3GPP TSG SA WG3 Security — S3#32 **S3-040162**
9 –13 February 2004
Edinburgh, UK

Title: LS on key derivation for the Generic Bootstrapping Architecture
Release: 6
Work Items: Support for Subscriber Certificates and Generic Authentication Architecture

Source: 3GPP SA3
To: ETSI SAGE
Cc:

Contact Person:
Name: Guenther Horn
Tel. Number: +498963641494
E-mail Address: guenther.horn@siemens.com

Attachments: 3G TS 33.220 v100, S3-030552, S3-040161

Abstract

3GPP SA3 currently works on a Generic Bootstrapping Architecture. The specification is contained in TS 33.220. The current version TS 33.220 v1.0.0 (which is attached to this LS) specifies the use of a key derivation function KDF in section 4.3.2. 3GPP SA3 kindly asks ETSI SAGE to assist in completing the specification of TS 33.220 by providing a specification for a key derivation function satisfying the requirements outlined in this LS. SA3 would also like to mention that TS 33.220 is not completely stable yet, and that additional requirements on the key derivation function may arise during the next SA3 meeting in May 2004.

Statement of the problem

The entities involved: a User Equipment (UE), composed of a Mobile Equipment (ME) and a Universal Integrated Circuit Card (UICC), communicates with a Bootstrapping Server Function (BSF) in the home network. The UE also communicates with an application server, called Network Application Function (NAF). There may be several application servers with which a UE communicates. The BSF acts as a key distribution server. It has interfaces with the NAFs and an interface with the Home Subscriber System (HSS) which contains the 3G Authentication Centre supplying authentication vectors. The situation is depicted in the figure below.

The protocol between UE and BSF over the Ub interface uses a 3G authentication vector (RAND, AUTN, CK, IK, RES) as input and results in a key Ks = CK | IK shared between UE and BSF. Ks is not tied to the use with a particular NAF. This may result in a security threat described in Tdoc S3-030552 (attached), if Ks is used as a key shared between UE and NAF. In order to mitigate this threat and obtain a key specific to one NAF, a key derivation function is applied to Ks, using further key derivation parameters as input.

From the text of TS 33.220 v100, section 4.3.2:
" 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 and RAND. "

NAF_Id is the fully qualified domain name of the NAF. It serves as an identity of the application server NAF.

Developments after the completion of TS 33.220 v100:
In TS 33.220 v100, only one key Ks_NAF is derived from one Ks. If a Ks_NAF for a different NAF is required then first a new Ks is to be established. This requirement has been relaxed, and it is now allowed to derivekeys Ks_NAF for different NAFs from one Ks, cf. S3-040043 in the attachment.

- In TS 33.220 v100, key derivation is performed on the ME, not on the UICC. At 3GPP SA3's meeting #32, it was proposed to specify a new variant of a Generic Bootstrapping Architecture, where key derivation is performed on the UICC. This may lead to additional requirements on the key derivation

function.
(A side note: 3GPP SA3 does not intend to mandate having the key derivation on the UICC as this would require the issuing of new UICCs to all subscribers who want to participate in the Generic Bootstrapping Architecture.)

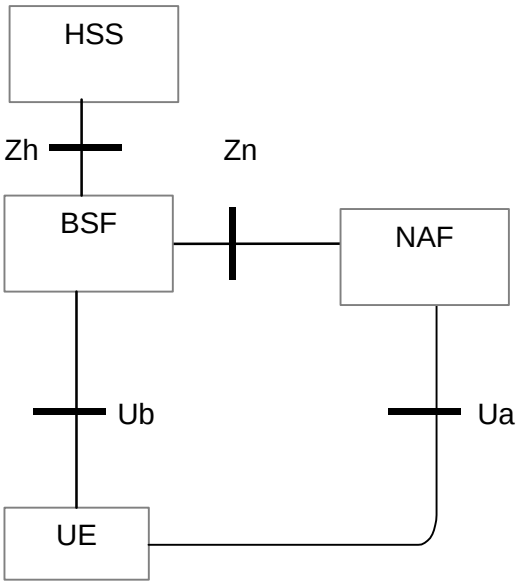

Figure 1: Simple network model for bootstrapping

Requirements on the key derivation function KDF

The following requirements have been identified so far:

1) *It shall not be possible to gain any useful knowledge about Ks from the derived keys Ks_NAF.*

2) *It shall not be possible to gain any useful knowledge about Ks1_NAFi from Ks2_NAFj for different i,j and any Ks1 and Ks2.*

3) Random input to key derivation: typically, key derivation needs random input in addition to the initial key Ks. It is suggested to use the RAND from the authentication vector, which was used to produce Ks.

4) *Identity of user and application server as input to key derivation:*
It is proposed to use IMSI and NAF_Id as input.

5) A fixed length of the key Ks_NAF of 256 bits is considered sufficient. This is not a hard requirement at this time, but rather results from the requirement to make the key derivation function as widely applicable as possible.

6) It would be desirable to design the key derivation function in a way which allows it to be easily adopted for future 3GPP applications with different input parameters.

ACTION:

3GPP SA3 kindly asks ETSI SAGE to assist in completing the specification of TS 33.220 by providing a specification for a key derivation function satisfying the requirements outlined in this LS. If SAGE agrees to take the work on 3GPP SA3 would also appreciate if SAGE could indicate a time-frame for the completion of the work by sending a reply to SA3's meeting in May. The work should be completed within the deadline for 3GPP Release 6. Unfortunately, the precise deadline is not known yet, but it is expected that the SA plenary will decide it in March. It will then be communicated to SAGE. Any further observations by ETSI SAGE would, of course, also be welcome. If more information is required SAGE is kindly asked to contact SA3.

