3GPP TSG SA WG3 Security — S3#32 **S3-040076**
9 - 13 February 2004
Edinburgh, UK

Source: **Nokia**

Title: **UE triggered unsolicited push from BSF to NAF**

Document for: **Discussion/Decision**

Agenda Item: **6.9.2 (TS 33.220)**

1. INTRODUCTION

This contribution is a follow up to Nokia contribution [S3-030729], which was presented in Munich meeting in November 2003. It discussed a possibility for UE to trigger BSF to do an unsolicited push of bootstrapping information, i.e. transaction identifier (TID), NAF specific shared secret (Ks_naf), and optional subscriber profile information, to a NAF. This would simplify procedures during shared secret usage over Ua interface (between UE and NAF) since if NAF already has received the bootstrapping information, it does not need to use Zn interface to fetch this information. Further more, it would also decrease the processing load on BSF since after BSF has delivered the bootstrapping information to NAF, it need not keep the bootstrapping information in its local database.

Several concerns were raised in the Munich meeting related to the unsolicited push procedure. This contribution updates the procedure that was introduced in [S3-030729] in section 2 and addresses these concerns in section 3.2.

2. DISCUSSION

2.1 Use case: Triggering during bootstrapping procedure

In order for UE to trigger BSF to do an unsolicited push of bootstrapping information to one NAF, the NAF_ID is inserted to the initial bootstrapping request. In the list there can be zero or one NAF_ID present. NAF_ID is known by the BSF so that it knows to which NAF the bootstrapping information is to be pushed.

Note: This is the “use case 1” listed in the original contribution [S3-030729] with a change that the bootstrapping information is pushed to a single NAF, not to a list of NAFs. This also the reason why the “use case 2” listed in the original contribution is not applicable any more. These changes were necessary because the bootstrapping procedure could be done everytime when UE contacts a NAF and that NAF asks for GBA-based authentication; hence the usage of single bootstrapping information is limited to one NAF.

2.2 NAF_ID considerations

Requirements for NAF_ID:

- NAF_ID shall be in a format, which is easily discovered or known by the UE.
- NAF_ID shall be globally unique to identify a NAF.

Solution for NAF_ID format:

- Fully qualified domain name (FQDN) consists of a host and domain name, including top-level domain of the NAF, e.g. “presence.operator.com”. An advantage of this NAF_ID format is that it uniquely identifies the NAF and is easily discovered or known by the UE.

2.3 Procedure details

Figure 1 describes the bootstrapping procedure where UE triggers the pushing of bootstrapping information from BSF to a NAF.

