3GPP TR 33.914 V2.0.0 (2012-02)

Technical Report

3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Single Sign On Application Security for Common IMS – based on SIP Digest (Release 11)

Keywords
IMS, Security

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 Report 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

The present document targets to describe the re-usage of non-UICC credentials, in particular SIP Digest credentials, to provide security for the access to applications.

The process of providing security in a certain context (application) based on security already defined in some other context (e.g. 3GPP network access, IMS) is often called bootstrapping of security. Bootstrapping enables Single Sign-On (SSO) to applications using the security infrastructure already present for e.g. 3GPP network access or IMS.

The Generic Bootstrapping Architecture (GBA), as defined in 3GPP TS 33.220 [2], provides a bootstrapping mechanism, but it is limited to UICC-based credentials. This means that other types of credentials, e.g. credentials used for access to the Common IMS, cannot benefit from GBA to provide security for the access to applications based on the security for network access or IMS. 3GPP TS 33.203 [4] defines, in particular, SIP Digest as an authentication mechanism for access to the Common IMS core over a non-3GPP access network, such as e.g. TISPAN NASS, or BBF, or cable access, or 3GPP2 access, or WiMAX access. The credentials used with SIP Digest are shared secrets, or passwords, stored in the HSS and in the terminal, or held by the user. By means of bootstrapping, GBA enables single sign-on to applications using the security infrastructure already present for 3GPP network access or IMS. As an example, GBA may be used for providing the security for the Ut interface used for self-administration of IMS subscribers, cf. TS 33.141 [4].

This Technical Report takes into consideration the benefits of SSO to applications and the provision of cryptographic keys to terminals and application servers, bootstrapped from IMS credentials that are available in those scenarios where non-UICC based authentication mechanisms, in particular SIP Digest, are used. SIP Digest is arguably the most commonly used authentication mechanism in current IMS deployments. As an example, an automated way for providing the security for the Ut interface, used for self-administration of IMS subscribers, would be for the benefit of subscribers using SIP Digest credentials.

The re-use of SIP Digest credentials for SSO to applications would bring the benefit, that there is no need for rolling out a separate security infrastructure for these applications. In this way, a SSO mechanism re-using SIP Digest credentials would ease the introduction of new applications and services for the operator whose subscribers use SIP Digest credentials in Common IMS.

Users would benefit from SSO as it reduces complexity for users when accessing applications. Furthermore, operators could provide a chargeable service to application providers. Charging users for the use applications could be tied to the IMS subscription, although this is a matter for further discussion.

A similar need for the re-use of SIP Digest credentials for applications has been recognized by ETSI TISPAN.

1 Scope

The objective of this study item is to provide reference material for IMS based non-UICC based Single Sign On (SSO) to applications. This study item targets to re-use the SIP Digest Credentials for SSO to applications by re-using Common IMS and existing security elements. The study should describe needed extension to enable a re-use of SIP Digest credentials in Common IMS for providing security between a terminal and an application server. The study aims to maximise the commonalities of the SSO_APS with the currently defined application security approaches in 3GPP while efficiently satisfying the needs of Common IMS deployments using SIP Digest.

The Technical Report targets to bring forth approaches with a security level for access to applications using SSO_APS that is at least as good as that provided by SIP Digest for Common IMS. This Technical Report is intended to be used where the usage of UICC is not possible in a UICC-less environment. If the usage of UICC is possible, then it is expected to used, but that is outside the scope of the present study.

The scope of this Technical Report (Study Item Code SSO_APS) is restricted to environments where the storage of credentials on a UICC is not mandated.

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 TR 21.905: "Vocabulary for 3GPP Specifications".

[2] 3GPP TS 33.220: "Generic Authentication Architecture (GAA); Generic Bootstrapping Architecture".

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

[4] 3GPP TS 33.141: "Presence Services, Security".