Date of Next SA3 Meetings:

SA3#33	10 - 14 May 2004	Beijing
SA3#34	5 – 9 July 2004	North America
SA3#35	4 – 8 October 2004	tbd

3GPP TSG SA WG3 Security — S3#32 **S3-040189**
09 - 13 February 2004, Edinburgh, Scotland, UK

3GPP TS 33.220 V1.1.0 (2004-02)

Technical Specification

3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Generic Authentication Architecture (GAA); Generic Bootstrapping Architecture (Release 6)

Keywords
Security, GAA

3GPP

Postal address

3GPP support office address
650 Route des Lucioles - Sophia Antipolis
Valbonne - FRANCE
Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Internet
http://www.3gpp.org

Contents

Foreword

This Technical Specification has been produced by the 3rd Generation Partnership Project (3GPP).

The contents of the present document are subject to continuing work within the TSG and may change following formal TSG approval. Should the TSG modify the contents of the present document, it will be re-released by the TSG with an identifying change of release date and an increase in version number as follows:

Version x.y.z

where:

- x the first digit:
  - 1 presented to TSG for information;
  - 2 presented to TSG for approval;
  - 3 or greater indicates TSG approved document under change control.
- y the second digit is incremented for all changes of substance, i.e. technical enhancements, corrections, updates, etc.
- z the third digit is incremented when editorial only changes have been incorporated in the document.

1 Scope

The present document describes the security features and a mechanism to bootstrap authentication and key agreement for application security from the 3GPP AKA mechanism. Candidate applications to use this bootstrapping mechanism include but are not restricted to subscriber certificate distribution [5], etc. Subscriber certificates support services whose provision mobile operator assists, as well as services that mobile operator provides.

The scope of this specification includes a generic AKA bootstrapping function, an architecture overview and the detailed procedure how to bootstrap the credential.

Editor's note: The specification objects are scheduled currently in phases. 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.

2 References

The following documents contain provisions which, through reference in this text, constitute provisions of the present document.

- References are either specific (identified by date of publication, edition number, version number, etc.) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies. In the case of a reference to a 3GPP document (including a GSM document), a non-specific reference implicitly refers to the latest version of that document *in the same Release as the present document*.

[1] 3GPP TS 31.102: "3rd Generation Partnership Project; Technical Specification Group Terminals; Characteristics of the USIM application".

[2] 3GPP TS 33.102: "3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; 3G Security; Security architecture".

[3] Franks J., et al,: "HTTP Authentication: Basic and Digest Access Authentication", RFC 2617, June 1999.

[4] A. Niemi, et al,: "Hypertext Transfer Protocol (HTTP) Digest Authentication Using Authentication and Key Agreement (AKA)", RFC3310, September 2002.

[5] 3GPP TS 33.221: "3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Generic Authentication Architecture (GAA); Support for Subscriber Certificates".

[6] T. Dierks, et al,: "The TLS Protocol Version 1.0", RFC 2246, January 1999.

[7] OMA: “Provisioning Content Version 1.1”, Version 13-Aug-2003. Open Mobile Alliance.

[8] 3GPP TS 23.228: “3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; IP Multimedia Subsystem (IMS); Stage 2 (Release 6) ”.

3 Abbreviations

For the purposes of the present document, the following abbreviations apply:

AK	Anonymity Key
AKA	Authentication and Key Agreement
BSF	Bootstrapping server functionality BSF is hosted in a network element under the control of an MNO.
BSP	BootStrapping Procedure
CA	Certificate Authority
CMP	Certificate Management Protocols
GAA	Generic Authentication Architecture
GBA	Generic Bootstrapping Architecture
HSS	Home Subscriber System
IK	Integrity Key
MNO	Mobile network operator
NAF	Operator-controlled network application function functionality. NAF is hosted in a network element under the control of an MNO.
PKCS	Public-Key Cryptography Standards
PKI	Public Key Infrastructure
SCP	Subscriber Certificate Procedure
UE	User Equipment

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.

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.

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.

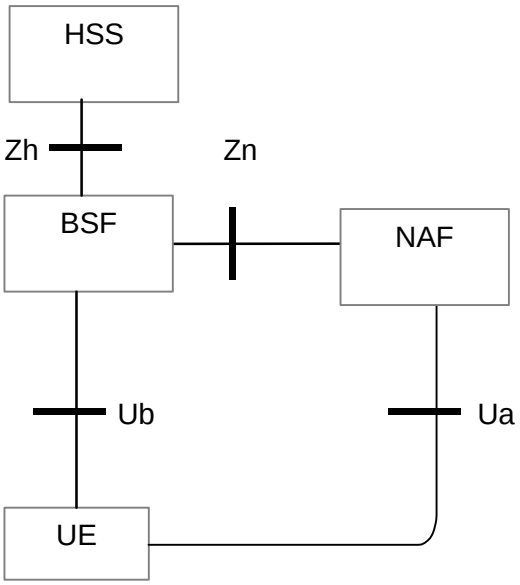

Figure 1: Simple network model for bootstrapping

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.

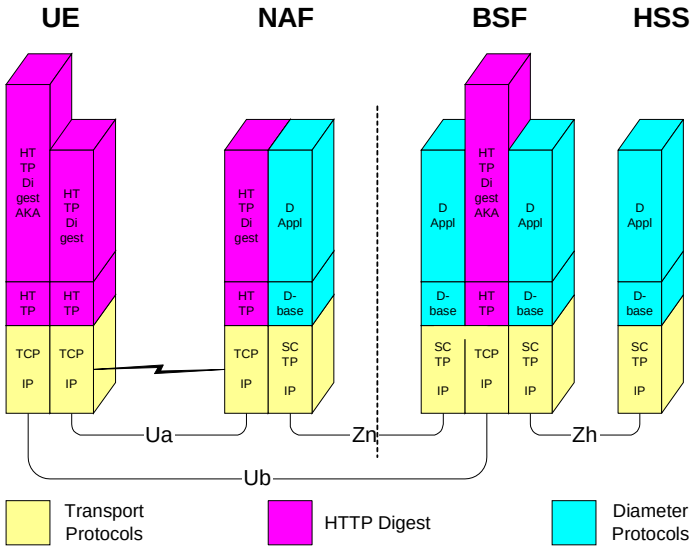

Figure 2: Protocol stack architecture

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).