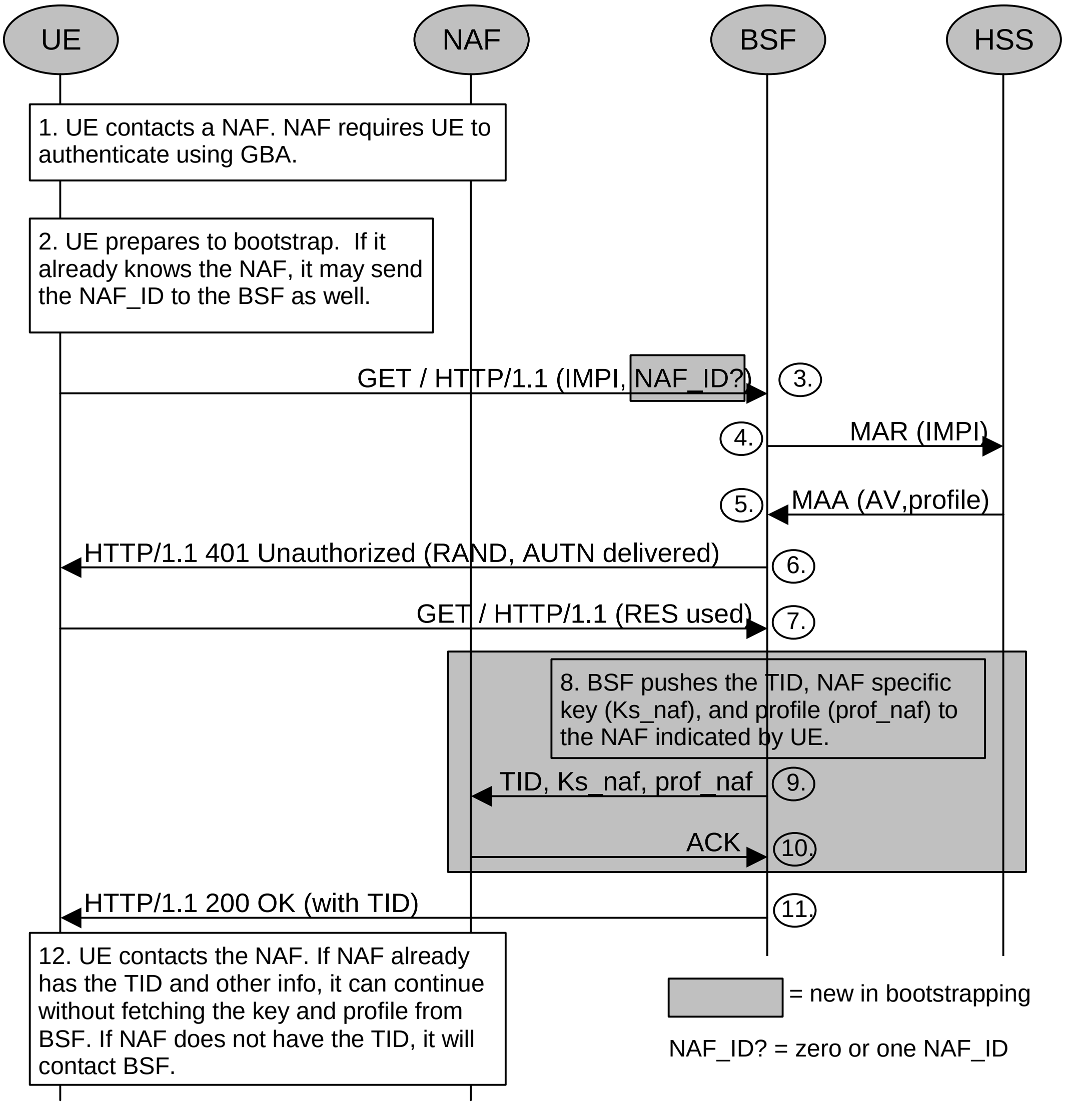

UE
NAF
BSF
HSS
1. UE contacts a NAF. NAF requires UE to authenticate using GBA.
2. UE prepares to bootstrap. If it already knows the NAF, it may send the NAF_ID to the BSF as well.
GET / HTTP/1.1 (IMPI, NAF_ID?)
3.
4.
MAR (IMPI)
5.
MAA (AV,profile)
HTTP/1.1 401 Unauthorized (RAND, AUTN delivered)
6.
GET / HTTP/1.1 (RES used)
7.
8. BSF pushes the TID, NAF specific key (Ks_naf), and profile (prof_naf) to the NAF indicated by UE.
TID, Ks_naf, prof_naf
9.
ACK
10.
HTTP/1.1 200 OK (with TID)
11.
12. UE contacts the NAF. If NAF already has the TID and other info, it can continue without fetching the key and profile from BSF. If NAF does not have the TID, it will contact BSF.
= new in bootstrapping
NAF_ID? = zero or one NAF_ID

Figure 1. UE triggered push of bootstrapping information from BSF to a NAF.

1. (Optional) UE contacts a NAF. NAF indicates to the UE that it requires authentication and that GAA should be used.

2. UE prepares to do bootstrapping procedure. If UE already knows the NAF, it may add the NAF_ID in the initial bootstrapping message 3.

3. UE sends initial HTTP request to BSF with IMPI, and optional NAF_ID. (“NAF_ID?” means that NAF_ID parameter may be either absent or present in the message.)

4-5. (Optional) BSF fetches authentication vectors AV and profile information from HSS. Optionally, if BSF already has AV for the UE it may skip steps 4 and 5 all together.

6-7. Ordinary HTTP Digest AKA steps are done and new bootstrapping info is established.

Note: Steps 8-10 are optional and are done only if the optional NAF_ID was present in the initial bootstrapping request in step 3.

8. (Optional) If NAF_ID was present in the initial HTTP request (step 2), then BSF pushes the bootstrapping info to the NAF indentified by NAF_ID.

9. (Optional) NAF specific bootstrapping info is pushed to the NAF by BSF.

10. (Optional) NAF acknowledges that the bootstrapping info was received and stored.

11. HTTP response 200 OK with TID is sent to the UE to indicate that the bootstrapping procedure was successful. There is no indication whether the optional unsolicited push operation was successful or not.

12. UE contacts the NAF. If the NAF already possesses the bootstrapping info identified by the TID, it does not need to fetch the bootstrapping info from BSF over Zn interface. If the bootstrapping info is not present in the NAF, it fetches the info from BSF over Zn interface.

