ETSI TS 133 222 V12.3.0 (2014-10)

**Digital cellular telecommunications system (Phase 2+);
Universal Mobile Telecommunications System (UMTS);
LTE;
Generic Authentication Architecture (GAA);
Access to network application functions using Hypertext Transfer Protocol over Transport Layer Security (HTTPS)
(3GPP TS 33.222 version 12.3.0 Release 12)**

Reference
RTS/TSGS-0333222vc30

Keywords
GSM, LTE, SECURITY, UMTS

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Intellectual Property Rights

IPRs essential or potentially essential to the present document may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (http://ipr.etsi.org).

Pursuant to the ETSI IPR Policy, no investigation, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Foreword

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

The present document may refer to technical specifications or reports using their 3GPP identities, UMTS identities or GSM identities. These should be interpreted as being references to the corresponding ETSI deliverables.

The cross reference between GSM, UMTS, 3GPP and ETSI identities can be found under http://webapp.etsi.org/key/queryform.asp.

Modal verbs terminology

In the present document "**shall**", "**shall not**", "**should**", "**should not**", "**may**", "**may not**", "**need**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the ETSI Drafting Rules (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

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.

Introduction

A number of services might be accessed over HTTP. For the Presence Service, it shall be possible to manage the data on the Presence Server over the Ut reference point, which is based on HTTP. Other services like conferencing, messaging, push, etc. might be accessed using HTTP.

Access to services over HTTP can be done in a secure manner. The present document describes how the access over HTTP can be secured using TLS in the Generic Authentication Architecture.

1 Scope

The present document specifies secure access methods to Network Application Functions (NAF) using HTTP over TLS in the Generic Authentication Architecture (GAA), and provides Stage 2 security requirements, principles and procedures for the access. The present document describes both direct access to an Application Server (AS) and access to an Application Server through an Authentication Proxy (AP).

NOTE: Any application specific details for access to Applications Servers are not in scope of this specification and are covered in separate documents. An example of such a document is TS 33.141 [5], which specifies the security for presence services.

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 23.002: "Network architecture".

[2] 3GPP TS 22.250: "IP Multimedia Subsystem (IMS) group management"; Stage 1".

[3] 3GPP TS 33.220: "Generic Authentication Architecture (GAA); Generic Bootstrapping Architecture".

[4] 3GPP TR 33.919: "Generic Authentication Architecture (GAA); System description".

[5] 3GPP TS 33.141: "Presence Service; Security".

[6] Void.

[7] Void.

[8] Void.

[9] IETF RFC 2818 (2000): "HTTP Over TLS".

[10] IETF RFC 2617 (1999): "HTTP Authentication: Basic and Digest Access Authentication".

[11] IETF RFC 3310 (2002): "Hypertext Transfer Protocol (HTTP) Digest Authentication Using Authentication and Key Agreement (AKA)".

[12] IETF RFC 2616 (1999): "Hypertext Transfer Protocol (HTTP) – HTTP/1.1".

[13] 3GPP TS 33.210: "3G Security; Network Domain Security; IP network layer security".

[14] Void.

[15] Void.

[16] 3GPP TS 33.221: "Generic Authentication Architecture (GAA); Support for subscriber certificates".

[17] Void.

[18] 3GPP TS 24.109: "Bootstrapping interface (Ub) and network application function interface (Ua); Protocol details".

[19] 3GPP TS 29.109: "Generic Authentication Architecture (GAA), Zh and Zn Interface based on the Diameter protocol; Stage 3".

[20] 3GPP TS 33.310: "Network Domain Security (NDS); Authentication Framework (AF)".

[21] V oid.

[22] Void.

[23] 3GPP TR 21.905: "Vocabulary for 3GPP Specifications".

[24] W3C Working Draft (Jan 22, 2013): "HTML5.1 Nightly – A vocabulary and associated APIs for HTML and XHTML", work in progress, http://dev.w3.org/html5/spec/.

[25] IETF RFC 5929 (2010): "Channel Bindings for TLS".

[26] W3C Working Draft (Oct 20, 2011): "File API", work in progress, http://www.w3.org/TR/FileAPI/.

[27] W3C Candidate Recommendation (Dec 8, 2011): "Web Storage", work in progress, http://www.w3.org/TR/webstorage/

[28] 3GPP TS 33.203: "3G security; Access security for IP-based services".

[29] IETF RFC 5705 (2010): "Keying Material Exporters for Transport Layer Security (TLS)".

3 Definitions, symbols and abbreviations

3.1 Definitions

For the purposes of the present document, the terms and definitions given in TR 21.905 [23] and the following apply. A term defined in the present document takes precedence over the definition of the same term, if any, in TR 21.905 [23].

GBA web session: A GBA web session consists of a sequence of related HTTP request/response transactions together with some associated server-side state with the following additional requirement: During a GBA web session, a NAF can identify that the messages relate to the same individual GBA enabled terminal and a particular browser instance running in that terminal. The lifetime of the session is the lifetime of the Ks_js_NAF which is equal or shorter than the Ks_NAF lifetime and it is also equal or shorter than the lifetime of the TLS session, which was used to derive the Ks_js_NAF.

NOTE: The NAF and the UE may have to recalculate the key, when the TLS session is re-established.

HTML5: HTML5 is a W3C specification [24] that defines the fifth major revision of the Hypertext Markup Language (HTML), the standard language for describing the contents and appearance of Web pages.

HTML FORM: A HTML form is a section of a HTML document containing normal content, markup, special elements called controls (checkboxes, radio buttons, text fields, password fields, etc.) and labels on those controls. End users generally "complete" a form on a web page by modifying its controls (entering text, selecting radio buttons, etc.), before submitting the form to an agent for processing (e.g., to a web server).**HTTPS:** For the purpose of this document, HTTPS refers to the general concept securing the HTTP protocol using TLS. In some contexts, like in the IETF, the term HTTPS is used to refer to the reserved port number (443) for HTTP/TLS traffic.

JavaScript: JavaScript is a prototype-based scripting language that was formalized in the ECMAScript language standard. JavaScript is primarily used in the form of client-side JavaScript, implemented as part of a Web browser in order to provide enhanced user interfaces and dynamic websites.

Reverse Proxy: A reverse proxy is a web server system that is capable of serving web pages sourced from other web servers (AS), making these pages look like they originated at the reverse proxy.

Same origin policy: Same origin policy is a security mechanism in a client browser that permits webpage scripts to access their associated website"s data and methods but restricts its access to scripts and data stored by other websites.

Session management mechanism: A mechanism for creating stateful sessions when using the HTTP protocol.

3.2 Abbreviations

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

AP	Authentication Proxy
API	Application Programming Interface
AS	Application Server
B-TID	Bootstrapping Transaction Identifier
BSF	Bootstrapping Server Functionality
CA	Certification Authority
DNS	Domain Name System
FQDN	Fully Qualified Domain Name
GBA	Generic Bootstrapping Architecture
HSS	Home Subscriber System
HTML	HyperText Markup Language
HTTP	HypertTxt Transfer Protocol
HTTPS	HTTP over TLS
IMPI	IP Multimedia Private Identity
IMPU	IP Multimedia Public Identity
ME	Mobile Equipment
NAF	Network Application Function
NAF_ID	NAF identifier
TLS	Transport Layer Security
UE	User Equipment
URL	Uniform Resource Locator

4 Overview of the Security Architecture

The overall security architecture conforms to the architecture defined in TS 33.220 [3]. Details of the solution with an authentication proxy are given in clause 6.

5 Authentication schemes

5.1 Reference model

Figure 1 shows a network model of the entities that utilize the bootstrapped secrets, and the reference points used between them.

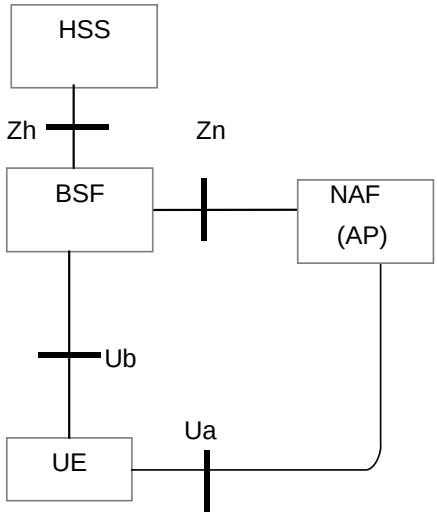

Figure 1: High level reference model for NAF using a bootstrapping service

5.2 General requirements and principles

This document is based on the architecture specified in TS 33.220 [3]. All notions not explained here can be found in TS 33.220 [3]. For the purposes of the present document Ks_(ext)_NAF refers to the key shared between the UE and a NAF. In the case of GBA_U, Ks_(ext)_NAF refers to Ks_ext_NAF, and in the case of GBA_ME, Ks_(ext)_NAF refers to Ks_NAF. Ks_int_NAF refers to the key shared between the UICC and a NAF.

The UE shall be able to indicate to the NAF which key (Ks_(ext)_NAF or Ks_int_NAF) the UE intends to use to secure the HTTPS Ua reference point.

The subscriber's home operator shall be able to require that a certain key (i.e., Ks_(ext)_NAF or Ks_int_NAF) shall be used to secure the HTTPS access between the UE and the NAF. This home operator control is exercised using USS.

5.2.1 Requirements on the UE

To utilise GBA as described in this document the UE shall be equipped with a HTTPS capable client (e.g. browser) implementing the particular features of GBA as specified in TS 33.220 [3].

The UE hosts the HTTPS client (i.e. both the HTTP client and the TLS client). The HTTP client and TLS client either resides both in the ME or in the UICC. The HTTPS client may reside in the ME or in the UICC or both might host an HTTPS client independently of each other. When the HTTPS capable client to be used is in the ME, Ks_(ext)_NAF shall be used as the shared key between the UE and the NAF. When the HTTPS capable client to be used is located in the UICC, Ks_int_NAF shall be used as the shared key between the UE and the NAF.

5.2.2 Requirements on the NAF

To utilise GBA as described in this document the NAF shall support the features of GBA as specified in TS 33.220 [3].

It shall be possible that the NAF is configured to restrict the access to the service based on which key is used, (e.g., access is allowed only for those HTTPS capable clients that reside in the UICC and use Ks_int_NAF). The key selection indication given in the USS shall overrule the local policy of the NAF.