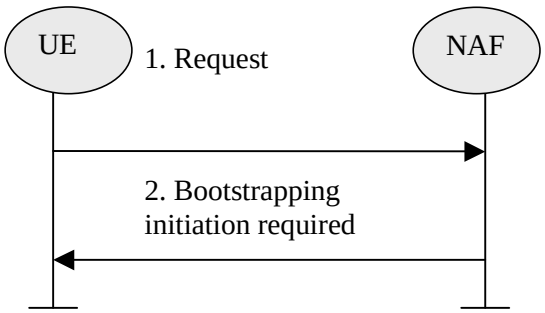

Figure 3: Initiation of bootstrapping

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).

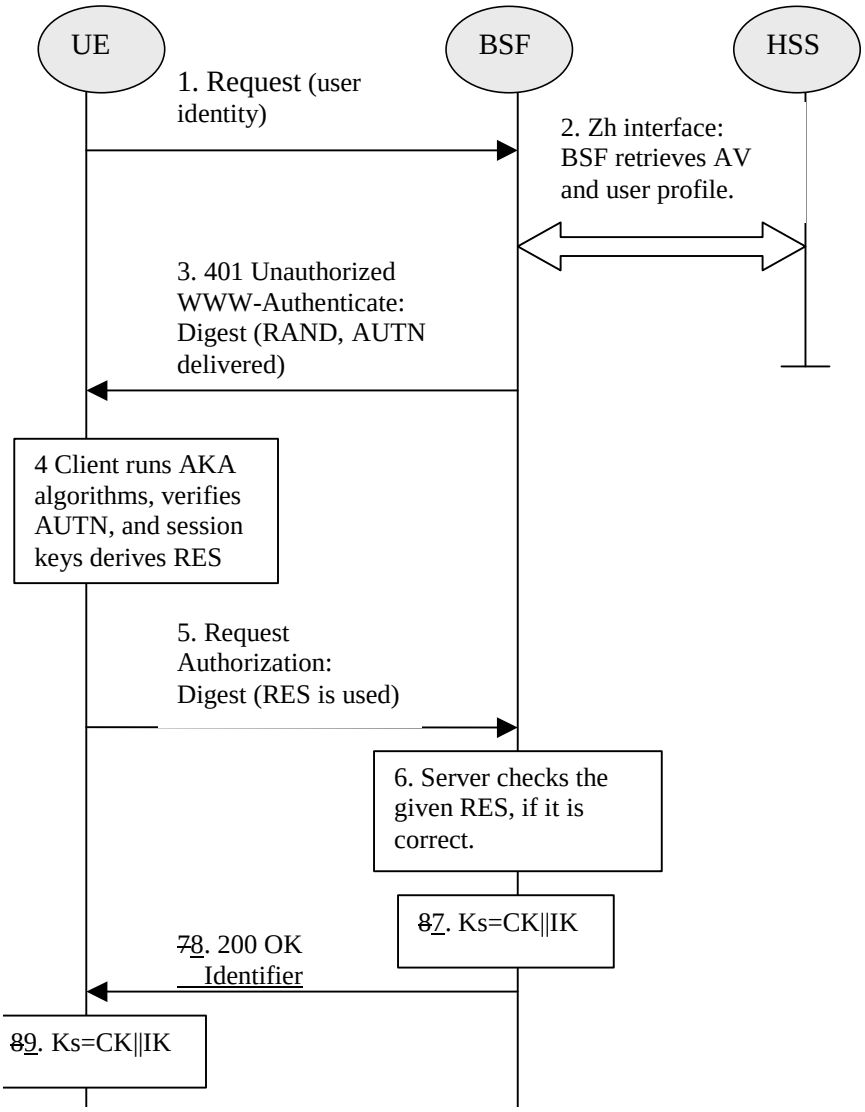

Figure 4: The bootstrapping procedure

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.

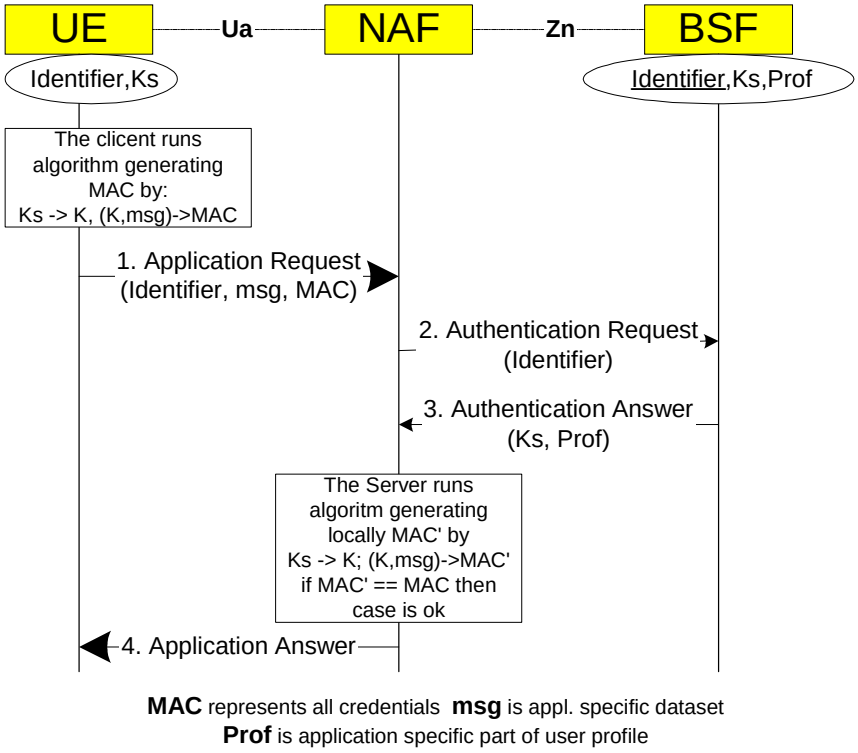

Figure 5: The bootstrapping usage procedure

4.3.4 Procedure related to service discovery

To enable the bootstrapping procedure, a procedure needs to be described on how to discover the location of BSF. It shall be possible to enable the UE to be configured either manually or automatically via one of the following approaches:

- The address information shall be published via reliable channel. Subscribers shall store all the parameters as part of the initial establishment of IP connectivity. The addresses need to be input only once.

