3GPP TSG SA WG3 Security — SA3#32 **S3-040077**
9 - 13 February 2004
Edinburgh, UK

Source:	**Nokia**
Title:	**Bootstrapping key lifetime and timestamp**
Document for:	**Discussion and decision**
Agenda Item:	**6.9.2**

FOREWORD

This discussion paper describes the implementation and reasoning of the key lifetime and key freshness approach to be added to current Zn interface stage-2 and stage-3 details in 3GPP TSs 33.220 and 29.109. The purpose of this discussion paper is to provide the guidance for detailed CRs to these GAA TSs.

1. INTRODUCTION

According the current state of 3GPP TS 29.109 [TS 29.109], BSF gives bootstrapping information, that consists of a transaction Identifier (TID), a NAF specific key (Ks_NAF), and an application specific profile (UserProf) to a NAF via Zn interface. On the one hand, NAF may have strict requirements on the freshness of the bootstrapping information, i.e., it may require that bootstrapping should have been done recently, e.g., not earlier than five minutes ago. On the other hand, BSF needs to limit the period of time in which bootstrapping information may be used by NAF and UE to authenticate communication over Ua interface; the bootstrapping information should not be stored in NAF forever.

These concerns can be addressed by adding a timestamp and a lifetime parameters to the bootstrapping information. BSF indicates to the NAF bootstrapping time and the validity period of the bootstrapping information with these parameters:

- timestamp (i.e., the creation time of the bootstrapping information),
- key lifetime (i.e., how long the bootstrapped key is valid)

In order to avoid the need for a separate procedure, BSF should send the key lifetime and the key freshness information (i.e., key timestamp) to NAF over Zn interface with the bootstrapping information.

The timestamp/lifetime approach imposes two new requirements on the Zn interface:

- BSF shall be able to indicate to NAF the expiration time of the bootstrapping information.
- BSF shall be able to indicate to NAF the creation time of the bootstrapping information.

The key lifetime parameter gives:

- BSF a possibility to to limit the period of time in which bootstrapping information is valid, and
- NAF a permission to clean up in a controlled way the obsolete bootstrapping information even before general storage capacity release procedures have to do it.

The key timestamp parameter gives a possibility for:

- BSF to indicate to the NAF the creation time of the bootstrapping information, and
- NAF to request UE and BSF to create new bootstrapping information if its policies require fresher keys.

The key lifetime also ensures in most cases the freshness of application specific subscriber profile, which is part bootstrapping information. (Application specific subscriber profile may change in HSS after bootstrapping and before the key lifetime expires. However, assuming that changes in subscribers' profiles are rare as compared to subscribers' use of services, this is a rare event.)

The rest of this contribution proposes how to implement the key lifetime and key timestamp parameters.

2. PROPOSALS

2.1 Key lifetime

When BSF sends the bootstrapping information to the NAF via Zn interface it includes the key lifetime information to the message.

There are basically two ways to express the lifetime:

1. As duration, from which the expiration time can be computed:
   The lifetime is expressed as duration in some feasible time units. The BSF simply sends the desired key lifetime value to the NAF. The NAF computes the expiration time by adding this duration to its system time and stores it with the bootstrapping information.

2. As expiration time:
   The lifetime is expressed as expiration time. The BSF computes the expiration time by adding the desired key lifetime to its system time and sends the result to the NAF. The NAF stores this expiration time with each bootstrapping information for comparation with its system time later.

The second alternative, i.e., expiration time is recommended.

The value of key lifetime is set in BSF configuration by home operator. Notice, that it may be different for different kinds of NAFs. Because the home operator configures also BSFs it is not necessary to include the key lifetime information to HSSs to be transferred to BSFs via Zh interfaces.

2.2 Key timestamp

When BSF creates the bootstrapping information, it will create the key timestamp intormation. The timestamp value could be, e.g., seconds since January 1st, 1970 (i.e., Unix style), or seconds since 0 AD (i.e., Symbian style).

The key timestamp can be communicated to the NAF in a number of ways:

1. as a parameter in Zn interface:
   The bootstrapping timestamp would be sent to the NAF with other bootstrapping information over Zn interface.

2. as part of the TID:
   The bootstrapping timestamp would be intergrated as part of the TID, e.g., first 40 bits of TID would indicate the timestamp.

The second option, i.e., key timestamp is sent as part of the TID is recommended.

The TID approach gives a possibility for NAF to request for a new bootstrapping session between UE and BSF even before it has contacted the BSF since the timestamp is part of the TID and which NAF receives over Ua interface. The TID approach also gives a possibility for the UE to track the bootstrapping creation times that are recorded by BSF. Of course, UE may also record the creation time itself since it participates in bootstrapping.

3. CONCLUSION

SA3 is asked to endorse the following:

1. *BSF shall be able to indicate to NAF the expiration time of the bootstrapping information.* This should be added as a new requirement into [TS 33.220] for Zn interface.

2. *BSF shall be able to indicate to NAF the creation time of the bootstrapping information.* This should be added as a new requirement into [TS 33.220] for Zn interface.