NOTE: The support of GBA_U is optional for the NAF. However, as indicated in TS 33.220 [3], the use of Ks_ext_NAF is supported by NAFs, which are GBA_U unaware.

Additionally in the scope of this specification, HTTP and TLS shall be supported by the NAF for the UE-NAF reference point (Ua).

5.3 Shared key-based UE authentication with certificate-based NAF authentication

5.3.0 Procedures

The authentication mechanism described in this section for ME-based application is mandatory to implement in ME and optional to implement in NAF.

The authentication mechanism described in this section for UICC-based application is mandatory to implement in the UICC and optional to implement in the NAF.

This section explains how the procedures specified in TS 33.220 [3] have to be enhanced when HTTPS is used between a ME and a NAF or between the UICC and the NAF. The following gives the complementary description with respect to the procedure specified in clauses 4.5.3, 5.3.3, I.5.3 and M.6.4 of TS 33.220 [3]. This document specifies the logical information carried in some header fields. The exact definition of header fields is part of TS 29.109 [19] and TS 24.109 [18]. In the text below, the HTTPS client can reside in the ME or in the UICC.

NOTE 0: It should be noted that the term "GBA mode" does not refer to used bootstrapping mechanism, but it refers to NAF keys in the following way:
- "3GPP-bootstrapping" and "3gpp-gba" refer to Ks_NAF from GBA_ME, Ks_ext_NAF from GBA_U or Ks_NAF from 2G GBA;
- "3GPP-bootstrapping-uicc" and "3gpp-gba-uicc" refer to Ks_int_NAF from GBA_U;
- "3GPP-bootstrapping-digest" and "3gpp-gba-digest" refer to Ks_NAF from GBA_Digest.

1) When the HTTPS client starts communication via Ua reference point with the NAF, it shall establish a TLS tunnel with the NAF. The NAF is authenticated to the HTTPS client by means of a public key certificate. The HTTPS client shall verify that the server certificate corresponds to the FQDN of the NAF it established the tunnel with. No client authentication is performed as part of TLS (no client certificate necessary).

2) The HTTPS client sends an HTTP request to the NAF inside the TLS tunnel (HTTPS, i.e. HTTP over TLS). The HTTPS client shall indicate to the NAF that GBA-based authentication is supported by adding one or more constant string(s) to the "User-Agent" HTTP header as product tokens as specified in IETF RFC 2616 [12]. If the client supports AKA-based authentication the constant string added shall be either "3gpp-gba" for ME-based applications or "3gpp-gba-uicc" for UICC based applications. Or, if the client supports GBA_Digest it shall add the constant string '3gpp-gba-digest' The UE shall send the hostname of the NAF in "Host" HTTP header.

NOTE 1: The ability to send the hostname of the NAF is particularly necessary if a NAF can be addressed using different hostnames, and the NAF cannot otherwise discover what is the hostname that the HTTPS client used to contact the NAF. The hostname is needed by the BSF during key derivation

NOTE 1a: Sending the product tokens in the HTTP request enables the optimisation that the NAF can reject the request right away based on a mismatch of GBA mode in the client and the NAF, cf. next step.

3) In response to the HTTP request received from HTTPS client over the Ua reference point, the NAF shall invoke HTTP digest as specified in RFC 2617 [10] with the HTTPS client in order to perform client authentication using the shared key as specified in clauses 4.5.3 and 5.3.3 of TS 33.220 [3].

   The NAF first verifies that the product tokens received in the HTTP request in step 2, if any, indicates a GBA mode acceptable to the NAF. If so, the NAF selects one acceptable GBA mode and includes the corresponding realm attribute within the WWW-Authenticate header field i.e. the realm attribute shall contain the constant string "3GPP-bootstrapping" (in case "3gpp-gba" is the selected GBA mode), or "3GPP-bootstrapping-uicc" (in case "3gpp-gba-uicc" is the selected GBA mode), or "3GPP-bootstrapping-digest" (in case "3gpp-gba-digest" is the selected GBA mode), and the FQDN of the NAF (for all cases) ., In the selection of the GBA mode by the NAF, AKA-based modes shall take priority over GBA_Digest.

If the NAF has been configured to forbid the access to the service for all indicated GBA modes (e.g. the HTTP request contains "3gpp-gba" whilst the NAF configuration for this service requires that Ks_int_NAF shall be used) or if the NAF does not support any of the indicated GBA modes (e.g. when a NAF, which is GBA_U unaware receives an HTTP request with "3gpp-gba-uicc" in "User-Agent" HTTP header) then the NAF shall respond with the appropriate error code and terminate the TLS connection with the UE.

4) On receipt of the response from the NAF, the HTTPS client shall verify that the FQDN in the realm attribute corresponds to the FQDN of the NAF it established the TLS connection with. On failure the HTTPS client shall terminate the TLS connection with the NAF.Furthermore, if no key Ks corresponding to the GBA mode indicated by the NAF in step 3 is available in the UE the UE shall perform a run of the Ub protocol with the BSF for the corresponding GBA mode. If no such key Ks can be obtained the UE shall abandon the GBA-based authentication.

5) In the following request to NAF the HTTPS client sends a response with an Authorization header field where Digest is inserted using the B-TID as username.The NAF-specific key (Ks_(ext)_NAF in the case of ME-based application (including GBA_Digest) or Ks_int_NAF in the case of UICC-based application) is used as password in the Digest calculation.

6) On receipt of this request the NAF shall verify the value of the password attribute by means of the NAF-specific key (Ks_(ext)_NAF or Ks_int_NAF) retrieved from BSF over Zn using the B-TID received as user name attribute in the query.

   If the NAF has requested a USS, and the USS indicates to the NAF that the Ks_int_NAF shall be used for HTTPS, then the NAF shall only accept the use of Ks_int_NAF as the NAF specific key. Therefore, if the Ks_(ext)_NAF was used as the NAF specific key with the HTTPS client, then the NAF shall respond with the appropriate error code and terminate the TLS connection with the UE. For information on usage of USS see Annex J in TS 33.220 [3].

   If the NAF is not able to obtain any NAF-specific key from the BSF the NAF shall respond with an appropriate error message not containing the realm attributes from step 3.

NOTE 1b: The last sentence in step 6 captures a failure case where both the ME and NAF support GBA, but the NAF cannot communicate with the BSF, e.g. for lack of commercial agreements. This allows UE and NAF to proceed with a different authentication scheme, if available.

7) After the completion of step 6), UE and NAF are mutually authenticated as the TLS tunnel endpoints.

NOTE 2: RFC 2617 [10] mandates in section 3.3 that all further HTTP requests to the same realm must contain the Authorization request header field, otherwise the server has to send a new "401 Unauthorized" with a new WWW-Authenticate header. In principle it is not necessary to send an Authorization header in each new HTTP request for security reasons as long as the TLS tunnel exists, but this would not conform to RFC 2617 [10].

In addition, there may be problems with the lifetime of a TLS session, as the TLS session may time-out at unpredictable (at least for the UE) times, so any request sent by UE can be the first request inside a newly established TLS tunnel requiring the NAF to re-check user credentials.

It shall be possible for the AP/AS to request a re-authentication of an active UE using a bootstrapping renegotiation request, see clauses 4.5.3, 5.3.3, I.5.3 and M.6.4 of TS 33.220 [11].

5.3.1 TLS profile

5.3.1.0 General

The UE and the NAF shall support TLS and TLS Extensions according to the TLS profile given in TS 33.310 [20], Annex E.

Support of certificate revocation and of the related fields in certificates is optional. If supported, the certificate and CRL profiles in clause 6.1 and 6.1a of TS 33.310 [20] should be followed.

NOTE 1: The management of Root Certificates is out of scope of this Technical Specification.

The UE and the NAF shall support the server_name TLS extension. All other TLS extensions are optional for implementation.

NOTE 2: If the NAF is doing virtual name based hosting (e.g. in the case of authentication proxy, see Annex A), the NAF needs to either have a TLS server certificate that contains all the hostnames that the NAF can be addressed with (i.e. virtual hostnames), or have one TLS server certificate for each of the hostnames mentioned above. In the latter case, the server_name extension is needed because the NAF needs to be able to select the correct TLS server certificate.

5.3.1.1 Protection mechanisms

The rules on allowed and mandatory ciphersuites are given in TS 33.310 [20], Annex E .

Cipher Suites with NULL encryption may be used. If NULL encryption is implemented and used, cipher suites according to the TLS profile given in TS 33.310 [20], Annex E, shall be supported. The UE shall always include at least one cipher suite that supports encryption during the handshake phase.

5.3.1.2 Void

5.3.1.3 Authentication of the AP/AS

The AP/AS is authenticated by the Client by use of a server certificate. The client shall match the server name as specified in RFC 2818 [9] section 3.1.

The AP/AS certificate profile shall comply with the requirements for TLS certificates in clause 6.1 of TS 33.310 [20].

5.3.1.4 Authentication Failures

If the UE receives a Server Hello Message from the AP/AS that requests a Certificate then the UE shall respond with a Certificate Message containing no Certificate if it does not have a certificate. The AP/AS upon receiving this message may respond with a failure alert, however if the AP/AS shall authenticate the UE as configured by the policy of the operator the AP/AS should continue the dialogue and assume that the UE will be authenticated as specified in TS 33.220 [11].

If there is no response within a given time limit from a network initiated re-authentication request an authentication failure has occurred after that the request has been attempted for a limited number of times. This failure can be due to several reasons, e.g. that the UE has powered off or due to that the message was lost due to a bad radio channel. The AP/AS shall then still assume that if a TLS session is still valid that it can be re-used by the UE at a later time. Should then the UE re-use an existing session then the AP/AS shall re-authenticate the UE and not give access to the AP/AS unless the authentication was successful.

5.3.1.5 Set-up of Security parameters

The TLS Handshake Protocol negotiates a session, which is identified by a Session ID. The Client and the AP/AS shall allow for resuming a session. This facilitates that a Client and Server may resume a previous session or duplicate an existing session. The lifetime of a Session ID is maximum 24 hours. The Session ID shall only be used under its lifetime and shall be considered by both the Client and the Server as obsolete when the Lifetime has expired.

5.3.1.6 Error cases

The AP/AS shall consider the following cases as a fatal error:

- if the received ciphersuites do not comply with the TLS profile.

5.4 Shared key-based mutual authentication between UE and NAF

5.4.0 Procedures

The authentication mechanism described in this section for ME-based application is optional to implement in ME and NAF.