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

[6] 3GPP TS 33.222: "Generic Authentication Architecture (GAA); Access to network application functions using Hypertext Transfer Protocol over Transport Layer Security (HTTPS)"

[7] 3GPP TS 24.623: "Extensible Markup Language (XML) Configuration Access Protocol (XCAP) over the Ut interface for Manipulating Supplementary Services"

[8] 3GPP TR 33.980: "Interworking of Liberty Alliance Identity Federation Framework (ID-FF), Identity Web Service Framework (ID-WSF) and the Generic Authentication Architecture (GAA)".

[9] 3GPP TR 33.924: "Identity management and 3GPP security interworking; Identity management and Generic Authentication Architecture (GAA) interworking"

[10] 3GPP TS 24.229: "IP multimedia call control protocol based on Session Initiation Protocol (SIP) and Session Description Protocol (SDP); Stage 3"

[11] 3GPP TS 29.109: "Generic Authentication Architecture (GAA); Zh and Zn Interfaces based on the Diameter protocol; Stage 3".

[12] 3GPP TS 24.109: "Bootstrapping interface (Ub) and network application function interface (Ua); Protocol details"

[13] IETF, RFC 3261: "SIP: Session Initiation Protocol"

[14] OpenID Foundation "OpenID Authentication 2.0", http://openid.net/.

[15] IETF, RFC 5705: "Keying Material Exporter for Transport Layer Security (TLS)"

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

[17] N. Asokan, V. Niemi, K. Nyberg: http://dblp.uni-trier.de/rec/bibtex/conf/spw/Asokan03

3 Definitions, symbols and abbreviations

3.1 Definitions

For the purposes of the present document, the terms and definitions given in TR 21.905 [1] 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 [1].

The definitions of Relaying Party (RP), OpenID Provider (OP) and Identity Provider (IdP) can be found in TR 33.924 [9].

3.2 Abbreviations

For the purposes of the present document, the abbreviations given in TR 21.905 [1] and the following apply. An abbreviation defined in the present document takes precedence over the definition of the same abbreviation, if any, in TR 21.905 [1].

AA	Authenticating Agent
AKA	Authentication and Key Agreement Protocol
AS	Application Server
BA	Browsing Agent
BSF	Bootstrapping Server Function
HSS	Home Subscriber Server
IdP	Identity Provider
IMPI	IP Multimedia Private Identity
IMS	IP Multimedia Subsystem
GAA	Generic Authentication Architecture
GBA	Generic Bootstrapping Architecture
GUSS	GBA User Security Settings
NAF	Network Application Function
OP	OpenID Provider
RP	Relaying Party
S-CSCF	Serving Call State Control Function
SD-AV	SIP Digest Authentication Vector
SSO	Single Sign On
SIP	Session Initiation Protocol
SLF	Subscriber Locator Function
UE	User Equipment

4 Description of SSO Feature

Single Sign On (SSO) is a feature of an access control system for a range of independent systems, which are affiliated. The systems often are application services. This feature allows that the authentication process takes place once, and the user gains access to all affiliated systems without the need to authenticate again. The SSO subsystem provides the initial authentication and provides authentication information to the Application Server which is part of the SSO subsystem.

The SSO feature in this report is meant to

- support the re-use of SIP Digest credentials as specified in TS 33.203, Annex N for initial authentication to the SSO subsystem for terminals that are not equipped with a UICC.

NOTE: The S-CSCF needs to be involved when SIP digest is performed in IMS as specified in TS 33.203, Annex N. However, the client need not even to be registered in IMS when SIP Digest credentials are re-used in the SSO subsystem. This implies in particular that the S-CSCF is not involved when SIP digest credentials are re-used in the SSO subsystem.

- support interworking and exploit commonalities with existing SSO subsystem deployments e.g. OpenID, GBA, Liberty Alliance.

5 System Architecture and Assumptions

5.1 Overview of Existing Systems

5.1.1 Use of SIP Digest in Common IMS

SIP Digest is specified as an authentication method for Common IMS in TS 33.203 [3], Annex N. SIP Digest is allowed to be used under conditions specified there. SIP Digest authentication and the requirements from TS 33.203 Annex N, do not apply to access networks defined in 3GPP specifications.

The use of SIP Digest in Common IMS is a 3GPP-defined profile of HTTP Digest as defined in RFC 2617 [5].