- The address information shall be pushed automatically to the UE over the air interface when the subscription to bootstrapping service is accepted. All the parameters shall be saved in the UE and used the same manner as above. The procedure is specified in [7].

- The location information shall be discovered automatically based on DHCP, after the IP connectivity has been established. The DHCP server shall provide the UE with the domain name of a BSF and the address of a Domain Name Server (DNS) that is capable of resolving the Fully Qualified Domain Name (FQDN) of the BSF. The procedure is specified in [8].

Note: The location of DHCP server may be pushed to UE through the procedure specified in [7].

Note: The case when BSF located in the visited network is ffs in later phase of this work item.

Annex A (informative): Generic secure message exchange using HTTP Digest Authentication

A.1 Introduction

Editor's note: This annex describes how HTTP Digest Authentication can be used between UE and any NAF where the protocol over Ua interface is based on HTTP messaging. The protocol over Ua interface may depend upon the final choice of scheme made by SA WG3 and this will need to be reviewed later by SA WG3.

HTTP Digest Authentication model can also be used as a generic authentication and integrity protection method towards any new NAF. If a new NAF uses BSF-based security association, it could use this generic method to authenticate the UE (and UE authenticate the NAF) and integrity protect any payload being transferred between NAF and UE. As a generic method, it will speed up the specification of new NAFs since the authentication and message integrity protection part of Ua interface are taken care of by HTTP Digest Authentication. It will also ease the implementation of BSF-based authentication in NAFs because there would be one well-defined way to do it.

A.2 Generic protocol over Ua interface description

The sequence diagram in figure A.1 describes the generic secure message exchange with HTTP Digest Authentication. The conversation may take place inside a server-authenticated TLS [6] tunnel in which case TLS handshake has taken place before step 1.

In step 1, UE sends an empty HTTP request to a NAF. In step 2, NAF responds with HTTP response code 401 "Unauthorized" which contains a WWW-Authenticate header. he header instructs the UE to use HTTP Digest Authentication with a bootstrapped security association. Quality of protection (qop) attribute is set to "auth-int" meaning that the payload of the following HTTP requests and responses should integrity protected. The realm attribute contains two parts. The first part is a constant string "3GPP-bootstrapping" instructing the UE to use a bootstrapped security association. The second part is the DNS name of the NAF.

In step 3, the UE shall verify that the second part of the realm attribute does in fact correspond to the server it is talking to. In particular, if the conversation is taking place inside a server-authenticated TLS tunnel, the UE shall verify that the server name in the server's TLS certificate matches the server name in the realm attribute of the WWW-Authenticate header. The UE generates client-payload containing the message it wants to send to the server. Then it will generate the HTTP request by calculating the Authorization header values using the transaction identifier (base64 encoded) it received from the BSF as username and the session key Ks (base64 encoded) as the password, and send the request to NAF in step 4.

When NAF receives the request in step 5, it will verify the Authorization header by fetching the session key Ks from the bootstrapping server using Zn interface and the transaction identifier. After successful retrieval, NAF calculates the corresponding digest values using K, and compares the calculated values with the received values in the Authorization header. The NAF shall also verify that the DNS name in the realm attribute matches its own. If the conversation is taking place inside a server-authenticated TLS tunnel, the NAF shall also verify that this DNS name is the same as that of the TLS server. If the verification succeeds, the incoming client-payload request is taken in for further processing. Thereafter, the NAF will generate a HTTP response containing the server-payload it wants to send back to the client in step 6. The NAF may use session key Ks to integrity protect and authenticate the response.

In step 7, UE receives the response and verifies the Authentication-Info header. If the verification succeeds, the UE can accept the server-payload for further processing.

Additional messages can be exchanged using steps 3 through 7 as many times as is necessary. The following HTTP request and responses must be constructed according to [3] (e.g., nc parameter must be incremented by one with each new HTTP request made by UE).