The authentication mechanism described in this section for UICC-based application is optional to implement in UICC and NAF.

The HTTP client and server may authenticate each other based on the shared key generated during the bootstrapping procedure. The shared key shall be used as a master key to generate TLS session keys, and also be used as the proof of secret key possession as part of the authentication function. The usage of Pre-Shared Key Ciphersuites for Transport Layer Security (TLS) is specified in the TLS profile given in TS 33.310 [20], Annex E.

This section explains how a GBA-based shared secret that is established between the UE and the BSF as specified in TS 33.220 [3] is used with Pre-Shared Key (PSK) Ciphersuites for TLS according to the TLS profile given in TS 33.310 [20], Annex E. The HTTPS client may reside in the ME or in the UICC. In former case, Ks_(ext)_NAF shall be used to establish the TLS session keys. In latter case, Ks_int_NAF shall be used to establish the TLS session keys.

NOTE 0: It should be noted that the term "GBA mode" does not refer to used bootstrapping mechanism, but it refers to NAF keys in the following way:
- "3GPP-bootstrapping" and "3gpp-gba" refer to Ks_NAF from GBA_ME, Ks_ext_NAF from GBA_U or Ks_NAF from 2G GBA;
- "3GPP-bootstrapping-uicc" and "3gpp-gba-uicc" refer to Ks_int_NAF from GBA_U;
- "3GPP-bootstrapping-digest" and "3gpp-gba-digest" refer to Ks_NAF from GBA_Digest.

1. When an UE contacts a NAF, it may indicate to the NAF that it supports PSK-based TLS by adding one or more PSK-based ciphersuites to the ClientHello message. The UE shall include ciphersuites other than PSK-based ciphersuites in the ClientHello message. The UE shall send the hostname of the NAF using the server_name extension to the ClientHello message according to TLS extensions.

NOTE 1: The ability to send the hostname of the NAF is particularly necessary if a NAF can be addressed using different hostnames, and the NAF cannot otherwise discover what is the hostname that the UE used to contact the NAF. The hostname is needed by the BSF during key derivation.

NOTE 2: When the UE adds one or more PSK-based ciphersuites to the ClientHello message, this can be seen as an indication that the UE supports PSK-based TLS. If the UE supports PSK-based ciphersuites but not GBA-based authentication, the TLS handshake will fail if the NAF selected the PSK-based ciphersuite and suggested to use GBA (as described in step 2). In this case, the UE should attempt to establish the TLS tunnel with the NAF without including PSK-based ciphersuites to the CientHello message, according to the procedure specified in clause 5.3. This note does not limit the use of PSK TLS to HTTP-based services.

2. If the NAF is willing to establish a TLS tunnel using a PSK-based ciphersuite, it shall select one of the PSK-based ciphersuites offered by the UE, and send the selected ciphersuite to the UE in the ServerHello message.

   The NAF shall send the ServerKeyExchange message with a list of PSK-identity hints. A constant string "3GPP-bootstrapping" is used as PSK-identity hint to indicate the local configuration in the NAF i.e. that the NAF accepts that AKA-based Ks_(ext)_NAF is used establish the TLS session keys. A constant string "3GPP-bootstrapping-uicc" is used as PSK-identity hint to indicate that the local configuration in the NAF accepts that Ks_int_NAF is used to establish the TLS sessions keys. A constant string "3GPP-bootstrapping-digest" is used as PSK-identity hint to indicate that the local configuration in the NAF accepts that GBA_Digest-based Ks_NAF is used to establish the TLS sessions keys. One of these PSK-identity hints shall be present in the ServerKeyExchange message, and it shall indicate the GBA as the required authentication method. If the local configuration in the NAF allows several authentication methods to be used to access its service then the ServerKeyExchange message shall include the PSK-identity hints for all allowed authentication methods. Also other PSK-identity hints may be supported, however, they are out of the scope of this specification. The NAF finishes the reply to the UE by sending a ServerHelloDone message.

NOTE 3: If the NAF does not wish to establish a TLS tunnel using a PSK-based ciphersuite, it shall select a non-PSK-based ciphersuite and continue TLS tunnel establishment based on the procedure described either in clause 5.3 or clause 5.5.

3. The UE shall use a GBA-based shared secret for PSK TLS, if the NAF has sent a ServerHello message containing a PSK-based ciphersuite, and a ServerKeyExchange message containing at least one of the constant strings "3GPP-bootstrapping", "3GPP-bootstrapping-uicc", or "3GPP-bootstrapping-digest" as the PSK identity hint. If the UE does not have a valid GBA-based shared secret it shall obtain one by running the bootstrapping procedure with the BSF over the Ub reference point as specified in TS 33.220 [3].

   If the HTTPS client resides in the ME, Ks_(ext)_NAF shall be used as the GBA shared key. If the HTTPS client resides in the UICC, Ks_int_NAF shall be used as the GBA shared key. In the selection of the GBA mode by the UE, AKA-based modes shall take priority over GBA_Digest.

   The UE derives the TLS premaster secret from the NAF specific key (Ks_(ext)_NAF if the initiating HTTPS client resides on the ME or Ks_int_NAF if the initiating HTTP client resides on the UICC).

   The UE shall send a ClientKeyExchange message. The PSK identity in the ClientKeyExchange message shall include a prefix indicating the PSK-identity name space that was selected (i.e. "3GPP-bootstrapping-uicc", "3GPP-bootstrapping", or "3GPP-bootstrapping-digest"), and the B-TID. The prefix shall match one of the PSK-identity hints that NAF offered in ServerKeyExchange message. The precise format of the PSK identity is specified in TS 24.109 [18]. The UE concludes the TLS handshake by sending the ChangeCipherSuite and Finished messages to the NAF.

4. If the NAF receives the "3GPP-bootstrapping" prefix and the B-TID in the ClientKeyExchange messages it fetches the NAF specific shared secret (Ks_(ext)_NAF) from the BSF using the B-TID, else if the NAF receives the "3GPP-bootstrapping-uicc" prefix and the B-TID in the ClientKeyExchange messages it fetches the NAF specific shared secret (Ks_int_NAF) from the BSF using the B-TID. If the NAF receives the "3GPP-bootstrapping-digest" prefix and the B-TID in the ClientKeyExchange messages it shall indicate to the BSF that GBA_Digest is acceptable.

   If the NAF has requested a USS, and the USS indicates to the NAF that only the Ks_int_NAF shall be allowed, then the NAF shall only accept the Ks_int_NAF as the NAF specific key. If the Ks_(ext)_NAF was used as the NAF specific key, the NAF shall respond with the appropriate error code and terminate the TLS connection with the UE.

   The NAF derives the TLS premaster secret from the NAF specific key (Ks_(ext)_NAF or Ks_int_NAF).

   The NAF concludes the TLS handshake by sending the ChangeCipherSuite and Finished messages to the UE.

The UE and the NAF have established a TLS tunnel using GBA-based shared secret, and then may start to use the application level communication through this tunnel.

5.4.1 TLS Profile

5.4.1.0 General

If the PSK TLS based authentication mechanism is supported, the HTTPS client in the UE or the NAF shall support the TLS version, PSK Ciphersuites and TLS Extensions as specified in the TLS profile given in TS 33.310 [20], Annex E.

The HTTPS client in the UE and the NAF shall support the server_name TLS extension. All other TLS extensions are optional for implementation.

NOTE: If the NAF is doing virtual name based hosting (e.g. in the case of authentication proxy, see Annex A), the NAF needs to be able to discover the correct server name to indicate the correct NAF_ID to the BSF. Otherwise the BSF is not able to derive the correct NAF specific keys.

5.4.1.1 Protection mechanisms

The same requirements as in clause 5.3.1.1 of the present document shall apply.

5.4.1.2 Authentication of the AP/AS

The AP/AS is authenticated by the Client according to PSK TLS as specified in Annex E of TS 33.310 [20].

5.4.1.3 Authentication Failures

If there is no response within a given time limit from a network initiated re-authentication request an authentication failure has occurred after that the request has been attempted for a limited number of times. This failure can be due to several reasons, e.g. that the UE has powered off or due to that the message was lost due to a bad radio channel. The AP/AS shall then still assume that if a TLS session is still valid that it can be re-used by the UE at a later time. Should then the UE re-use an existing session then the AP/AS shall re-authenticate the UE and not give access to the AP/AS unless the authentication was successful.

If the AP/AS, acting as NAF, has requested a USS, and the USS indicates to the NAF that only the Ks_int_NAF shall be allowed, then the NAF shall only accept the Ks_int_NAF as the NAF specific key therefore if the Ks_(ext)_NAF was used as the NAF specific key, then the NAF shall respond with the appropriate error code and terminate the TLS connection with the UE.

5.4.1.4 Set-up of Security parameters

The TLS Handshake Protocol negotiates a session, which is identified by a Session ID. The Client and the AP/AS shall allow for resuming a session. This facilitates that a Client and Server may resume a previous session or duplicate an existing session. The lifetime of a Session ID is the lifetime of the GAA shared secret or maximum of 24 hours. The Session ID shall only be used under its lifetime and shall be considered by both the Client and the Server as obsolete when the Lifetime has expired.

5.5 Certificate based mutual authentication between UE and application server

5.5.1 General

The authentication mechanism described in this section is optional to implement in UE and AS.

The certificate based mutual authentication between an UE and an application server shall be based on TLS and TLS Extensions as specified in the TLS profile given in TS 33.310 [20], Annex E.

Annex B of this specification provides guidance on certificate mutual authentication between UE and application server.

5.5.2 TLS Profile

5.5.2.1 General

The UE and the AS shall support TLS and TLS Extensions according to the TLS profile given in TS 33.310 [20], Annex E.

Support of certificate revocation and of the related fields in certificates is optional. If supported, the certificate and CRL profiles in clause 6.1 and 6.1a of TS 33.310 [20] should be followed.

NOTE 1: The management of Root Certificates is out of scope of this Technical Specification.

The UE and the AS shall support the server_name TLS extension. All other TLS extensions are optional for implementation.

NOTE 2: If the AS is doing virtual name base hosting (e.g. in the case of authentication proxy, see Annex A) the AS needs to either have a TLS server certificate that contains all the host names that the AS can be addressed with (i.e. virtual hostnames), or have one TLS server certificate for each of the hostnames mentioned above. In the latter case, the server_name extension is needed because the AS needs to be able to select the correct TLS server certificate.