The IMS entities actively involved in providing SIP Digest in IMS are: the UE, the S-CSCF, and the HSS. The UE plays the role of the HTTP client. The user name from RFC 2617 corresponds to the IMPI in IMS. The S-CSCF and the HSS combined play the role of the HTTP server in RFC 2617. As the interplay of S-CSCF and HSS is not a standard feature of HTTP Digest as used outside the IMS world we quote the crucial text from 3GPP TS 33.203, Annex N.2 here:

"Upon receiving the SIP REGISTER the S-CSCF shall use a SIP Digest Authentication Vector (SD-AV) for authenticating the user. If the S-CSCF has no valid SD-AV for the specific IMPI, then the S-CSCF shall send a request for SD-AV(s) to the HSS in CM1 where the number m of SD-AVs wanted is equal to 1. " and

"Upon receipt of a request from the S-CSCF, the HSS sends one SD-AV to the S-CSCF using CM2. The SD-AV consists of the qop (quality of protection) value, the authentication algorithm, realm, and a hash, called H(A1), of the IMPI, realm, and password. Refer to RFC 2617 [12] for additional information on the values in the authentication vector for SIP Digest based authentication. The qop value shall be set to "auth" since SIP Digest, as used in IMS, can only provide authentication, not message integrity." and

"The S-CSCF generates a random nonce, stores H(A1) and the nonce against the IMPI, and then sends a SIP 401 Auth_Challenge i.e., an authentication challenge towards the UE including the nonce in SM4. It also includes the realm, qop and algorithm parameters. RFC 2617 [12] specifies how to populate the parameters of a 401 Auth_Challenge."

According to TS 24.229 [10], the "realm" header field parameter is set to the domain name of the home network.

As a general feature of the IMS security architecture, SIP-based access to IMS application servers attached to the S-CSCF via the Isc interface does not require a separate authentication in addition to the authentication performed during IMS (re-)registration. This applies to all authentication mechanisms supported in Common IMS, and thus consequently applies also when SIP Digest is used as an authentication mechanism.

5.1.2 Uses of GBA

The GBA baseline architecture is described in TS 33.220. It can be considered a Single Sign-On system in the sense that it re-uses credentials already available on the user and the operator side for accessing the IMS using IMS AKA for authentication (with the ISIM as the credential) or for accessing a 3GPP-defined radio network using GSM AKA, UMTS AKA, or EPS AKA for authentication (with the SIM or USIM as the credential).

Whether and how the GBA extensions specified in TSs 33.221 [16] and 33.222 [6] could be applied when an UE is authenticated using HTTP Digest is assessed in the evaluation sections of the solutions in clause 7.

In Figure 5.1-1 only the relation of GBA with IMS is shown. The relation with 3GPP-defined radio network is analogous.

It is important to note that GBA depends on IMS and 3GPP-defined radio networks only in that GBA re-uses the corresponding credentials. The UE is not required to be registered in IMS or the 3GPP-defined radio network while employing GBA and hence does not rely on transport in IMS or transport provided by 3GPP-defined radio networks.