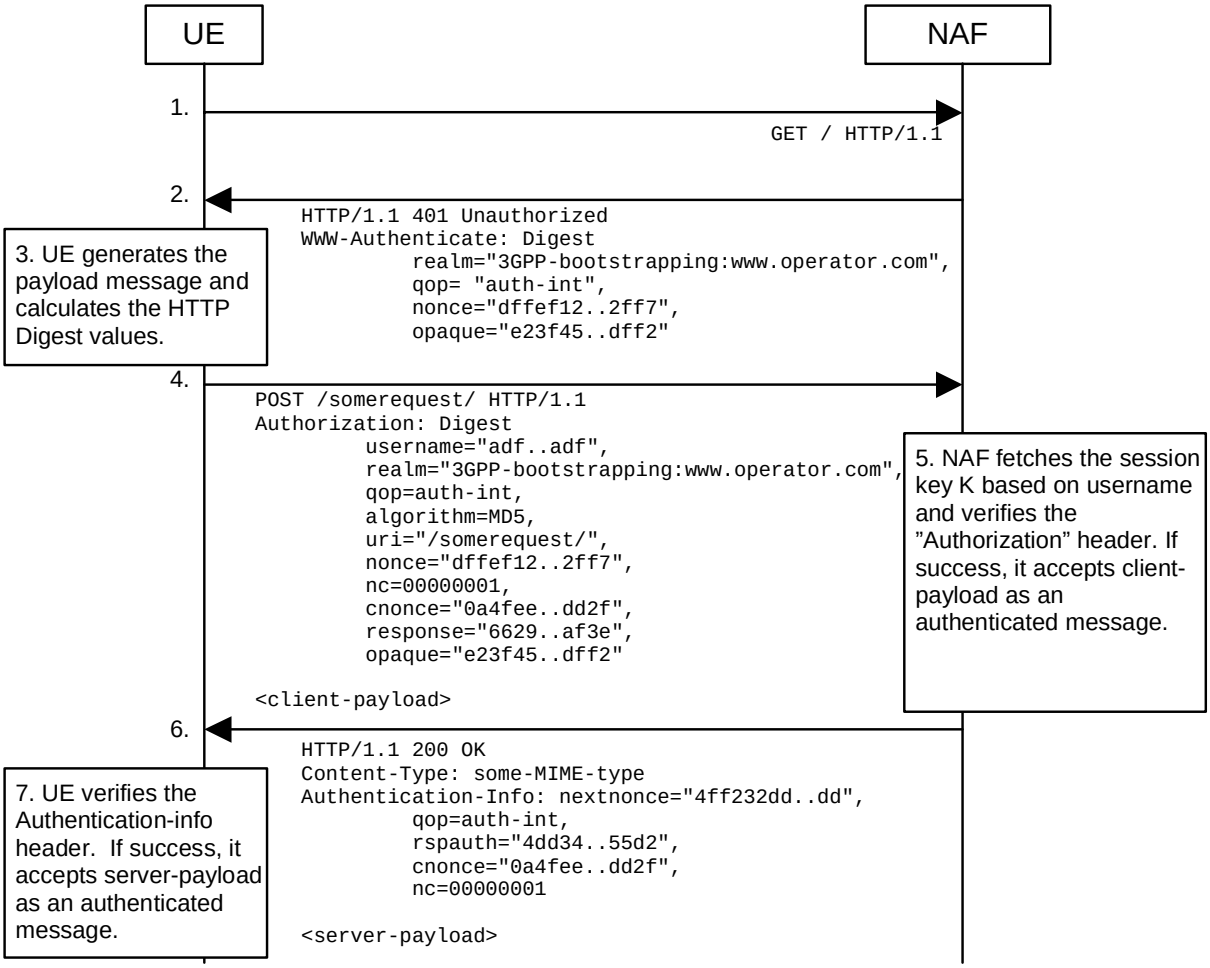

Figure A.1: Generic secure message exchange using HTTP Digest Authentication and bootstrapped security association

Annex B (informative): Change history

Change history							
Date	**TSG #**	**TSG Doc.**	**CR**	**Rev**	**Subject/Comment**	**Old**	**New**
2003-10	SA3#30				New draft TS: Generic Bootstrapping Architecture (GBA). Extracted from 33.109 clause 4 and Annex B.		0.1.0
2003-10	SA3#30	S3-030537			New interface names.		0.1.0
2003-10	SA3#30	S3-030538			Requirements for Ub and Zh interfaces added.	0.1.0	0.1.1
2003-10	SA3#30	S3-030545			NAF initiated bootstrapping added	0.1.0	0.1.1
2003-10					Imported Zn interface requirements from SSC TS.	0.1.0	0.1.1
2003-11	SA3#31	S3-030728			Bootstrapping procedure: merging of last two messages	0.1.1	0.2.0
2003-11	SA3#31	S3-030793			Key separation	0.1.1	0.2.0
2003-11	SA3#31	S3-030794			Removal of application specific user profile requirements from GBA	0.1.1	0.2.0
2003-11					Annex A: changed "session key K" to "session key Ks"	0.1.1	0.2.0
2003-12	SP-22	SP-030583	-	-	Presentation to TSG SA#22 for Information	0.2.0	1.0.0

3GPP TSG SA WG3 Security#30 **S3-030552**
6 – 10 October, 2003
Povoa do Varzim, Portugal

Agenda Item: **7.9 (GAA)**

Source: **Siemens**

Title: **Key separation in a Generic Bootstrapping Architecture**

Document for: **Discussion and decision**

Abstract

This contribution proposes to enhance the current procedure to distribute shared keys to a UE and a NAF defined in TS 33.109 by allowing the derivation of shared keys, which can only be used with a specific NAF or a group of NAFs.

1. Problem statement

Establishment of key shared between UE and NAF: The draft TS 33.109 "Bootstrapping of application security using AKA and support for Subscriber Certificates" (latest version in S3-030488) describes in section 4.3.1 how a UE and an application server (NAF) can obtain a shared key in a generic way from a run of a protocol A (http digest aka) between the UE and a Bootstrapping Server Function (BSF). The procedures using bootstrapped security associations are described in section 4.3.2 of TS 33.109. According to section 4.3.2, a user first has to run protocol A with the BSF in case he does not yet share a key with the NAF he is about to access. In the final message of protocol A, the BSF sends the UE a transaction identifier. The UE sends the transaction identifier to the NAF in protocol B. The NAF then retrieves the key sending the transaction identifier to the BSF in protocol D. This transaction identifier carries no information about the NAF. The BSF will give the key to any NAF which presents the transaction id and which authenticates to the BSF as a legitimate NAF. The BSF has no clue for which NAF the key is intended by the UE.

This has the following consequence:

Lack of key separation: the UE does not know with which NAF, or type of NAF (e.g. presence, PKI portal), it shares a particular key. But this could be important for the UE to know if the key shared with the NAF is used for authentication of the NAF to the UE, and the UE wants to perform certain actions only with a certain (type of) NAF. The bootstrapping architecture is just meant to provide shared keys to entities (UE and NAF) who initially do not share keys in a generic way. The architecture should not restrict the uses of the shared key. So it should be allowed that the shared key is used for NAF to UE authentication. A Generic Authentication Architecture should not be limited to server authentication by certificates.