5.5.2.2 Protection mechanisms

The same requirements as in clause 5.3.1.1 of the present document apply.

5.5.2.3 void

6 Use of Authentication Proxy

An Authentication Proxy (AP) is an HTTP proxy which takes the role of a NAF for the UE. It handles the TLS security relation with the UE and relieves the application server (AS) of this task. Based on GBA the AP can assure the ASs that the request is coming from an authorized subscriber of the MNO.

6.1 Architectural view

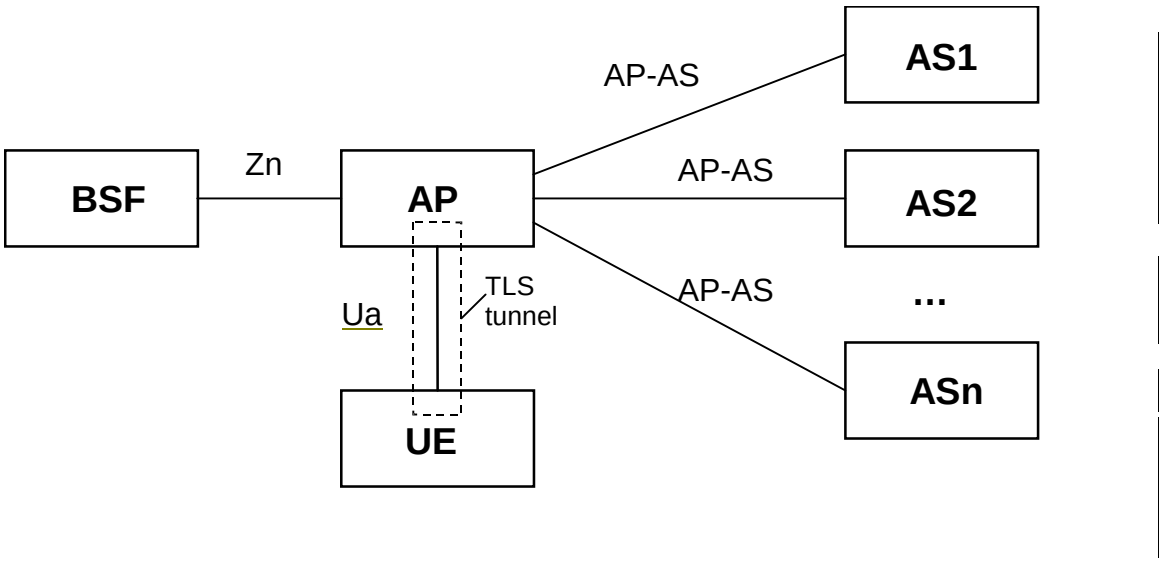

Figure 2: Environment and reference points of AP

The use of an authentication proxy (AP) is fully compatible with the architecture specified in TS 33.220 [3] and in clauses 4 and 5 of this specification. When an AP is used in this architecture, the AP takes the role of a NAF. When an HTTPS request is destined towards an application server (AS) behind an AP, the AP terminates the TLS tunnel and performs UE authentication. The AP proxies the HTTP requests received from UE to one or many application servers. The AP may add an assertion of identity of the subscriber for use by the AS, when the AP forwards the request from the UE to the AS.

NOTE: As an example, the following condition allows accessing multiple application servers AS(s) through one shared TLS tunnel (over the Ua interface) AS_n hostname = AP hostname = NAF_ID (e.g.: *services.operator.com*). There might be alternative ways to access multiple ASs behind an AP.

Figure 3 presents an architectural view of using Authentication Proxy, for example, for IMS SIP based services. The UE shall manipulate own data such as groups, through the Ua/Ut reference point. The reference point Ut specified in TS 23.002 [1] shall be applicable to data manipulation of IMS based SIP services, such as Presence, Messaging and Conferencing services. The stage 1 requirements are specified in TS 22.250 [2].

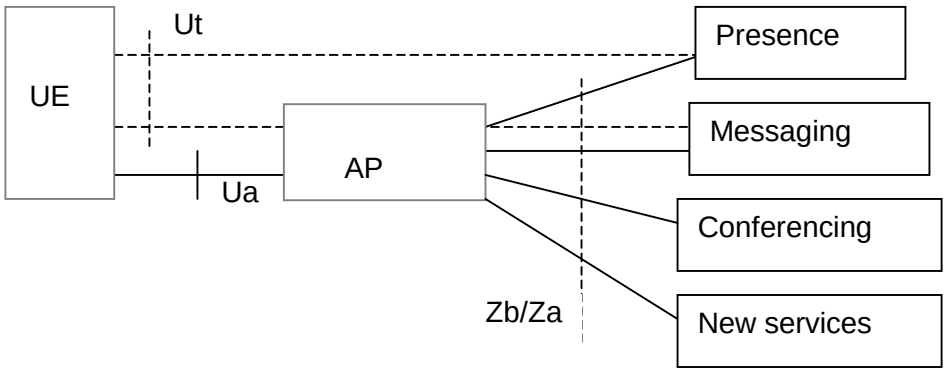

Figure 3: The architectural view using Authentication Proxy for IMS SIP based services

Management of UE identities is described in clause 6.5.

Annex A contains further guidance on technical solutions for authentication proxies.

6.2 Requirements and principles

The authentication proxy may reside between the UE and the AS as depicted in figure 2. The usefulness of an Authentication Proxy may be to reduce the consumption of authentication vectors and/or to minimize SQN synchronization failures. Also the AP relieves the AS of security tasks.

The following requirements apply for the use of an Authentication Proxy:

- authentication proxy shall be able to authenticate the UE using the means of Generic Bootstrapping Architecture, as specified in TS 33.220 [3];
- if the application server requires an authenticated identity of the UE the authentication proxy shall send it to the application server belonging to the trust domain with every HTTP request;
- if required, the authentication proxy may not reveal the authenticated identity of the UE to the application server not belonging to the trust domain;
- the authenticated identity management mechanism shall not prevent the application server to use an appropriate session management mechanisms with the client;
- the UE shall be able to create multiple parallel HTTP sessions via the authentication proxy towards different application servers;

NOTE 1: The used session management mechanism is out of the scope of 3GPP specifications.

NOTE 2: One motivation for having AP between UE and AS's is to minimize the number of TLS connections. However, there are situations when UE and AP may end-up having parallel TLS connections, e.g. if two applications in the UE are not able to share the same TLS connection.

- implementation of check of asserted user identity in the AS is optional;
- activation of transfer of asserted user identity shall be configurable in the AP on a per AS basis.

The use of an authentication proxy should be such that there is no need to manage the authentication proxy configuration in the UE.

NOTE 3: This requirement implies that the authentication proxy should be a reverse proxy in the following sense: A reverse proxy is a web server system that is capable of serving web pages sourced from other web servers - in addition to web pages on disk or generated dynamically by CGI - making these pages look like they originated at the reverse proxy.

6.4 Reference points

6.4.1 Ua reference point

The Ua reference point is standardised in specification TS 33.220 [3] and in clauses 4 and 5 of this specification.

NOTE: The optional introduction of an AP has advantages which are stated elsewhere. However, the following consequences should be taken into account to decide whether an AP is to be used:

- The AP terminates TLS and HTTP digest. This relieves the AS of the burden to handle TLS and HTTP digest, but it should be noted that then the UE is not able to establish an additional end-to-end TLS tunnel to the AS, nor can the UE additionally authenticates itself to AS by use of client authentication within TLS. Furthermore, if GBA authentication uses HTTP Digest Authentication, then the UE cannot use Basic or Digest Authentication directly with AS.

6.4.2 AP-AS reference point

The HTTP protocol is run over the AP-AS reference point.

Confidentiality and integrity protection can be provided for the reference point between the AP and the AS using NDS/IP mechanisms as specified in TS 33.210 [13]. For traffic between different security domains, the Za reference point shall be operated. For traffic inside a security domain, it is up to the operator to decide whether to deploy the Zb reference point. As AP terminates the TLS tunnel from UE, also a TLS tunnel is possible.

The AP may support the transfer of an identity of the UE authenticated by the AP from AP to AS in a standardised format. The format of this information element in the HTTP request header is left to stage 3 specifications.

6.5 Management of UE identity

Different ASs need different kinds of authentication information. To support the requirements of different servers, the AP needs to perform authentication with varying granularity and with varying degree of assertion to the AS. The authentication and the corresponding assertion is therefore AS specific and has to be configured in the AP per AS.

6.5.1 Granularity of Authentication and Access Control by AP

The AP is configured per AS if the particular application or applications served by the AS is in need of an application specific user security setting, see definitions in TS 33.220 [3]. This user security setting may contain the public user identities in the authentication part of the USS. The authorisation part of the USS may contain indications, which of the applications residing on the AP, and the Application Servers behind the AP, a user is allowed to access.

NOTE: There are two ways of implementing application specific user security settings (USS). One can either assign a GSID (GAA Service Identifier) to each application and store multiple USS in the BSF/HSS or one can assign a GSID to the AP and store a single USS in the BSF/HSS. In the latter case the USS contains identity and authorization information for all the ASs/applications served by the AP. Unless indicated otherwise, the term "application specific user security settings" refers to both implementation options.

6.5.1.1 Authorised Participant of GBA

The AP checks that the UE is an authorised participant of GBA. Access is granted on success of the basic GBA mechanism, i.e. the HTTPS client in the UE sends a valid B-TID and performs digest authentication with the NAF specific keys received from BSF.

The AP is configured not to request an application specific user security setting from BSF for the AS named in the request. Depending on configuration of BSF the AP may receives the private user identity (IMPI) from BSF.

This case shall be supported by AP.

NOTE: This case may apply when all subscribers of an operator, but no other users, are allowed access to operator defined services. The BSF may not send the IMPI out of privacy considerations or because the AP does not need it. If the BSF does not send the IMPI to the AP, the user remains anonymous towards the AP; or more precisely, the B-TID functions as a temporary user pseudonym.

6.5.1.2 Authorised User of Application

The AP is configured to request an application specific user security setting from the BSF. Depending on the policy of the BSF, the AP receives the application specific user security setting and the private user identity (IMPI) from the BSF. Access is granted if allowed according to the application specific user security setting received from BSF.

The AP may do further checks on user inserted identities in the HTTP request if required according to clause 6.5.2.4.

This case shall be supported by AP.