3. *BSF shall send the key lifetime value to NAF with other bootstrapping information over Zn interface.* This should be incorporated into[TS 29.109].

4. *BSF shall encode the key timestamp value into the TID value.* The method of creating the TID should be incorporated into [TS 33.220].

4. REFERENCES

[TS 29.109] 3GPP TS 29.109 “Generic Authentication Architecture (GAA); Zh and Zn Interfaces based on the Diameter protocol; Protocol details (Release 6)”.

[TS 33.220] 3GPP TS 33.220, “Generic Authentication Architecture (GAA); Generic Bootstrapping Architecture”.

***** 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 transaction identifier shall contain a time stamp indicating the creation time of bootstrapping information.

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 indicate to the NAF the lifetime of the bootstrapping information
- The BSF shall be able to indicate to the NAF the creation time of the bootstrapping information.

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.

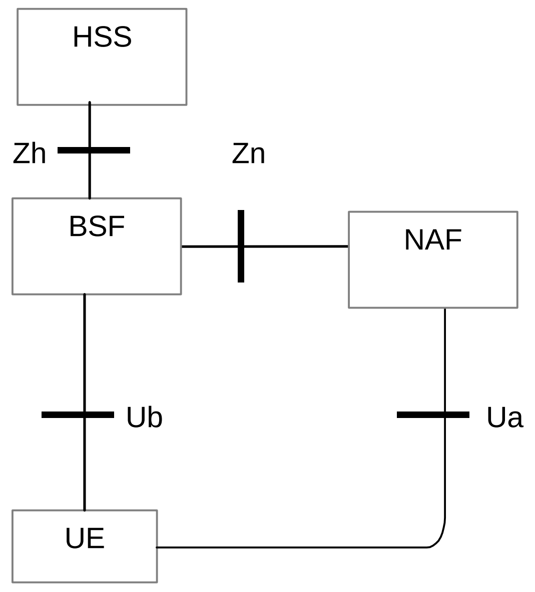


Figure 1: Simple network model for bootstrapping

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.

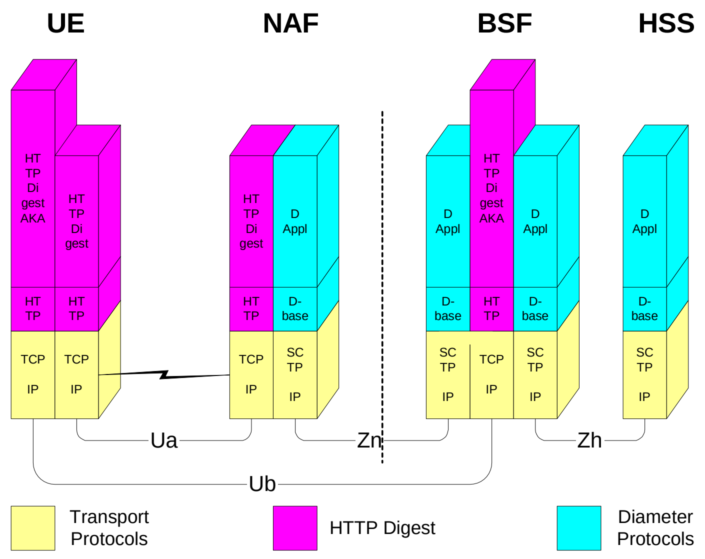


Figure 2: Protocol stack architecture

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.

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 note: 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. 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).

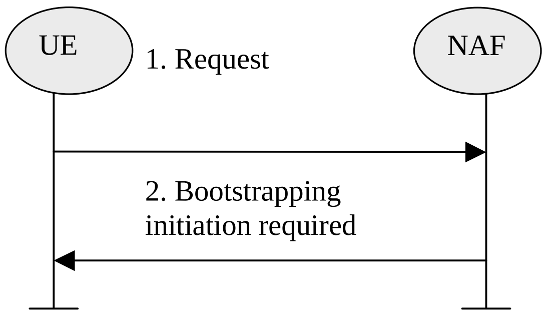


Figure 3: Initiation of bootstrapping

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 note: 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 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 (cf. subclause 4.3.3).