3. ANALYSIS

3.1 Updated analysis (compared to S3-030729)

Attacks

NAF_ID is not integrity protected in the initial bootstrapping request (step 3 in Figure 1). Thus, an active attacker may have opportunities to change the NAF_ID as well as the IMPI value. A change of those values by an active attacker causes a denial of service attack against UE without permanent effect, since BSF would send the bootstrapping info only to valid and well-known NAF, and ignore bad or unknown NAF_ID. Afterwards, when UE accesses a NAF, which has not received the bootstrapping info, NAF would fetch this information from BSF over Zn interface as specified in [TS GBA].

Simplification of procedures

If the unsolicited push of bootstrapping information to a NAF is done, it simplifies the procedures in both BSF and NAF:

- during bootstrapping over Ub interface, BSF pushes the bootstrapping information to the NAF, and there is no need to store the bootstrapping information in BSF.
- during the usage of bootstrapping information over Ua interface, NAF does not need to fetch the information from BSF over Zn interface, because the bootstrapping information (i.e., TID, shared secret, optional profile information) would already be in its local database.

Also, the amount of computatuions in BSF caused by database usage can be reduced if the unsolicited push of bootstrapping information is used. During normal operation where NAF would fetch the bootstrapping information from BSF, there are several procedures related to database usage that BSF and NAF must do:

1. During bootstrapping: BSF has *inserted* the bootstrapping information to its database.
2. NAF *searches* its local database for bootstrapping information.
3. NAF requests the bootstrapping information from BSF using the TID.
4. BSF *searches* its database for the bootstrapping information and optionally does key derivation if needed.
5. BSF *deletes* the bootstrapping information entry from its database.
6. BSF returns the bootstrapping information to NAF.
7. NAF *stores* bootstrapping information to the local database.
8. NAF uses the bootstrapping information to authenticate the UE.

9. After Ua interface usage, NAF *deletes* the bootstrapping information from its local database.

For unsolicited push, the procedures related to database usage are:

1. During bootstrapping: BSF optionally does key derivation if needed and pushes the bootstrapping information to the NAF, which NAF *stores* to its local database. If push is successful, BSF does not store the bootstrapping information to its local database, and just deletes the information after bootstrapping procedure is over.

2. NAF *searches* its local database for bootstrapping information.

3. NAF uses the bootstrapping information to authenticate the UE.

4. After Ua interface usage, NAF *deletes* the bootstrapping information from its local database.

As can be seen from the above steps and from Table 1 below, using unsolicited push procedure, the database usage in general and especially in BSF is diminished compared to the normal bootstrapping procedure. Further more, if the unsolicited push would be mandatory, the BSF would not need database for storing the bootstrapping information at all.

Database operations	**without push**	**with push**
Inserts	1 in BSF, 1 in NAF	1 in NAF
Searches	1 in BSF, 1 in NAF	1 in NAF
Deletes	1 in BSF, 1 in NAF	1 in NAF

Table 1. Database operations during GAA usage with or without unsolicited push.

3.2 Concerns raised in S3#31

This chapter addresses concerns raised in S3#31 meeting.

Bad UEs may cause network load

A concern was raised in the last meeting whether bad UEs may cause network load if they are able to trigger unsolicited push of bootstrapping information to NAF or list of NAFs.

The concern is valid, if UE would be allowed to send a list of NAF_IDs to BSF as was proposed in [S3-030729]. However, during the meeting it was clarified that bootstrapping is done everytime a new NAF is contacted and that bootstrapped security association (i.e., TID, and the NAF specific session key Ks_NAF) is used with only one NAF. Thus, UE can only trigger the unsolicited push of bootstrapping to one NAF – not to a list of NAFs. If UE tries to send a list of NAF_ID to the BSF, BSF should ignore such a request.

Authentication method decision

It was unclear in the previous contribution [S3-030729], who does the authentication method decision: UE or application server.

As can been seen from the updated sequence diagram (see Figure 1), it can be either application server itself (i.e., NAF if GBA is used), or UE itself if it knows beforehand that GBA is acceptable with application server (e.g., PKI portal in subscriber certificate enrollment case).

HTTP Digest AKA integrity protection is too weak

A concern was raised in the last meeting whether the password used in HTTP Digest AKA (i.e., XRES, which can be as short as 32 bits) may be too weak. However, an active attacker that discovers the password still cannot, e.g., discover the secret shared between

UE and the network. It is able only to replace TID (in message11) and cause DoS attack on UE without permanent effect.