NOTE: If there is no application specific user security setting configured for an application, this case reduces to authentication according to clause 6.5.1.1.

6.5.2 Transfer of Asserted Identity from AP to AS

The AP is configured per AS to perform authentication and access control according to one of the following subclauses: if required in the subclause, the user identity is transferred to AS in every HTTP request proxied to AS.

6.5.2.1 Authorised Participant of GBA

The AP checks that the UE is an authorised participant of GBA. If the authentication of the UE by the AP fails, the AP does not forward the request of the UE to the AS.

This case shall be supported by AP.

NOTE: This case simply implies that the NAF checks that the user is known to, and has established a valid key, with the BSF, according to the GBA procedures described in TS 33.220 [3].

6.5.2.2 Authorised User of Application Anonymous to AS

The AP checks that the UE is an authorised user of the application according to application specific user security setting received from BSF. No user identity shall be transferred to AS.

This case shall be supported by AP.

6.5.2.3 Authorised User of Application with Transferred Identity asserted to AS

The AP checks that the UE is an authorised user of the application. The user identity (or user identities) received from the BSF shall be transferred to AS. Based on AS-specific configuration of the AP, any authorization flags existing in application-specfic user security settings shall also be transferred to AS.

Depending on the application specific user security setting and the AS-specific configuration of the AP, the transferred user identity (or identities) may be the private user identity (IMPI), or may be taken from the application specific user security setting (e.g. an IMPU), or may be a pseudonym chosen by AP (e.g. Random, B-TID).

This case may be supported by AP.

NOTE 1: If the AP is configured to transfer a pseudonym to AS, any binding of this pseudonym to the user identity (e.g. for charging purposes by AS) is out of scope of this specification.

NOTE 2: If the AP is configured not to request an application specific user security setting from BSF, only the private user identity (IMPI) or a pseudonym may be transferred to AS. In this case any authorised participant of GBA is supposed to be an authorised user of the application.

6.5.2.4 Authorised User of Application with Transferred Identity asserted to AS and Check of User Inserted Identity

This case resembles clause 6.5.2.3 with the following extension:

Based on the user identity received from BSF, the AP authenticates user related identity information elements as sent from UE. These "user inserted identities" may occur within header fields or within the body of the HTTP request.

Depending on application specific user security setting and AS-specific configuration of AP, all user-inserted identities (or a subset thereof) are authenticated by checking against the private user identity (IMPI) or the application specific user security setting.

Depending on the application specific user security setting and the AS-specific configuration of AP, the transferred user identity (or identities) may also be selected from the authenticated user inserted identities.

This case may be supported by AP.

NOTE 1: If AP authenticates certain or all user related identity information elements of a request, and the AS shall rely on the check of these elements, then a corresponding policy between the AP and the AS needs to be in place between the AP and the AS.

NOTE 2: Any application specific details are beyond the scope of this document and may be specified within the application, e.g. for Presence in TS 33.141 [5]. This specification does not preclude that any other application specific specifications (e.g. Presence) declare this feature as mandatory in their scope.

Annex A (informative): Technical Solutions for Access to Application Servers via Authentication Proxy and HTTPS

This annex gives some guidance on the technical solution for authentication proxies so as to help avoid misconfigurations. An authentication proxy acts as reverse proxy which serves web pages (and other content) sourced from other web servers (AS) making these pages look like they originated at the proxy.

To access different hosts with different DNS names on one server (in this case the proxy) the concept of virtual hosts was created.

One solution when running HTTPS is to associate each host name with a different IP address (IP based virtual hosts). This can be achieved by the machine having several physical network connections, or by use of virtual interfaces which are supported by most modern operating systems (frequently called "*ip aliases*"). This solution uses up one IP address per AS and it does not allow the notion of "one TLS tunnel from UE to AP-NAF" for all applications behind a NAF together.

If it is desired to use one IP address only or if "one TLS tunnel for all" is required, only the concept of name-based virtual hosts is applicable. Together with HTTPS, however, this creates problems, necessitating workarounds which may deviate from standard behaviour of proxies and/or browsers. Workarounds, which affect the UE and are not generally supported by browsers, may cause interoperability problems. Other workarounds may impose restrictions on the attached application servers.

To access virtual hosts where different servers with different DNS names are co-located on AP, either of the solutions could be used to identify the host during the handshaking phase:

- TLS Extensions are specified in IETF, cf. Annex E of TS 33.310 [20]. TLS Extensions support the UE to indicate a virtual host that it intends to connect in the very initial TLS handshaking message (see clause 5.3.1);

- The other alternative is to issue a multiple-identities certificate for the AP. The certificate will contain identities of AP as well as each server that rely on AP's proxy function. The verification of this type of certificate is specified in RFC 2818 [9].

Either approach may be chosen by the operator who operates the authentication proxy.

Annex B (informative): Guidance on Certificate-based mutual authentication between UE and application server

This section explains how subscriber certificates (see TS 33.221 [16]) are used in certificate-based mutual authentication between a UE and an application server. The certificate-based mutual authentication between a UE and an application server shall be based on TLS as specified in the TLS and TLS Extensions profile given in TS 33.310 [20], Annex E.

When a UE and an application server (AS) want to mutually authenticate each other based on certificates, the UE has previously enrolled a subscriber certificate as specified in TS 33.221 [16]. After UE is in the possession of the subscriber certificate it may establish a TLS tunnel with the AS as specified in the TLS and TLS Extensions profile given in TS 33.310 [20], Annex E.

The AS may indicate to the UE, that it supports client certificate-based authentication by sending a CertificateRequest message during the TLS handshake. This message includes a list of certificate types and a list of acceptable certificate authorities. The AS may indicate to the UE that it supports subscriber certificate-based authentication if the list of acceptable certificate authorities includes the certification authority of the subscriber certificate (i.e. the operator's CA certificate).

The UE may continue with the subscriber certificate-based authentication if the list of acceptable certificate authorities includes the certification authority of the subscriber certificate. This is done by sending the subscriber certificate as the Certificate message during the TLS handshake. If the list of acceptable certificate authorities does not include the certification authority of the subscriber certificate, then UE shall send a Certificate message that does not contain any certificates.

NOTE 1: Due to the short lifetime of the subscriber certificate, the usage of the subscriber certificate does not require on-line interaction between the AS and the PKI portal that issued the certificate.

If the AS receives a Certificate message that does not contain any certificates, it can continue the TLS handshake in two ways:

- if subscriber certificate-based authentication is mandatory according to the AS's security policy, it shall response with a fatal handshake failure alert, or

- if subscriber certificate-based authentication is optional according to AS's security policy, AS shall continue with TLS handshake.

In the latter case, if the AS has NAF functionality, the NAF may authenticate the UE as specified in clause 5.3 of the present specification, where after establishing the server-authenticated TLS tunnel, the procedure continues from step 4.

NOTE 2: In order to successfully establish a TLS tunnel between the UE and the AS using certificates for mutual authentication, the UE must have the root certificate of the AS's certificate in the UE's certificate store, and the AS must have the root certificate of the UE's subscriber certificate (i.e. operator's CA certificate) in the AS's certificate store. The root certificate is the root of the certification path, and should be marked trusted in the UE and the AS.

NOTE 3: In order to enable access to an AS in a visited network with subscriber certificates requires that the AS has the CA certificate of subscriber's home operator and it is marked trusted in the visited AS. The procedure to do this is outside the scope of this specification.

Annex C (informative): Considerations for GBA security using a web browser and Javascript

C.1 Usage Scenario

End user wants to use some service provider"s services (e.g., an operator), and the service provider wants to use GBA to authenticate the user.

1) End user opens web browser application in the ME, and instructs it to go the service provider"s web page. The web page redirects the web browser to a login page if end user has not yet authenticated.

2) Service provider"s login page has logic to discover whether Javascript access to GBA is enabled in the browser or not (can be done with Javascript). If this GBA access is not supported, the web page reverts to other means of authentication, e.g., legacy username/password. If it is supported, proceed to step 3.

3) The web page has code implemented in Javascript that obtains a NAF specific token (Ks_js_NAF) and the B-TID from the GBA function in the UE. In simplest case, the browser uses these variables as username and password in an HTML FORM, and the web browser to send this information back to the web server.

4) The web server extracts the NAF specific token (Ks_js_NAF) and the B-TID, and uses the B-TID to fetch the NAF specific key Ks_NAF from the BSF over Zn interface. The NAF generates then the NAF specific authentication token and compares it with the one received from the UE. If they are equal, the user is authenticated, and the requested service is provided to the ME and the user.

C.2 Threats

The usage scenarios described in clause A.1 are susceptible to five serious threats:

Threat 1: ME downloads a web page from an attacker that has Javascript which requests all NAF specific keys that the attacker is interested in.

Threat 2: ME uses a public access point that is controlled by the attacker, i.e., a classic man-in-the-middle attack. When the ME requests the login page from the service provider, the attacker sends back a rogue login web page as it controls the DNS. This rogue login page has Javascript that is able to extract any NAF specific authentication token of the service provider, and sends it back to the attacker.

Threat 3: It is possible for any third party on the internet connection to eavesdrop on the B-TID and the NAF specific authentication token, and impersonate the user as long as the B-TID has not expired.

Threat 4: If an attacker gets hold of the authentication token Ks_js_NAF, then he can utilize it to attack the communication between theweb browser and the NAF.

Threat 5: ME downloads a web page from an attacker that has JavaScript which repeatedly triggers GBA re-bootstrapping to be performed. This can have the effect that the malicious web page can coordinate a distributed DoS attack against the BSF/HSS.

C.3 Control of GBA Credentials and GBA Module in the UE

C.3.1 General

The threats identified in clause C.2 are countered using a set of control mechanisms as defined in this clause. Using only a subset of the control mechanisms leaves some threats open. Therefore all control mechanisms need to be applied to mitigate the outlined threats.

C.3.2 Control Mechanism 1– Same Origin Authentication Tokens

To mitigate threat 1 in clause A.2, the web browser should limit a web page to access only to those NAF specific authentication tokens that belong to the origin web server. This way Javascript has access only to NAF"s authentication tokens, which is the NAF identified by the origin of the web page. This implies that the browser can authenticate the server, cf. control mechanism 2. Web browsers observing good security practices implement a single-origin policy where the Javascript is able to send HTTP requests only to the server from where the original web page came from.