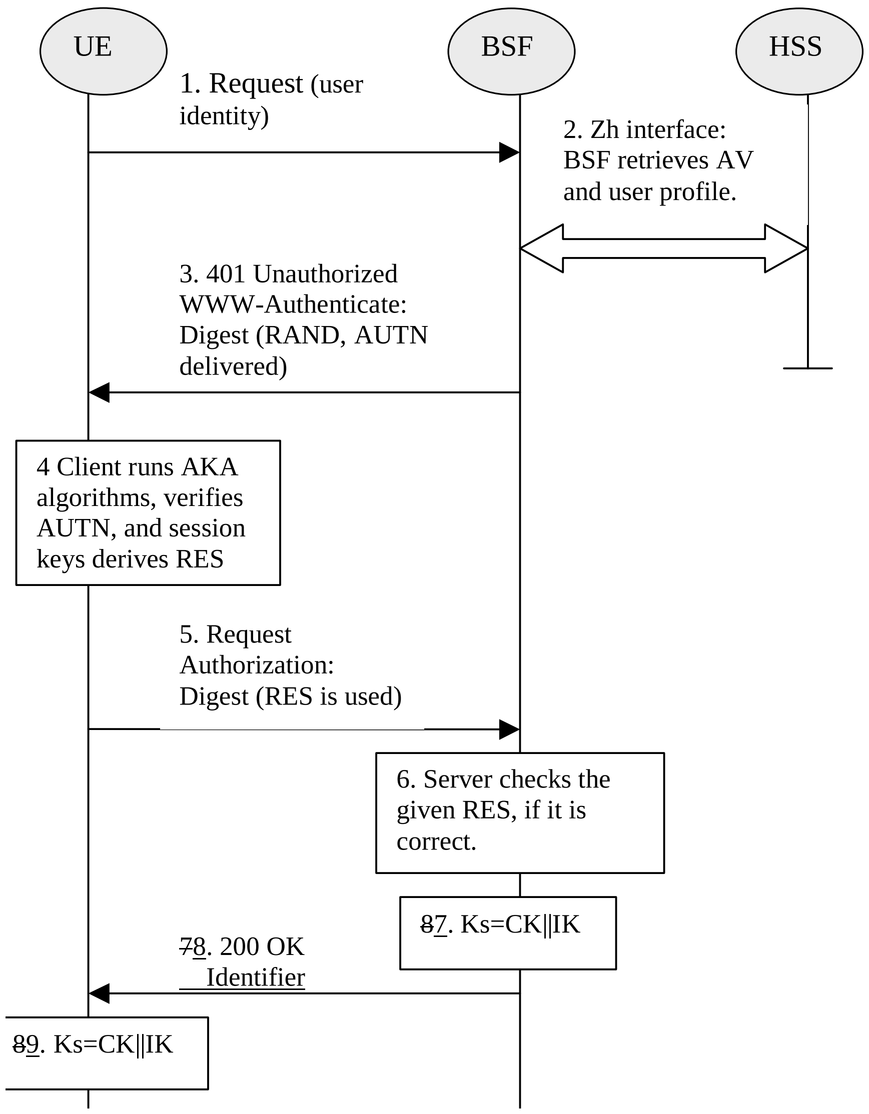

UE
BSF
HSS
1. Request (user identity)
2. Zh interface: BSF retrieves AV and user profile.
3. 401 Unauthorized WWW-Authenticate: Digest (RAND, AUTN delivered)
4 Client runs AKA algorithms, verifies AUTN, and session keys derives RES
5. Request Authorization: Digest (RES is used)
6. Server checks the given 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.

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.

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 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.

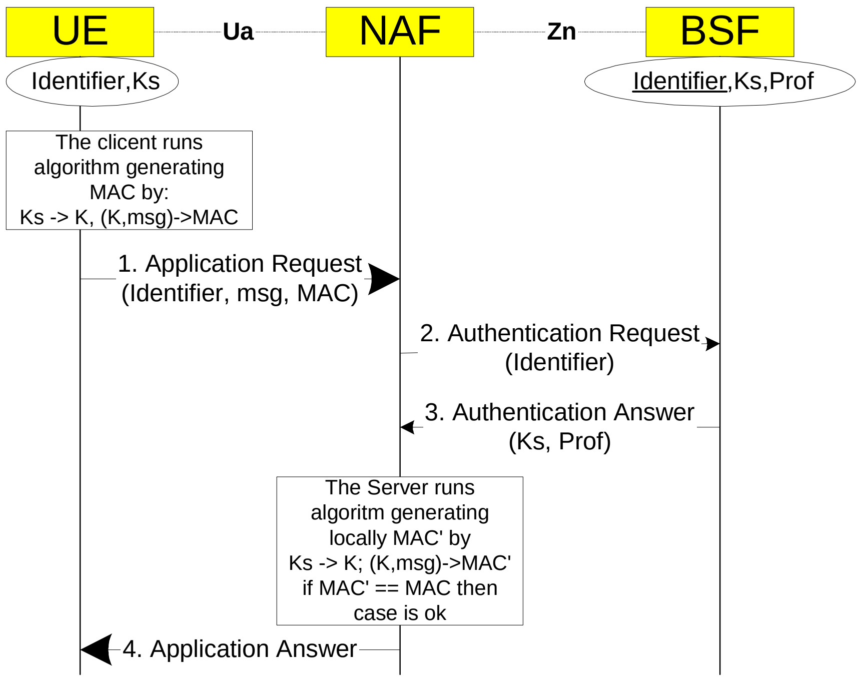
- The BSF derives the keys required to protect the protocol used over Ua interface from the key material and the key derivation parameters, as specified in clause 4.3.2, and supplies to NAF the requested key material. If the key identified by the transaction identifier supplied by the NAF is not available 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

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