4. PROPOSAL

We propose to add the unsolicited push mechanism described in this contribution to the bootstrapping procedure described in 3GPP TS 33.220 [TS GBA]. There is a pseudo CR attached to this contribution implementing the required changes on the TS.

REFERENCES

[TS GBA] Draft 3GPP TS 33.220: “Generic Authentication Architecture (GAA); Generic Bootstrapping Architecture”, Rel-6.

[S3-030729] “UE triggered unsolicited push from BSF to NAFs”, Nokia, S3#31, Link: S3-030729.

***** BEGIN CHANGE *****

4 Generic Bootstrapping Architecture

The 3GPP authentication infrastructure, including the 3GPP Authentication Centre (AuC), the USIM, and the 3GPP AKA protocol run between them, is a very valuable asset of 3GPP operators. It has been recognised that this infrastructure could be leveraged to enable application functions in the network and on the user side to communicate in situations where they would not be able to do so without the support of the 3GPP authentication infrastructure. Therefore, 3GPP can provide the “bootstrapping of application security” to authenticate the subscriber by defining a generic bootstrapping function based on AKA protocol.

4.1 Requirements and principles for bootstrapping

Editor’s note: The description of AKA bootstrapping shall be added here.

- The bootstrapping function shall not depend on the particular network application function

- The server implementing the bootstrapping function needs to be trusted by the home operator to handle authentication vectors.

- The server implementing the network application function needs only to be trusted by the home operator to handle derived key material.

- It shall be possible to support network application functions in the operator’s home network

- The architecture shall not preclude the support of network application function in the visited network, or possibly even in a third network.

- To the extent possible, existing protocols and infrastructure should be reused.

- In order to ensure wide applicability, all involved protocols are preferred to run over IP.

- It shall be prevented that a security breach in one application server using the Generic Bootstrapping Architecture can be used by an attacker to mount successful attacks to the other application servers using the Generic Bootstrapping Architecture.

4.1.1 Access Independence

Bootstrapping procedure is access independent. Bootstrapping procedure requires IP connectivity from UE.

4.1.2 Authentication methods

Authentication method that is used to authenticate the bootstrapping function must be dependent on cellular subscription. In other words, authentication to bootstrapping function shall not be possible without valid cellular subscription. Authentication shall be based on AKA protocol.

4.1.3 Roaming

The roaming subscriber shall be able to utilize the bootstrapping function in home network.

Editor’s note: For the first phase of standardisation, only the case is considered where bootstrapping server functionality and network application function are located in the same network as the HSS. In later phases, other configurations may be considered.

4.1.4 Requirements on Ub interface

The requirements for Ub interface are:

- The BSF shall be able to identify the UE.

- The BSF and the UE shall be able to authenticate each other based on AKA.

- The BSF shall be able to send a transaction identifier to UE.

- The UE shall be able to indicate to the BSF, that it wants to push key material and subscriber profile during bootstrapping procedure to a NAF identified by a NAF identifier (NAF_ID).

Editor's note: The format of NAF_ID is ffs. NAF_ID must uniquely identity a NAF and it must be easily discovered by the UE. One solution for NAF identifier is fully qualified domain name (FQDN).

4.1.5 Requirements on Zh interface

The requirements for Zh interface are:

- The BSF shall be able to communicate securely with the subscriber's HSS.

Editor's note: this requirement is fulfilled automatically if BSF and HSS are in same operator's network.

- The BSF shall be able to send bootstrapping information request concerning a subscriber.

- The HSS shall be able to send authentication vectors to the BSF in batches.

- The HSS shall be able to send the subscriber's GAA profiles to the BSF.

Editor's note: the intention is not to send all the application-specific profile information, but only the information needed for security purposes.

Editor's note: it's ffs how to proceed in the case where profile is updated in HSS after profile is forwarded. The question is whether this profile change should be propagated to BSF.

- No state information concerning bootstrapping shall be required in the HSS.

- All procedures over Zh interface shall be initiated by the BSF.

- It is preferred to reuse existing specifications if possible.

- The number of different interfaces to HSS should be minimized.

4.1.6 Requirements on Zn interface

The requirements for Zn interface are:

- Mutual authentication, confidentiality and integrity shall be provided.

- The BSF shall verify that the NAF is authorised.

- The NAF shall be able to send a key material request to the BSF.

- The BSF shall be able to send the requested key material to the NAF.

- The NAF shall be able to get the subscriber profile from BSF.

- The BSF shall be able to push key material and subscriber profile to the NAF.