An example where the bootstrapped key is used for mutual authentication between UE and NAF would be its use with shared_key_TLS, which was recently discussed on the SA3 mailing list.

The rest of the contribution is organised as follows: section 2 goes into some more detail regarding the threats when there is no key separation. Section 3 suggests an improvement of the currently specified procedure. The improvement consists in performing a key derivation, including NAF-specific parameters, before a key is sent from the BSF to a NAF. Section 4 discusses possible key derivation parameters. Section 5 proposes a flexible mechanism to signal information about the key derivation scheme.

2. Threats when key separation is lacking

A NAF may be just any type of application server. It may be e.g. a presence server, or a PKI portal, or a conference server, or a game server. It may reside in the operator's home network, or in a visited network. The applications provided by the NAF may be defined outside 3GPP, e.g. by OMA or other standards organisations, or may even be

proprietary. Application servers may be provided by a large variety of different vendors. Some applications may have to be introduced quickly by operators to satisfy market needs. As a consequence, it can be expected that the security levels of NAF implementations, and the security of the operating environment of NAFs, will vary considerably.

The main threats to the current bootstrapping architecture in TS 33.109 arise when the shared key is used for NAF to UE authentication., e.g. in shared_key_TLS. Due to the possibly different security levels of NAFs, it may happen that one of the many NAFs in one network (not necessarily the home network) is successfully attacked, either by insiders, e.g. administrators, or remotely by hackers. An attacker controlling a NAF could then retrieve just any key from the BSF whose transaction identifier is known to him. A rogue NAF could in this way impersonate any other NAF known to a particular BSF as legitimate, without a possibility for the user to know.

In this way, a security breach in one type of application (e.g. through a software glitch) may then spread to other types of applications, and a security breach in one network may be exploited to perform attacks in other networks.

But, in general, it is important for the user to know to which NAF he is talking. E.g., a user may want to entrust certain personal data to a PKI portal of his home network, but perhaps not to a commerce server in a visited network.

It would therefore be prudent to design a Generic Bootstrapping Architecture in such a way that a spreading of security failures across the entire system can be prevented by subdividing the system in compartments which are mutually separated from each other from a key distribution point of view.

Examples of attacks (other types of attacks may be possible):

Variant1: a user may want to talk to an application server with the name PKI_portal.com. Now assume that one application server NAF1 was successfully attacked, and that the attacker has control over interface D from NAF1 to the BSF. The attacker now sets up a rogue server with the name PKI-portal.com, and may be able to entice a user to connect to it. As social engineering goes, the user will not notice, or not care about, the minimal difference in the names of the two PKI portals. Note that the attacker needs no control over the routing path between the UE and the rogue server, as the attacker could legitimately obtain an entry in the DNS for the modified server name, and the DNS could return an entirely different IP address for the modified server name than for the original name. The attacker can eavesdrop on the messages of protocol A between UE and BSF, or the attacker can simply wait for the first message from the UE in protocol B, and in this way the attacker can learn the transaction identifier. The attacker then triggers NAF1 to retrieve the shared key corresponding to the transaction identifier from the BSF, and provides the rogue "PKI-portal.com" server with that key. The UE and the rogue server will then successfully perform mutual authentication.

Variant2: here, it is assumed that the attacker can play man-in-the-middle on the first link from the UE, and, in this way, has control over the routing path from the UE to the server. This would be the case e.g. when the attacker used a false base station to which the UE attached over the radio link. Then the attacker could even use the true server name PKI_portal.com towards the user. The rest of the attack works as for variant 1.

The attack can be prevented for both variants if

- interface D is assumed to provide mutual authentication, confidentiality and integrity; and
- the BSF checks the NAF identity against a list of authorised NAFs; and
- the key shared between UE and NAF is specific to one NAF or a defined set of NAFs.

The attack is prevented because, in both variants, the attacked server NAF1 would be able retrieve only keys from the BSF, which are specific to NAF1or the set to which NAF1 belongs. The keys could be used with neither PKI-portal.com nor PKI_portal.com.

3. Key derivation

In order to mitigate the threats identified in section 2, the following key derivation procedure is proposed:

In the current text of TS 33.109, the key shared between a UE and a NAF is Ks, the key resulting from a run of protocol A between UE and BSF. It is proposed in this contribution that a NAF-specific key Ks_NAF is used between UE and a NAF instead. This NAF-specific key is derived from Ks using suitable input parameters.

Ks_NAF may be specific to a particular NAF or a set of NAFs. This is ffs and is determined by the type of key derivation parameters. Alternatives are discussed in section 4. The derived key is computed as

Ks_NAF = H(Ks, Key_deriv_par) where Key_deriv_par is the key derivation parameter which characterises the NAF, or set of NAFs, and H is a suitable key derivation function.

The procedure would then be as follows:

- the UE runs protocol A with the BSF, as in the current text of TS 33.109. During the protocol run, key Ks is established, and the UE receives a transaction identifier;
- the UE invokes protocol B with the NAF, sending the transaction identifier received in protocol A.
- The NAF invokes protocol D with the BSF to retrieve the key shared with the UE, sending the transaction identifier received in protocol B. Upon receiving the request from the NAF, the BSF derives the key Ks_NAF from Ks and the key derivation parameter, and sends it to the NAF.
- The UE derives the key Ks_NAF from Ks and the key derivation parameter, and uses it in protocol B.

(Only the marked text deviates from the current TS 33.109.)

It could also be envisaged that, in future releases of the USIM, the key Ks remains in the UICC, and the key derivation is performed on the UICC. This is, however, not part of this contribution, and should not be mandated even in future releases.