C.3.3 Control Mechanism 2 – Server Authenticated TLS

To mitigate threats 1, 2 and 3 of C.2, HTTPS, i.e., server authenticated TLS, should be used with integrity and confidentiality protection. This way attacking DNS does not help the attacker as the origin of the web page is authenticated using TLS, and the web page content, and B-TID and Ks_js_NAF are confidentially protected against eavesdropping and the Ks_(ext)_NAF is not used directly here.

C.3.4 Control Mechanism 3 - Channel Binding

The usage of server authenticated TLS as described in clause C.3.3 is not sufficient on its own if one were to consider the threat of a compromised TLS server certificate a likely event. Given that in commonly used browsers there are 100+ root certificates from certification authorities (CAs) that have different levels of security protection when issuing and managing certificates, A second line of defense for the case that a TLS server certificate is compromised seems useful. If one CA is compromised the attacker can use a compromised certificate to lure the user into believing that the attacker"s server is the genuine NAF the user wants to communicate with. The attacker can exploit this to realize the following two threats:

- Threat A: If the Javascript used the Ks_NAF directly and an attacker obtained the Ks_NAF from the user, then it could use this Ks_NAF to impersonate the user towards the genuine NAF, obtain the services and let the user foot the bill.
- Threat B: The attacker makes the user reveal information valuable for the attacker that the user would want to reveal only to the genuine NAF.

Even though TLS with server certificates can generally be trusted, it imporves the security of usage of GBA from a web browser if the authentication token derivation process of GBA and the TLS tunnel are bound together. This shall however not be taken as a general clue that TLS with server side certificate authentication is insecure. As the key derivation of Ks_(ext)_NAF is already defined with a fixed set of input parameters, and backward compatibility by not changing this key derivation should be ensured, a new Javascript specific authentication token (Ks_js_NAF) should be derived from Ks_(ext)_NAF using a channel binding mechanism. This channel binding mechanism is based on RFC 5929 (Channel Bindings for TLS) [25].

This mechanism does not help against threat B. The mitigation of threat B is further discussed in clause C.4.

C.3.5 Control Mechanism 4 – Key Usage

In Threat 4 in clause C.2, the attacker may get hold of the Ks_js_NAF by one of the following means:

- One of the endpoints can be considered as compromised i.e. NAF or web browser are compromised.
- Ks_(ext)_NAF and authentication token derivation parameters are compromised.

The compromise of an endpoint might be made more difficult by usage of additional hardware functionalities, but those would require that all communication for usage of such keys would be routed over the secure hardware. This would still leave the challenge, how to ensure that no fake traffic is routed over the secure hardware. The handle used to authorize the usage of the Ks_js_NAF authentication token inside the secure module need to be secured to avoid unauthorized usage, but that would require a trustworthy browser, which then negates the effect of using a handle for authentication tokens. The usage of the Ks_js_NAF should be done in the TLS session that was used to create the token. This makes usage in another TLS session impossible, as long as the end points check that the TLS tunnel used to receive the information is the same as was used to derive the token.

If the compromised token has been derived, by usage of the compromised Ks_(ext)_NAF key and corresponding parameters, then usage of additional secure hardware would not gain any significant security improvement for the usage from the token of the originating terminal, since the source of the Ks_js_NAF token is compromised.

C.4 Security Considerations

C.4.1 General Scripting Security Considerations

JavaScript has been designed as an open scripting language, and it has its own security model. This model has not been designed to protect the server administrator or the data that is passed between the browser and the external application server. The scripting language security model is designed to protect the user from malicious servers, and as a result, capabilities of Javascript have been restricted. For example, there are deployed Javascript implementations that cannot read or write files on users' computers, or interact between different web pages that are open at the same time in the browser. W3C has been extending Javascript APIs to include functionality, including File API [26] enabling reading and writing files, and HTML5 Web Messaging enabling communication between the web pages in the browser.

C.4.2 GBA key control

When the Javascript specific authentication token (Ks_js_NAF) is requested by a web page, its creation is controlled by the web browser as specified in Annex D. The Ks_js_NAF is bound to the web server, to the javascript context, and to the specific of TLS tunnel used by using NAF_ID as described in Annex D. The Ks_js_NAF should not be used outside the designed web page context.

C.4.3 User grants

When Javascript in a web page is trying to access the Javascript specific authentication token via the Javascript GBA API, the browser executing the Javascript may prompt the end user with a permission dialog asking the end user to grant access to the token. The end user can then decide whether to allow access or deny it, and also additionally have the browser remember the decision. This mimics the functionality of the browsers today that support geolocation Javascript API. There Javascript notifies the end user, that the current page is requesting location information. The end user has then the possibility to either grant access or deny it. Additionally, the end user may have the browser remember that decision, so that the next time the same page is requesting access to the location information, the answer from the previous query from the end users is used without disturbing the end user.

C.4.4 Root CAs in Browser

Clause C.3.4 describes the threats related to a compromised CA where either the CA certificate itself or the certificate of some root CA above the compromised CA is present in browsers' root CA list. For threat B it is assumed that it is possible to issue certificates containing any DNS name, and therefore pretend to be any server. If the attacker can spoof https://www.popular-social-web-site.com or https://accounts.popular-mail-service.com for instance, he can easily trick users into entering their username and password to attacker's webpages by just mimicking the look-n-feel of the attacked webpages. Additionally, with the introduction of HTML5 there are additional things to consider as HTML5 introduces features like WebStorage API [27], where a web site can use "localStorage" function to store name-value pairs to the browser, which can be later accessed only by those web pages that have been downloaded from same server identified by protocol/site/port tuple. With this threat, the attacker can fully read from and write to the localStorage of the attacked site.

There is no way to mitigate this threat if a compromised CA is listed in browsers' root CA list except strongly recommend that the browser vendors should carefully consider which CAs they include to their browser offering as trust roots by default, and that the browser implementation should show proper warnings to the end user, if the user (or some service on behalf of the user) tries to add a new CA as trust root. In addition, root CA stores managed online by some external instance, e.g., browser vendors updating root CA stores of their browsers online, should also be kept up-to-date.

Annex D (normative) Security measures for usage of GBA with a web browser

D.1 Extension of Protocol Mechanism used on Ua Reference Point

D.1.1 General

The Annex D is compatible with the chapters and sections up to and including clause 5.2. The clauses 5.3, 5.4 and 5.5 can be seen as different alternatives, since the client authentication mechanism in Annex D is different from those in 5.3, 5.4 and 5.5.

D.1.2 Key derivation

FQDN

When the web browser in the ME downloads a web page using HTTPS, the web browser verifies that the FQDN in the URL matches the FQDN used in the TLS certificate used by the server (NAF). It is common good security practice for web browsers to perform this check. Any web browser that does not perform this check is not secure enough to be used for security sensitive applications with or without GBA, and therefore should not be considered for the purpose of this specificaton.

Once the web browser has verified that the FQDN in the URL matches the FQDN in the server (NAF) certificate, the browser makes this verified FQDN available to the GBA API.

The GBA API uses the verified FQDN to derive the authentication token Ks_js_NAF.

NOTE1: Security associated with the use of the FQDN in Javascript in the manner described above is dependent upon the implementation of the web browser, which is out-of-scope for 3GPP.

The Ua security protocol identifier to be used is (0x01,0x00,0x02,yy,zz) as specified in Annex H of 3GPP TS 33.220 [3], where yy and zz are the protection mechanism CipherSuite as specified in relevant TLS specifications by IETF.

NOTE2: HTML FORM is tunneled through TLS, therefore the first consideration might be to use the Ua security protocol identifier for Ua security protocols that are based on TLS (HTTP Digest with HTTPS and Pre-shared key TLS) that is already specified in Annex H of 3GPP TS 33.220 [3] (0x01,0x00,0x01,yy,zz). This protocol id is used when the NAF specific key is used as a password in the TLS tunneled HTTP Digest case. This is substantially different, for example, from the case where HTML FORM based authentication within TLS tunnel is used, therefore a different Ua protocol id is used.

D.1.3 Channel binding

D.1.3.1 Background

To mitigate the threats described in Annex C, a second level of key derivation is introduced. When Javascript code that is downloaded from the web server via the server authenticated TLS tunnel requests for a GBA based key, the request is first handled by the web browser and more specifically the GBA API module in the web browser. The GBA API module will request the Ks_(ext)_NAF key from the GBA Function in the ME using the Javascript specific NAF_ID as specified in clause D.1.2. After receiving the Ks_(ext)_NAF key from the GBA Function, the GBA API will derive a Javascript specific authentication token Ks_js_NAF that is bound to the server authenticated TLS tunnel.

The channel binding is performed using RFC 5705 [29] and RFC 5929 [25], as is described below. An example sequence flow is in clause D.2.1.

D.1.3.2 Channel binding using RFC 5705 and RFC 5929

After receiving the Ks_(ext)_NAF key from the GBA Function the GBA API obtains the TLS_MK_Extr, which is extracted from the TLS master key using the exporter function as specified in RFC 5705 [29]. The label for the exporter function shall be "TLS_MK_Extr". The GBA API obtains the tls-server-endpoint as specified in RFC 5929 [25]. The Ks_js_NAF shall be derived from Ks_(ext)_NAF as follows:

Ks_js_NAF = KDF (Ks_(ext)_NAF, TLS_MK_Extr, tls-server-endpoint)

A sequence flow is in clause D.2.1.

D.2 Sequence flow

D.2.1 Sequence flow with channel binding

In this message flow with channel binding the following architecture is assumed:

- **GBA Function:** The GBA Function handles establishment of GBA-specific keys. In particular, the establishment of the key Ks can use any of the methods defined by TS 33.220 [3] (e.g., based on AKA or GBA_Digest). The GBA Function is not part of the web browser.

  NOTE: In the case of GBA_Digest, the GBA Function treats SIP Digest credentials as specified in Annex N of TS 33.203 [28].

- **Web Browser:** The web browser is either native or downloaded and contains some functions which support usage of GBA. In particular we have in the architecture:

  - **GBA_API:** Part of the browser that communicates with the GBA Function and receives GBA authentication token material requests from the Javascript code.
  - **Javascript:** Downloaded Javascript code.
  - **Engine:** Sets up communication with the NAF.