Editor's note: the intention is not to send all the application-specific profile information, but only the information needed for security purposes.

Editor's note: in later phases there is an additional requirement that the NAF and the BSF may be in different operators' networks.

4.2 Bootstrapping architecture

4.2.1 Reference model

Figure 1 shows a simple network model of the entities involved in the bootstrapping approach, and the protocols used among them.

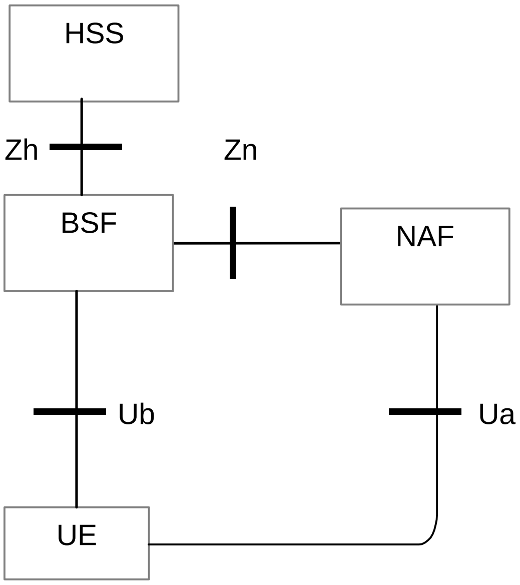


Figure 1: Simple network model for bootstrapping

Figure 2 illustrates a protocol stacks structure in network elements that are involved in bootstrapping of application security from 3G AKA and support for subscriber certificates.

Editor’s note: The current protocol stack figure is placed here as a holder. The actual protocols will be defined later.

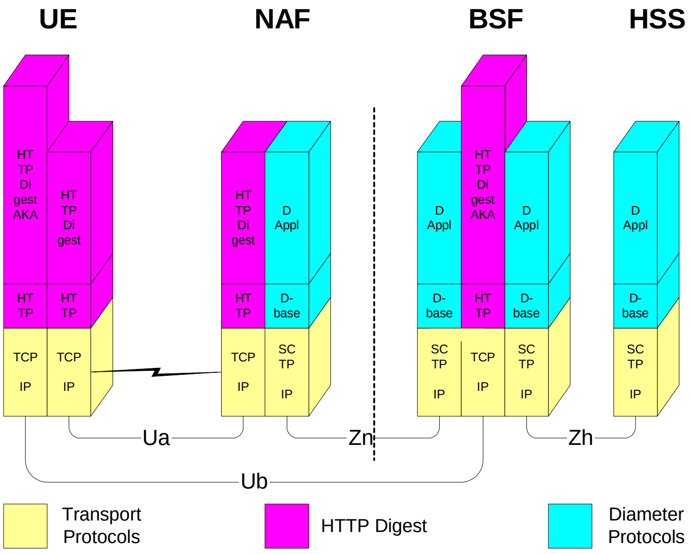


Figure 2: Protocol stack architecture

4.2.2 Network elements

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.

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.

4.2.2.2 Network application function (NAF)

After the bootstrapping has been completed, the UE and an operator-controlled network application function (NAF) can run some application specific protocol where the authentication of messages will be based on those session keys generated during the mutual authentication between UE and BSF.

General assumptions for the functionality of an operator-controlled network application function (NAF):

- There is no previous security association between the UE and the NAF.
- NAF shall able to locate and communicate securely with subscriber's BSF.
- NAF shall be able to acquire a shared key material established between UE and the bootstrapping server function (BSF) during running application-specific protocol.
- BSF shall be able to push a shared key material established between UE and the BSF to NAF before running application-specific protocol.

4.2.2.3 HSS

HSS shall store new parameters in subscriber profile related to the usage of bootstrapping function. Possibly also parameters related to the usage of some network application function are stored in HSS.

Editor's note: Needed new parameters are FFS.

4.2.2.4 UE

The required new functionalities from UE are:

- The support of HTTP Digest AKA protocol,
- The capability to derive new key material to be used with the protocol over Ua interface from CK and IK, and
- Support of NAF specific application protocol (see [5]).

4.2.3 Reference points

4.2.3.1 Ub interface

The reference point Ub is between the UE and the BSF. The functionality is radio access independent and can be run in both CS and PS domains.

Editor's notes: The solution for CS domain is ffs.

4.2.3.1.1 Functionality

Reference point Ub provides mutual authentication between the UE and the BSF entities. It allows the UE to bootstrap the session keys based on the 3G infrastructure. The session key as result of key agreement functionality, is used to support further applications e.g. certificate issuer.