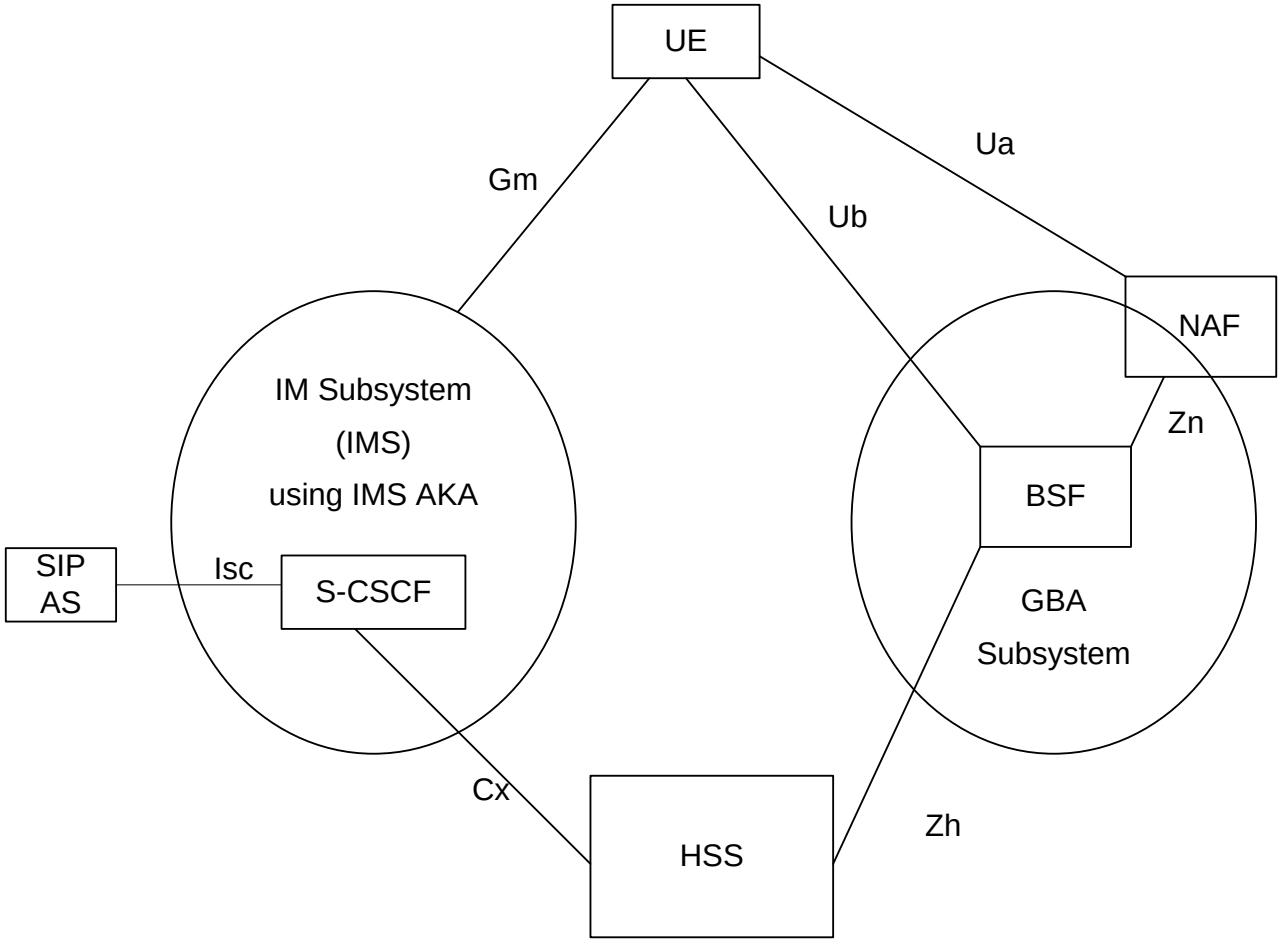


Figure 5.1-1: Relation of GBA with the Common IMS

3GPP has also described how GBA can interwork with non-3GPP-defined SSO systems, namely with Liberty Alliance systems in TR 33.980 and with OpenID in TR in 33.924 [9]. Here, only the architectural principles of the interworking between GBA and Liberty Alliance are presented. Interworking between GBA and OpenID follows similar principles.

From a high-level architectural point of view, basically two different approaches are described in TR 33.980:

- The Identity Provider (IdP) of Liberty Alliance is realized as a NAF in the sense of GBA;
- The Identity Provider (IdP) of Liberty Alliance is realized as a BSF in the sense of GBA. In this case, the full GBA baseline architecture is not used.

The following two figures 5.1-2 and 5.1-3 depict these two different approaches described in TR 33.980.