4. Key derivation parameters

The key derivation parameters Key_deriv_par are expected to include:

- the user identity or the transaction identifier (The user identity may be part of the transaction identifier.);
- a NAF identifier (see below);
- possibly additional random information (e.g. the RAND from the run of protocol A).

For a discussion of suitable key derivation parameters in a slightly different context see also the LS from SAGE in S3-030219.

The rest of this section deals with the question which type of NAF identifier should be included. This type of NAF identifier determines the degree of assurance the user gets about the identity of the NAF-server it talks to through the mutual authentication using Ks_NAF.

We consider several alternatives for the NAF identifier to be included in the key derivation parameter:

1. *DNS server name*: this would be the finest granularity. The UE would use the DNS name used in requests in protocol B (e.g http requests). The BSF would use the FQDN from the DIAMETER request in protocol D.
   In the case of shared_key_TLS, this solution would imply that, in the case of name-based virtual hosts on the same machine (e.g. a reverse proxy), several different TLS connections would have to be used between the UE and the NAF (e.g. a reverse proxy), one for each server name, because the session keys would be different for each virtual host. But this apparent shortcoming may not be specific to the use of the shared key for NAF-to-UE authentication, because a companion contribution by Siemens to this meeting shows that the same problem exists, when server certificates are used for NAF-to-UE authentication. So this drawback may be difficult to avoid even in a more general setting.

But it is true that there may be situations where the user does not care about which server in particular he is talking to, as long as he knows that the server is authorised, e.g. the user's home network provider, to provide a particular service. So, a coarser granularity of server authentication may suffice. This leads to the points 2 and 3 below.

2. *Higher-level domains in the DNS server name*: only the n (e.g. two or three) highest domains in the server name would be used as NAF identifier, not the full DNS name. E.g in a NAF server name "server1.presence.bootstrap.operator.com", the server-side key derivation parameter would be "operator.com " or "bootstrap.operator.com ". The UE and the BSF would have to be explicitly told which scheme to use, and it would probably be difficult to make updates or changes, unless there is flexibility built into the system from the start. A flexible solution is outlined in section 5.
   With this type of NAF identifier, in the case of shared_key_TLS, several name-based virtual hosts would be able to use the same key, so, in principle, only one TLS connection between a UE and a reverse proxy could suffice. But

this advantage may be theoretical only, as it is unclear how a UE would know which servers would be served by the same reverse proxy.

3. *Types of NAFs or application names*: DNS server names of NAFs would be mapped to pre-defined types or application names, which would be used as NAF identifiers. The UE and the BSF would have to be told which DNS server names are grouped with which type or application name. 3GPP would have to standardise a format for, and a complete list of, types or application names. The configuration effort for updates in the UE would be probably be considerable, if not prohibitive.
   Regarding shared_key_TLS, the same remark as for item 2 above applies.

4. *NAF IP address*: several DNS server names could map to the same IP address of a NAF, and the IP address would be used as NAF identifier.
   Regarding shared_key_TLS, there would be the possibility to have only one TLS connection between UE and a NAF, which is a reverse proxy. But the fundamental problem with this approach is that the UE may sit behind a forward proxy and may never know the IP address of the server. Therefore, this approach is considered infeasible.

From the four alternatives above, only 1 and 2 are proposed for further study. Alternative 3 is considered to come with too much configuration effort, and alternative 4 has the problem with forward proxies.

In the next section we show how sufficient flexibility could be built into the system so that both, alternatives 1 and 2, could be accommodated.

5. Flexible signalling of key derivation schemes

The last message in protocol A is a 200 OK from BSF to UE, which includes the transaction identifier. In TD S3-030341, Nokia proposed an encoding of the transaction identifier as an XML document in the HTTP response payload. The transaction identifier could be extended to include a “key derivation scheme identifier” KEY_DER_ID. Remember that KEY_DER_ID would be integrity-protected by http digest as it is part of the http payload.

It is believed that at most one byte would suffice for KEY_DER_ID. We show the effect of a particular value of KEY_DER_ID on the NAF identifier which is input to the key derivation parameter, by using the example of a UE accessing a NAF with server name “server1.presence.bootstrap.operator.com".

KEY_DER_ID could take the following values:

KEY_DER_ID = 0: this means no key derivation. The key shared between UE and NAF is Ks.

KEY_DER_ID = 1: this means that only the highest domain in the DNS server name is input to the key derivation. The NAF identifier is “com”.

KEY_DER_ID = 2: this means that the two highest domains in the DNS server name is input to the key derivation. The NAF identifier is “operator.com”.

KEY_DER_ID = 3: this means that the three highest domains in the DNS server name is input to the key derivation. The NAF identifier is “bootstrap.operator.com”.

Etc.

KEY_DER_ID = 7: this means that ALWAYS the FULL DNS server name is input to the key derivation (even if there should be more than seven domains in the server name). The NAF identifier is “server1.presence.bootstrap.operator.com".

In this way, flexibility could be built into the system so that several alternatives for key derivation could be accommodated. But this would, of course, somewhat increase the complexity of the system. It is ffs whether the full flexibility is needed.

Conclusions and Proposals

1. The Generic Bootstrapping Architecture is meant to be a generic tool to provide shared keys to UEs and application servers, independent of particular key uses. Section 2 showed threats, which become possible if only one application server is successfully attacked. SA3 is asked to endorse that the threats should be mitigated by appropriate provisions in the standard. In particular, it shall be prevented that a security breach in one application server can spread across the entire system.

2. It was proposed in section 3 to limit the effect of a security breach in one part of the system to a small part of the system by introducing key derivation for the keys shared between UE and NAF. SA3 is asked to endorse the use of a suitable key derivation procedure.

