, the BSF shall indicate this in the reply to the NAF. The NAF then indicates a key update request to the UE.

NOTE 2: The NAF may adapt the key material Ks_NAF to the specific needs of the Ua interface in the same way as the UE did. This adaptation is outside the scope of this specification.

NAF continues with the protocol used over Ua interface with UE.

Once the run of the protocol used over Ua interface is completed the purpose of bootstrapping is fulfilled as it enabled UE and NAF to use Ua interface in a secure way.

Editor's note: Message sequence diagram presentation and its details will be finalized later.

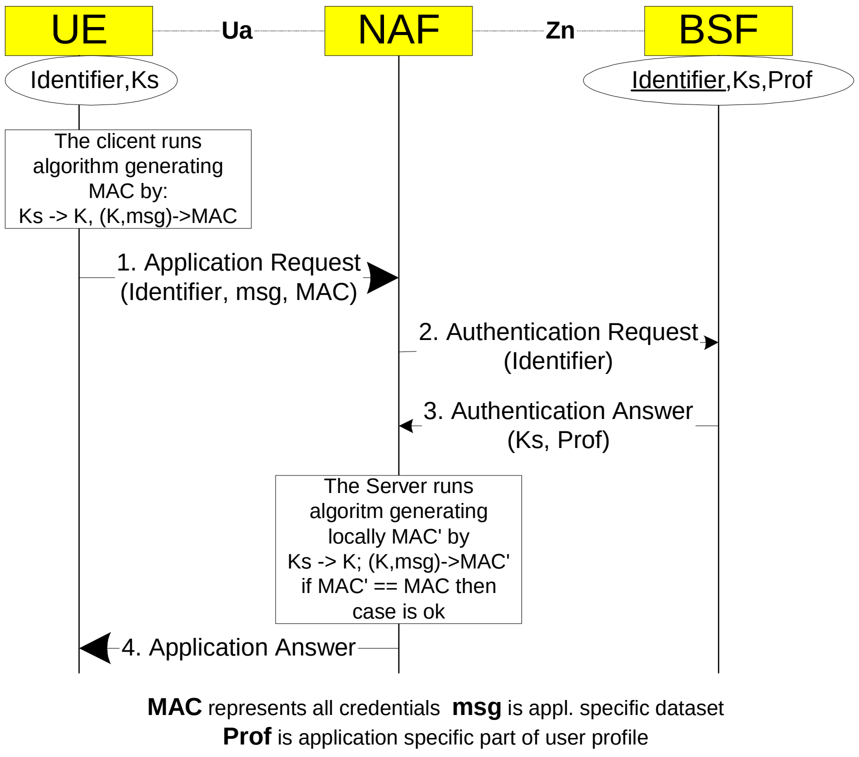
UE
Ua
NAF
Zn
BSF
Identifier,Ks
Identifier,Ks,Prof
The clicent runs algorithm generating MAC by: Ks -> K, (K,msg)->MAC
1. Application Request (Identifier, msg, MAC)
2. Authentication Request (Identifier)
3. Authentication Answer (Ks, Prof)
The Server runs algoritm generating locally MAC' by Ks -> K; (K,msg)->MAC' if MAC' == MAC then case is ok
4. Application Answer
MAC represents all credentials msg is appl. specific dataset
Prof is application specific part of user profile

Figure 5: The bootstrapping usage procedure