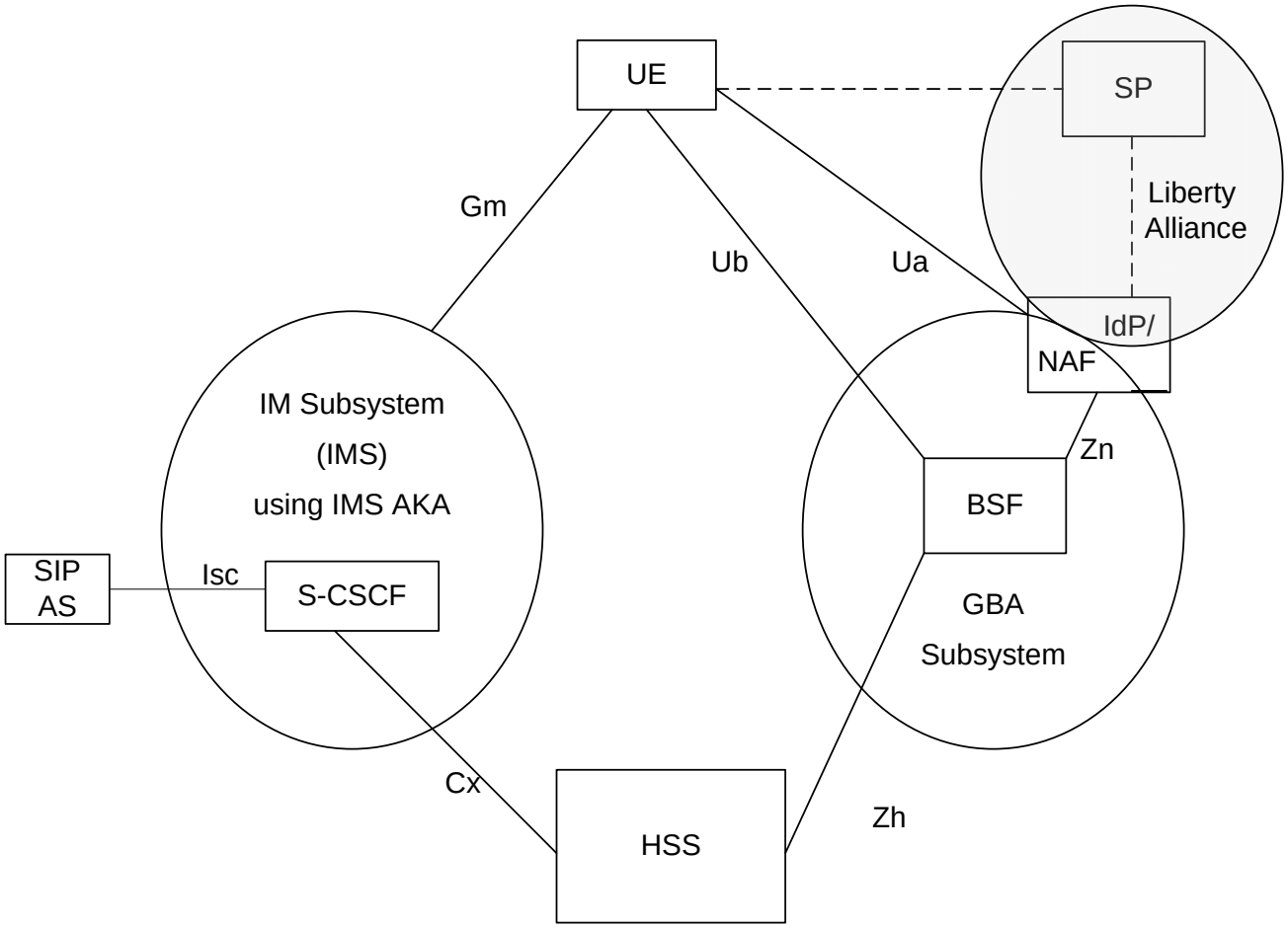


Figure 5.1-2: Relation of GBA with the Common IMS and interworking with Liberty Alliance collocating IdP and NAF