4.2.3.1.2 Protocol

Ub interface is in format of HTTP Digest AKA, which is specified in [4]. It is based on the 3GPP AKA [2] protocol that requires information from USIM and/or ISIM. The interface to the USIM is as specified for 3G [1].

4.2.3.2 Ua interface

Ua interface is the application protocol which is secured using the keys material agreed between UE and BSF as a result of the run of HTTP Digest AKA over Ub interface. For instance, in the case of support for subscriber certificates [5], it is a protocol, which allows the user to request certificates from the NAF. In this case NAF would be the PKI portal.

4.2.3.3 Zh interface

Zh interface is used between the BSF and the HSS to allow the BSF to fetch the required authentication information and subscriber profile information from the HSS. The interface to the 3G Authentication Centre is HSS-internal, and it need not be standardised as part of this architecture.

4.2.3.4 Zn interface

Zn interface is used by the NAF to fetch the key material agreed during previous HTTP Digest AKA protocol run over Ub interface from the BSF. Zn interface is also used by the BSF to push the key material agreed during HTTP Digest AKA protocol run over Ub interface to a NAF. It may also be used to fetch subscriber profile information from BSF.

4.3 Procedures

This chapter specifies in detail the format of the bootstrapping procedure that is further utilized by various applications. It contains the AKA authentication procedure with BSF, and latter the key material generation procedure.

4.3.1 Initiation of bootstrapping

When a UE wants to interact with an NAF, but it does not know if bootstrapping procedure is required, it shall contact NAF for further instructions (see Figure 3).

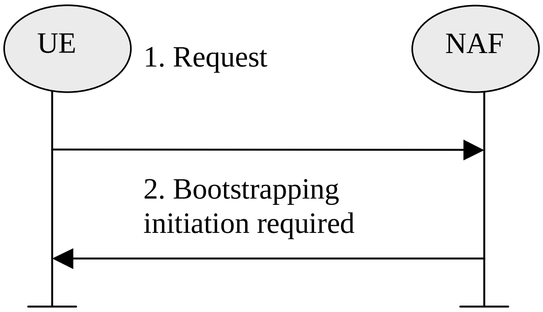


Figure 3: Initiation of bootstrapping

1. UE starts communication over Ua interface with the NAF without any bootstrapping related parameters.
2. If the NAF require bootstrapping but the request from UE does not include bootstrapping related parameters, NAF replies with a bootstrapping initiation message. The form of this indication may depend on the particular Ua interface and is ffs.

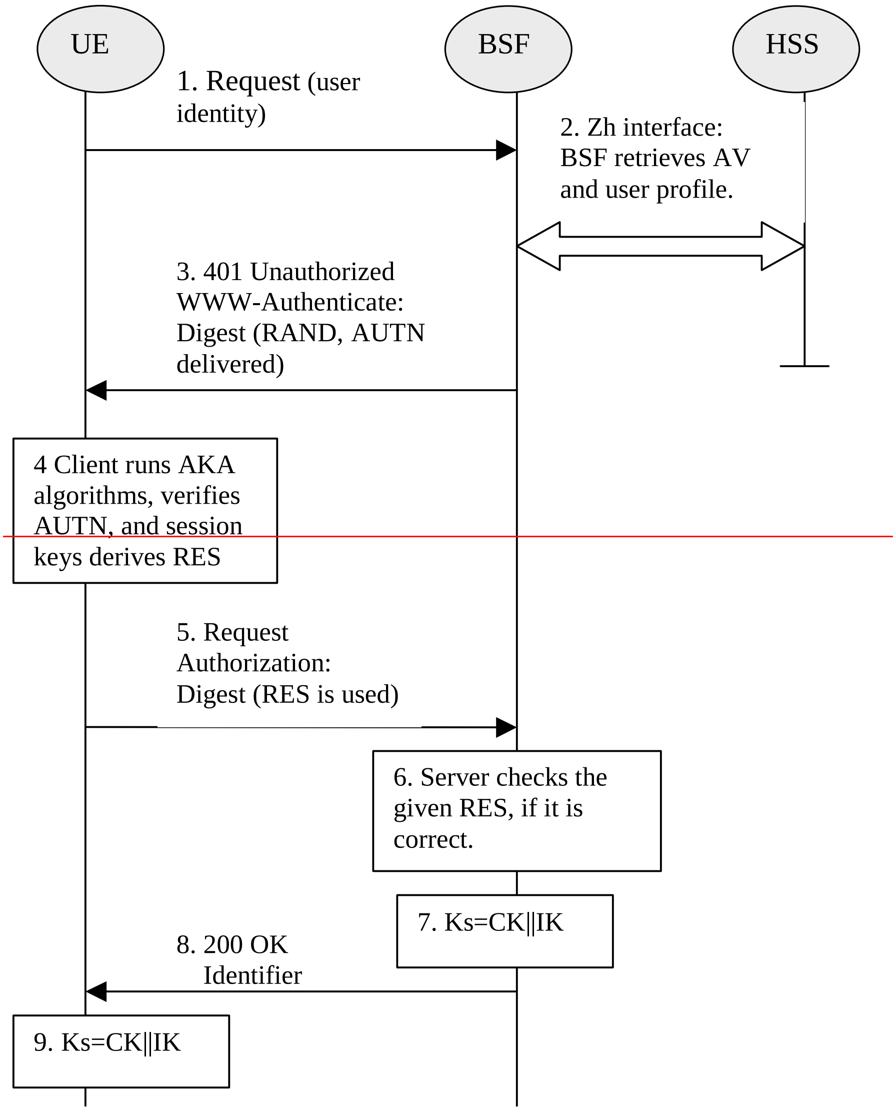
Editor’s notes: If the protocol over Ua interface is based on HTTP, then NAF can initiate the bootstrapping procedure by using HTTP status codes (e.g. 401 Unauthorized).