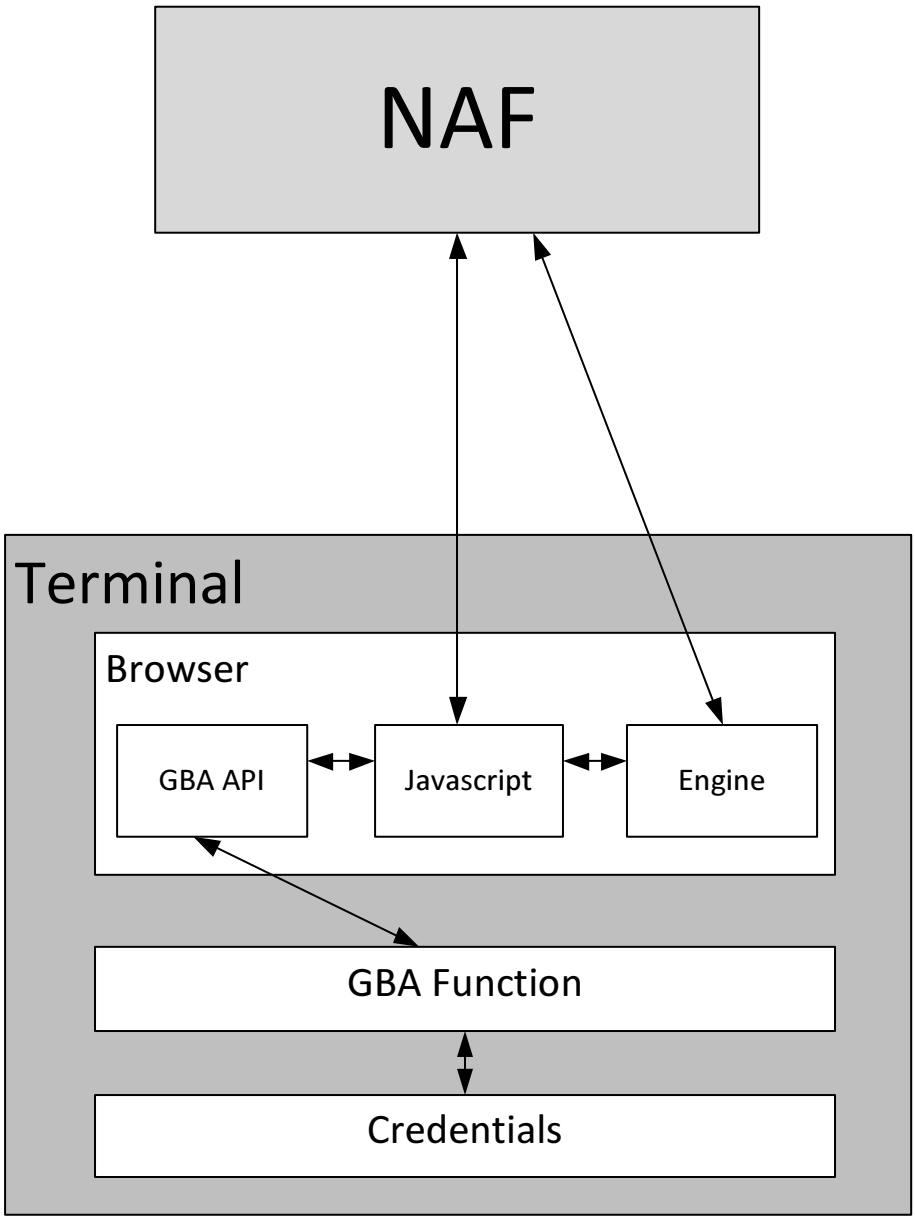

Figure D.2-1.Architecture

Below is a sequence flow diagram of GBA usage in Web context, i.e., within Javascript.

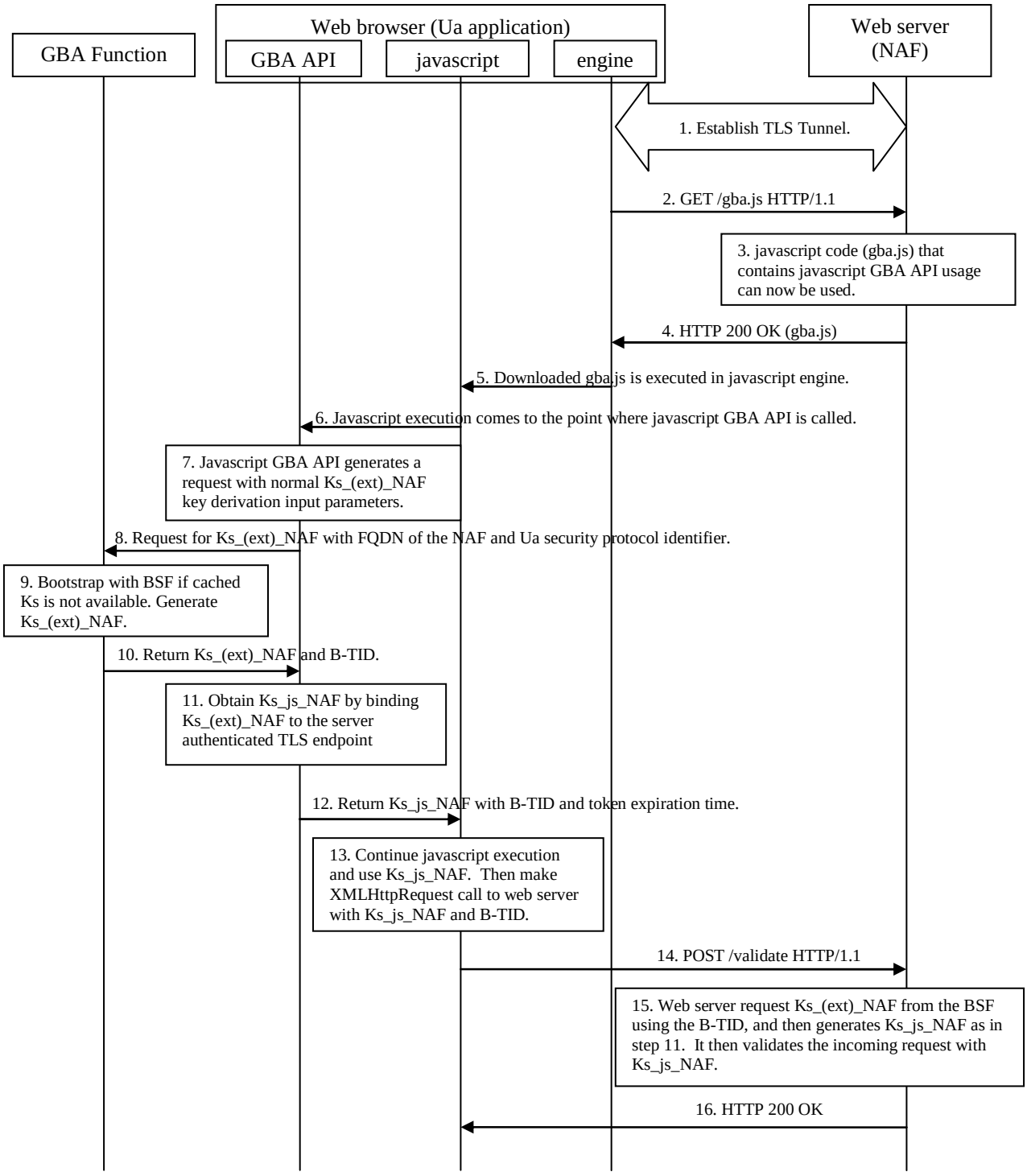

Figure D.2-2. Sequence flow.

The web browser is considered to be a trusted application in the sense that the user trusts it to handle security related functions properly, i.e., setting TLS sessions with servers, sandboxing the Javascript code that is downloaded from the web servers, and not leaking sensitive information like a password to third parties. In the sequence flow diagram, the web browser is divided into three functional blocks:

- engine module handles the basic functionalities for the web browser like setting up TLS with web servers, downloading web resources from network, and providing the user interface with the end user.

- GBA API module offers the API towards any Javascript executing in the web browser. As Javascript should not be explicitly trusted, the web browser and the GBA API should not reveal any sensitive information to the Javascript, nor should they accept any sensitive information from the Javascript more than necessary.

- Javascript module executes the downloaded Javascript. Any Javascript executing in web browser should be considered not trusted and should not be granted access to sensitive resources or the access to those resources should be controlled.

The communication between web browser and web server in the depicted sequence flow diagram is executed inside a server authenticated TLS tunnel. Also, the web browser is in the process of downloading an html page where one of the linked Javascript resources is "gba.js".

1) The web browser and the web server establish a server authenticated TLS session. The use of TLS message integrity is mandatory, while the use of TLS encryption is optional. All further messages between the web server and the UE shall be sent through this tunnel.

2) The web browser engine makes a HTTP GET request to the server to download gba.js resource from the server.

3) The web server now knows that it can use the gba.js file that contains the Javascript GBA API call on the browser. The gba.js can also contain additional logical elements that make use of the Javascript specific authentication token Ks_js_NAF.

   Editor"s note: The definition of the GBA API details needs to be done, ffs if this happens in W3C or 3GPP SA3, depending on communication with them.

   The GBA API is:

```
document.gba.getGBAToken(successCallback,
                         errorCallback);
```

4. As a HTTP response to the HTTP request made in step 2, the web server returns the gba.js to the web browser.

5. The engine in the web browser starts to execute the Javascript in gba.js in Javascript sandbox.

6. The Javascript comes to a point where a call to GBA API is made.

7. Browser's Javascript GBA API locates the relevant information about the Javascript, i.e., in what html page it is executing, from what url was the html page downloaded from, and which TLS ciphersuite is used in the TLS tunnel. The FQDN of the NAF can be extracted from the url of the web page, and the Ua security protocol identifier can be derived from the used TLS ciphersuite. FQDN of the NAF and the Ua security protocol identifier form the NAF_ID.

8. Browser's Javascript GBA API makes a call to ME's GBA Function with the NAF_ID derived in step 7.

9. The GBA Function bootstraps with the BSF if there is no valid GBA master key Ks. From the Ks, Ks_(ext)_NAF, the NAF specific key is derived using the NAF_ID.

10. The GBA Function returns the Ks_(ext)_NAF key to browser's Javascript GBA API with the bootstrapping transaction identifier (B-TID).