3. Section 4 proposed certain alternatives for the NAF identifier which is input to the key derivation parameters. The NAF identifier would determine the degree of assurance the UE gets about the identity of the NAF in NAF-to-UE authentication. SA3 is asked to agree to study only alternatives 1 (use DNS server name) or 2 (use defined parts of DNS server name) further and select between these alternatives at the next meeting.

4. Section 5 proposed a flexible mechanism to signal that one out of possibly multiple key derivation schemes be used with a certain key. As a minimum, the mechanism could be used to signal whether no key derivation or some pre-determined key derivation is used. SA3 is asked to endorse the use of this flexible signalling mechanism.

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).

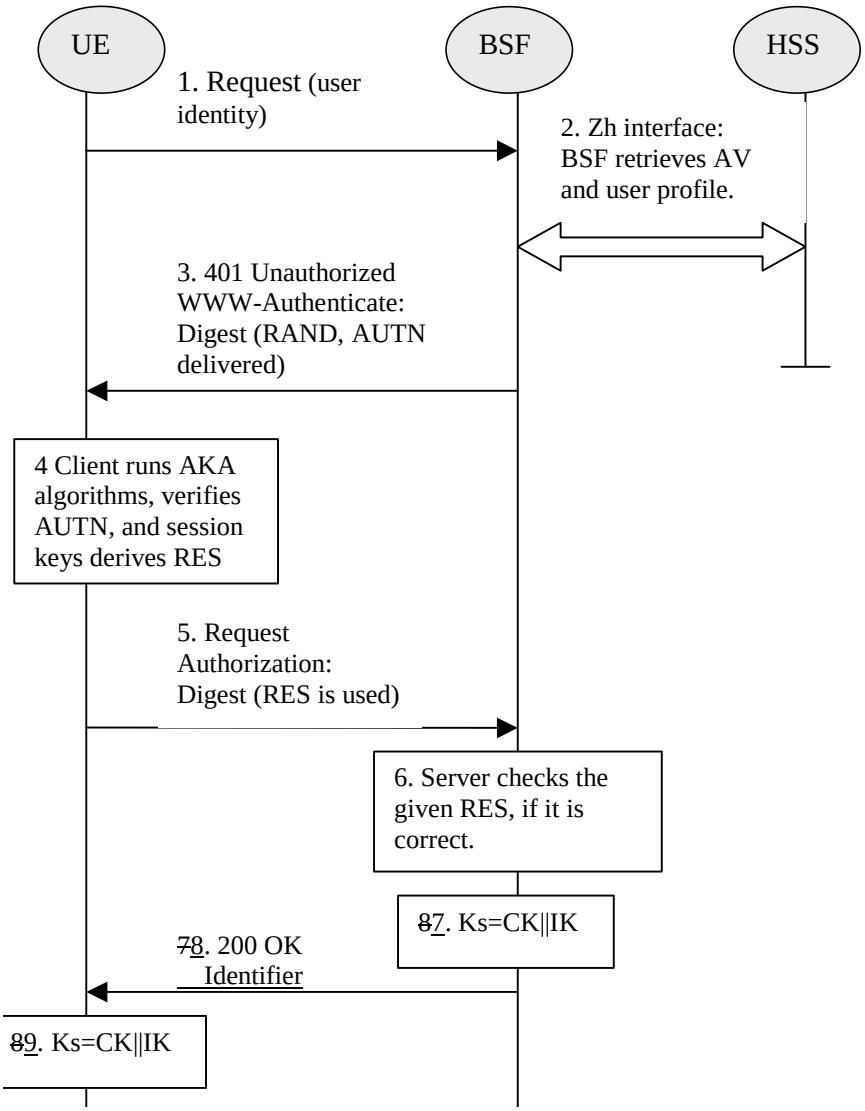

Figure 4: The bootstrapping procedure

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.

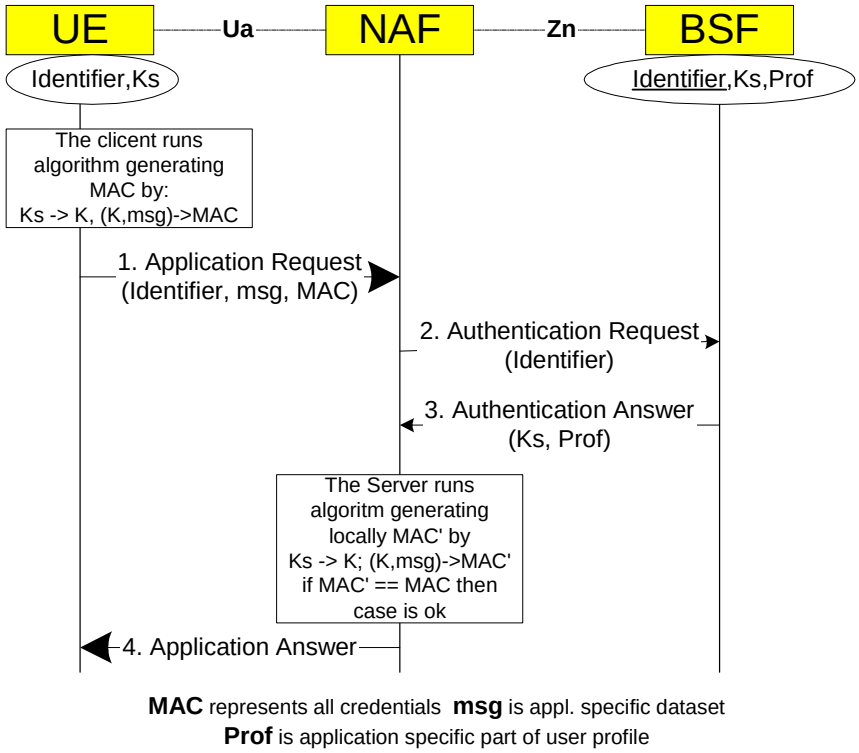

Figure 5: The bootstrapping usage procedure