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 notes: 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 (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.
7. Ks=CK||IK
8. 200 OK Identifier
9. Ks=CK||IK

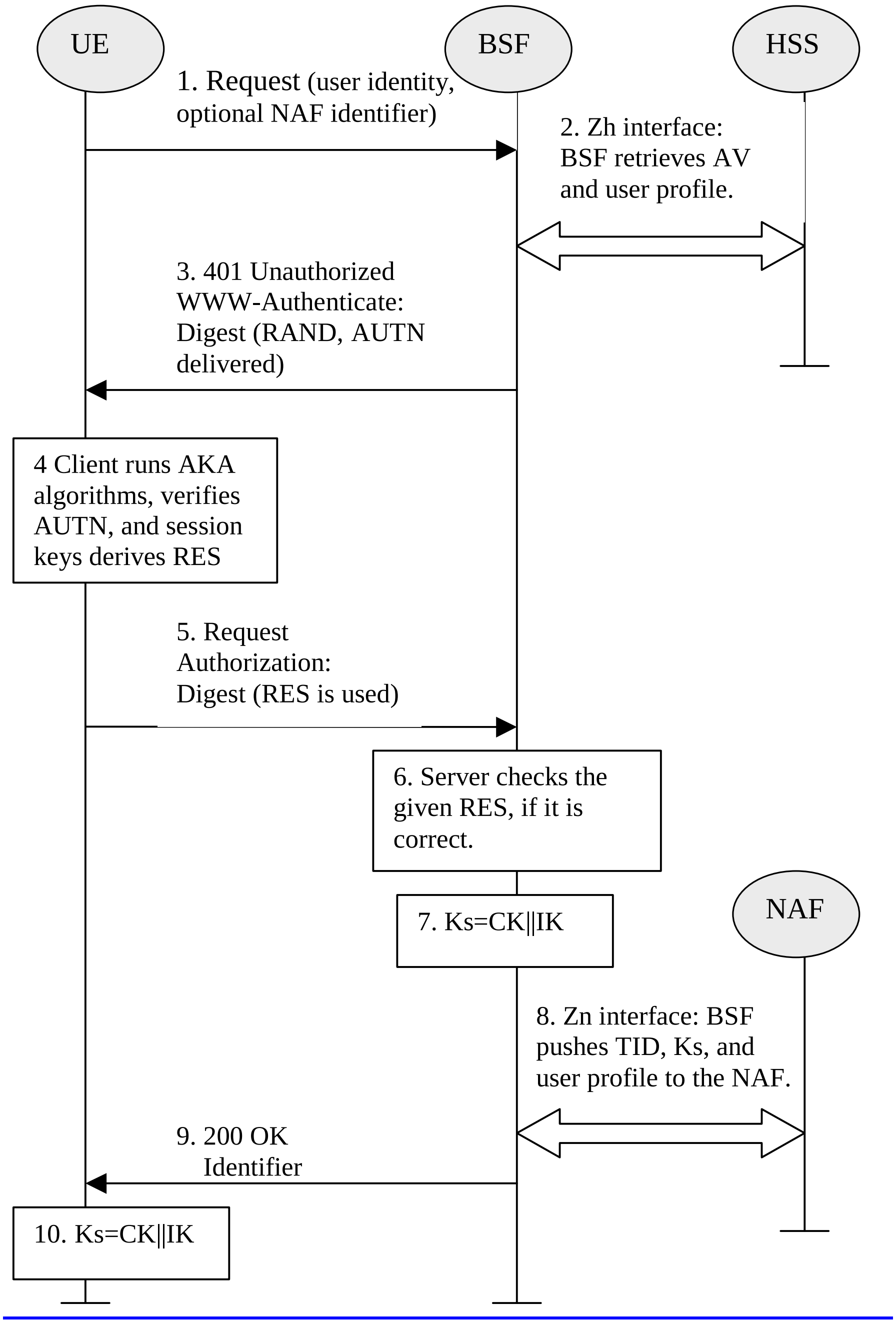

UE
BSF
HSS
1. Request (user identity, optional NAF identifier)
2. Zh interface: BSF retrieves AV and user profile.
3. 401 Unauthorized WWW-Authenticate: Digest (RAND, AUTN delivered)
4 Client runs AKA algorithms, verifies AUTN, and session keys derives RES
5. Request Authorization: Digest (RES is used)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

Figure 4: The bootstrapping procedure

1. The UE sends an HTTP request towards the BSF. The requst contains the user identity. Request may contain a NAF identifier.

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. If NAF identifiers were present in the initial bootstrapping request, BSF pushes the TID, Ks, and NAF specific user profile to NAFs identified by the NAF identifiers.

~~8~~9. 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. next 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.