11. Upon receiving the Ks_(ext)_NAF key, browser's javascript GBA API will derive the Javascript specific authentication token Ks_js_NAF that is bound to the server authenticated TLS session as follows:

    The valuse of the bindingType in GBAOptions are "tls-key-extractor" (i.e. RFC 5705 is used with the label "TLS_MK_Extr") and tls-server-endpoint (i.e. RFC 5929 [25]). T he Ks_js_NAF is derived as:

    Ks_js_NAF = KDF (Ks_(ext)_NAF, TLS_MK_Extr, tls-server-endpoint)

    The tls-server-endpoint value and TLS_MK_Extr are all related to the TLS connection that established the TLS session in step 1.

12. Browser's Javascript GBA API returns Javascript specific Ks_js_NAF authentication token, B-TID and authentication token lifetime to the executing javascript.

13. The Javascript continues to execute and it uses the Ks_js_NAF authentication token the way the web server has instructed (via Javascript).

Javascript can extract parameters from result object in Javascript (continued from step 2).

```
function successCallback(result) {
   var token = result.token;
   var btid = result.btid;
   var lifetime = result.expiryTime;
}
```

14. After executing the client side logic, the Javascript makes a XMLHttpRequest (ajax call, HTTP request) to the web server. This request contains at least Ks_js_NAF or hash of it, and B-TID.

15. The web server fetches the Ks_(ext)_NAF key from the BSF, and it then derives the Ks_js_NAF the same way it was done in step 11. The web server will then compare the received Ks_js_NAF with the locally derived one and validate that the TLS session is the same as was used for the request that established the TLS session in step 1.

16. If the received Ks_js_NAF is valid, the web server will continue to process the request made in step 14 and return the result to the web browser (to the Javascript).

D.3 Javascript GBA API description

D.3.1 GBA API Description

Below is a description how Javascript based GBA API is specified:

```
[NoInterfaceObject]
interface DocumentGBA {
    readonly attribute GBA gba;
};

Document implements DocumentGBA;

[NoInterfaceObject]
interface GBA {
   void getGBAToken(in GBACallback successCallback,
                    in optional GBAErrorCallback errorCallback,
                    in optional GBAOptions options);
};

[Callback=FunctionOnly, NoInterfaceObject]
interface GBACallback {
   void handleEvent(in GBATokenInfo keyinfo);
};

[Callback=FunctionOnly, NoInterfaceObject]
interface GBAErrorCallback {
   void handleEvent(in GBAError error);
};

[Callback, NoInterfaceObject]
interface GBAOptions {
   attribute boolean forceBootstrap;          // force bootstrapping; default false
   attribute DOMString bindingTypeExtr;       // TLS channel binding tls-key-extractor;
                                                                    attribute DOMStrong
bindingTypeEndPoint    // TLS channel binding tls-server-endpoint

};

// The NAF_ID is determined by the web browser. The FQDN is taken from the origin URL
// of the web page that has the javascript.  The Ua security protocol identifier is
// (0x01,0x00,0x02,yy,zz) where the yy,zz is CipherSuite in the used TLS tunnel (HTTPS).
```

```
// If TLS tunnel was not used, (0xFF, 0xFF, 0xFF, 0xFF, 0xFF) is used as Ua security
// protocol identifier. The latter case is not specified in 3GPP and it should only be
// used for testing purposes.

interface GBATokenInfo {
   readonly attribute DOMString key;          // base64 encoded GBA key: Ks_(ext)_NAF
   readonly attribute DOMString btid;         // B-TID
   readonly attribute long bootstrapTime;     // Bootstrap time; millisecs since 1.1.1970
   readonly attribute long expiryTime;        // Token expiry: millisecs since 1.1.1970
   readonly attribute DOMString fqdn;         // used FQDN
   readonly attribute DOMString uaSecProtId;  // base64 encoded Ua security prot. id;
};

interface GBAError {
   readonly attribute unsigned short code;    // error code (to be specified)
   readonly attribute DOMString message;      // textual description of the error
};
```

D.3.2 API usage

Below is a description how to use javascript based GBA API:

```
// Basic example of requesting GBA token
document.gba.getGBAToken(gbaSuccess,gbaError);

function gbaSuccess(tokeninfo) {
   // gba token was successfully created, and for example use
   // tokeninfo.btid as username and tokeninfo.token as password
}

function gbaError(error) {
   // an error occured during gba token creation
}
```

Annex -E (informative): Change history

Change history									
Date	TSG #	TSG Doc.	CR	Rev	Cat	Subject/Comment	Old	New	WI
05-2004	SP-24	SP-040368	-	-		Presentation to TSG SA for Approval	1.1.1	2.0.0	
06-2004	SP-24	-	-	-		Approved at TSG SA #24 (Release 6)	2.0.0	6.0.0	
09-2004	SP-25	SP-040621	0001	-		GBA User Security Settings	6.0.0	6.1.0	SEC1-SC
09-2004	SP-25	SP-040621	0002	-		GBA supported indication and NAF hostname transfer in HTTP and in PSK TLS	6.0.0	6.1.0	SEC1-SC
09-2004	SP-25	SP-040621	0003	-		Editorial clean-up of TS 33.222	6.0.0	6.1.0	
09-2004	SP-25	SP-040621	0004	-		Further modifications to TLS profile related text in 33.222	6.0.0	6.1.0	SEC1-SC
12-2004	SP-26	SP-040889	0005	-		GBA supported indication in PSK TLS	6.1.0	6.2.0	GBA-SSC
12-2004	SP-26	SP-040889	0007	1		Adding Support for AES in the TLS Profile	6.1.0	6.2.0	GBA-SSC
12-2004	SP-26	SP-040889	0010	1		Authorization flag transfer between AP and AS	6.1.0	6.2.0	GBA-SSC
12-2004	SP-26	SP-040889	0012	-		Correction of inconsistencies within AP specification	6.1.0	6.2.0	GBA-SSC
12-2004	SP-26	SP-040889	0013	1		TLS extensions support	6.1.0	6.2.0	SEC1-SC
12-2004	SP-26	SP-040889	0014	-		Visited AS using subscriber certificates	6.1.0	6.2.0	SEC1-SC
03-2005	SP-27	SP-050141	0015	3		Keeping PSK TLS in 3GPP Rel-6	6.2.0	6.3.0	SEC1-SC
03-2005	SP-27	SP-050141	0016	1		Clarification to TS 33.222	6.2.0	6.3.0	SEC1-SC
03-2005	SP-27	SP-050166	0017	3		Clarify the GBA requirements for https supporting applications at Ua reference point	6.2.0	6.3.0	GBA-SSC
2005-06	SP-28	SP-050264	0019	-	F	Removal of editor"s note	6.3.0	6.4.0	SEC1-SC
2005-09	SP-29	SP-050558	0020	-	F	Adding additional mandatory CipherSuites for PSK TLS	6.4.0	6.5.0	SEC1-SC
2005-09	SP-29	SP-050568	0021	-	F	Removing an inconsistence within TS 33.222 (Section 6.2)	6.4.0	6.5.0	GBA-SSC
2005-09	SP-29	SP-050568	0023	-	F	Adding a clarification to TS 33.222 (Section 6.1)	6.4.0	6.5.0	GBA-SSC
2005-09	SP-29	SP-050575	0022	-	B	Usage of Ks_int_NAF for HTTPS connection between a UICC and a NAF	6.4.0	7.0.0	GBA-SSC
2006-03	SP-31	SP-060054	0025	-	A	Update PSK TLS Reference	7.0.0	7.1.0	SEC7-GAA2 (GAAExt)
2006-09	SP-33	SP-060501	0026	-	F	Clarification of using HTTP digest with HTTPS	7.1.0	7.2.0	SEC1-SC
2007-12	SP-38	SP-070787	0027	-	F	Usage of OMA References – Update of References	7.2.0	7.3.0	GAA2
2007-12	SP-38	SP-070793	0028	-	F	Certificate based mutual authentication: TLS profile	7.2.0	7.3.0	GAA2
2008-06	SP-40	SP-080265	0029	-	F	Clarification of usage of NULL encyption and TLS	7.3.0	8.0.0	TEI8
2009-12	-	-	-	-	-	Update to Rel-9 version (MCC)	8.0.0	9.0.0	-
2010-06	SP-48	SP-100361	0030	1	F	Deprecation of SHA-1	9.0.0	9.1.0	TEI9
2010-10	SP-49	SP-110482	0031	1	C	Unification of TLS and certificate references in TS 33.222 with TS 33.310	9.1.0	10.0.0	TEI10
2011-12	SP-54	--	--	--	--	Editorial correction of change history	10.0.0	10.0.1	--
2012-03	SP-55	SP-120039	0032	1	F	Correction of Reference	10.0.1	11.0.0	Sec11
			0033	-	F	Clarification of Mandatory Implementation			
			0034	-	F	Update of TLS extensions version			
2012-06	SP-56	SP-120341	0035	-	F	Correction of TLS Extensions References to point toTS 33.310	11.0.0	11.1.0	SEC11
2012-09	SP-57	SP-120604	0036	-	F	Taking into account GBA_Digest for HTTPS-based Ua interface	11.1.0	11.2.0	GBA-ext

2012-09	SP-57	SP-120605	0037	-	F	Correction of psk-TLS references and NULL cipher profiles to point toTS 33.310	11.1.0	11.2.0	SEC11
2012-10						Correction of History Table	11.2.0	11.2.1	
2012-10						Editorial corrections	11.2.1	11.2.2	
2012-12	SP-58	SP-120856	0038	1	D	Clarification of the term application specific user security settings	11.2.2	12.0.0	SEC12
			0039	1	B	Addition of definitions, abbreviations and references for web GBA	12.0.0	12.1.0	Web_GBA
2013-03	SP-59	SP-130035	0040	2	B	Addition of informative Annex on usage scenarios, threats and control of GBA credentials to guide developers			
			0041	2	B	Addition to the new informative Annex on security considerations			
			0042	1	B	Addition of new normative Annex on channel binding, key derivation and message flow			
			0044	1	C	Correction of Ua security protocol identifier text			
			0045	1	F	Missing reference added			
2013-06	SP-60	SP-130301	0046	-	C	Clarification on usage of confidentiality and integrity for TLS tunnel	12.1.0	12.2.0	Web_GBA
			0047	-	F	Removal of editor's note			
			0048	1	C	Combination of Channel Binding Methods			
			0049	1	B	Addition of details on API			
2013-12	SP-62	SP-130684	0051	1	A	Clarification of NAF key negotiation on Ua	12.2.0	12.3.0	GBA-ext

History

Document history		
V12.3.0	October 2014	Publication