~~9~~10. 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.

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, there is no need for NAF to retrieve the key(s) over Zn interface.

- If the NAF shares a key with the UE, but an update of that key 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: 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

- If the NAF already has the 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 due to the fact that the key material has been pushed to the NAF by the BSF, NAF will continue with the protocol used over Ua interface with UE.

- 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: 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.

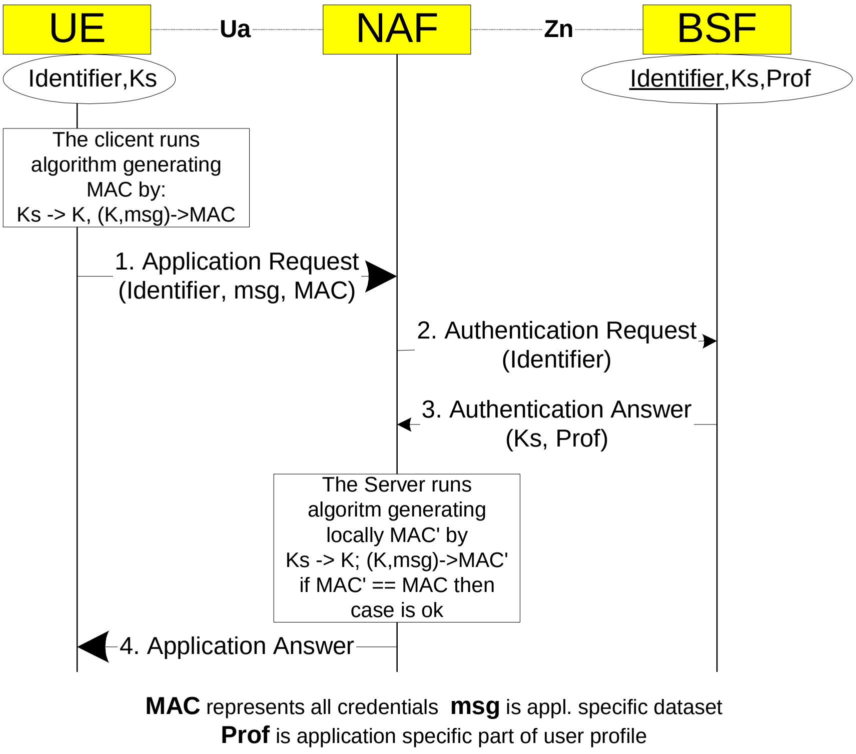

UE
Ua
NAF
Zn
BSF
Identifier,Ks
Identifier,Ks,Prof
The clicent runs algorithm generating MAC by: Ks -> K, (K,msg)->MAC
1. Application Request (Identifier, msg, MAC)
2. Authentication Request (Identifier)
3. Authentication Answer (Ks, Prof)
The Server runs algoritm generating locally MAC' by Ks -> K; (K,msg)->MAC' 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

***** END CHANGE *****