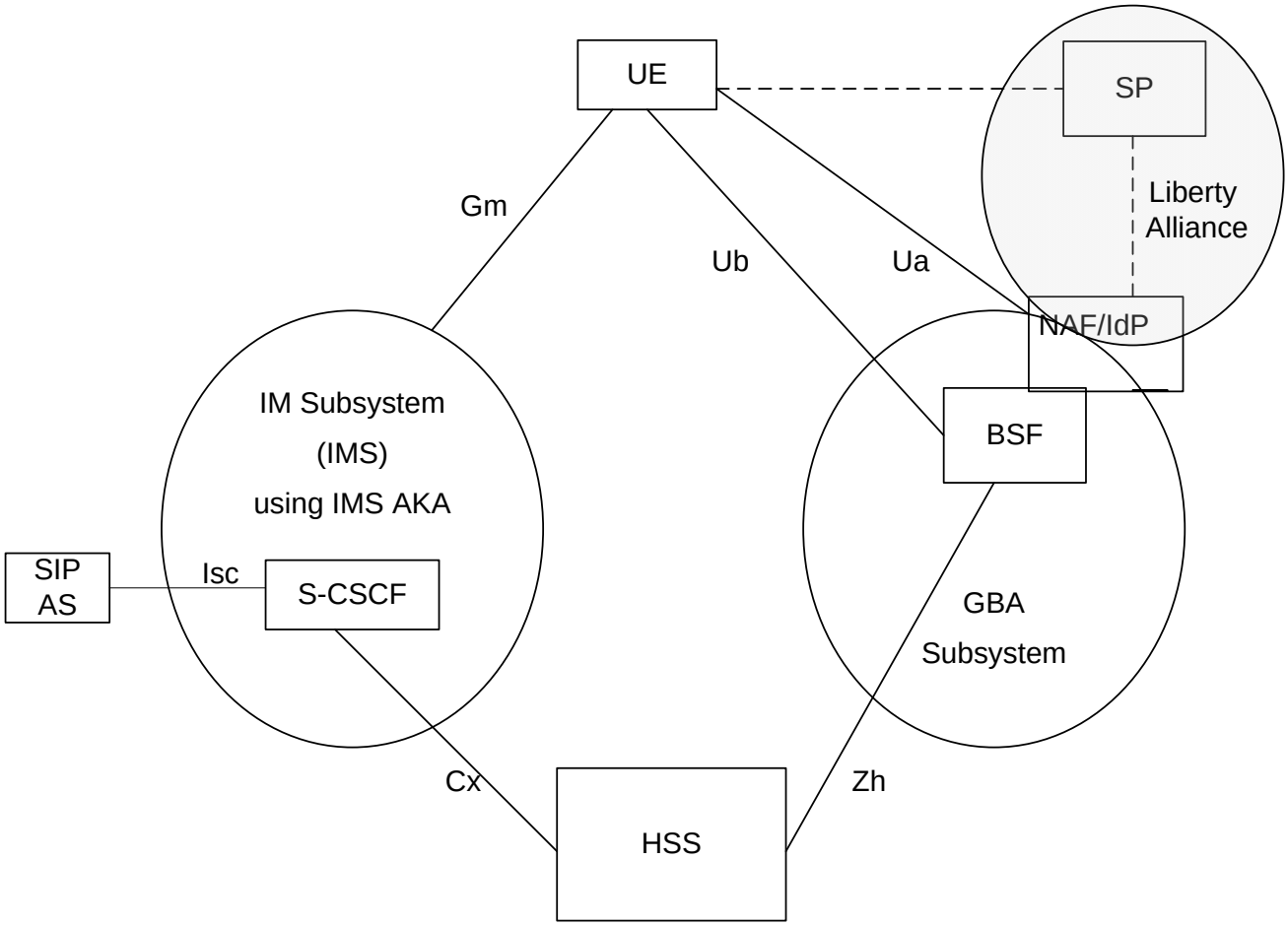


Figure 5.1-3: Relation of GBA with the Common IMS and interworking with Liberty Alliance collocating IdP and BSF

From a high-level architectural point of view, basically two different approaches are described in TR 33.924:

- The authenticating agent (AA) and the browsing agent (BA) reside in the same device;
- The AA and the BA reside in different devices (split terminal scenario), in that case some variants may occur depending on the availability of a local link and on the triggering of the authentication procedure.

The following two figures 5.1-4 and 5.1-5 depict these two different approaches described in TR 33.924.

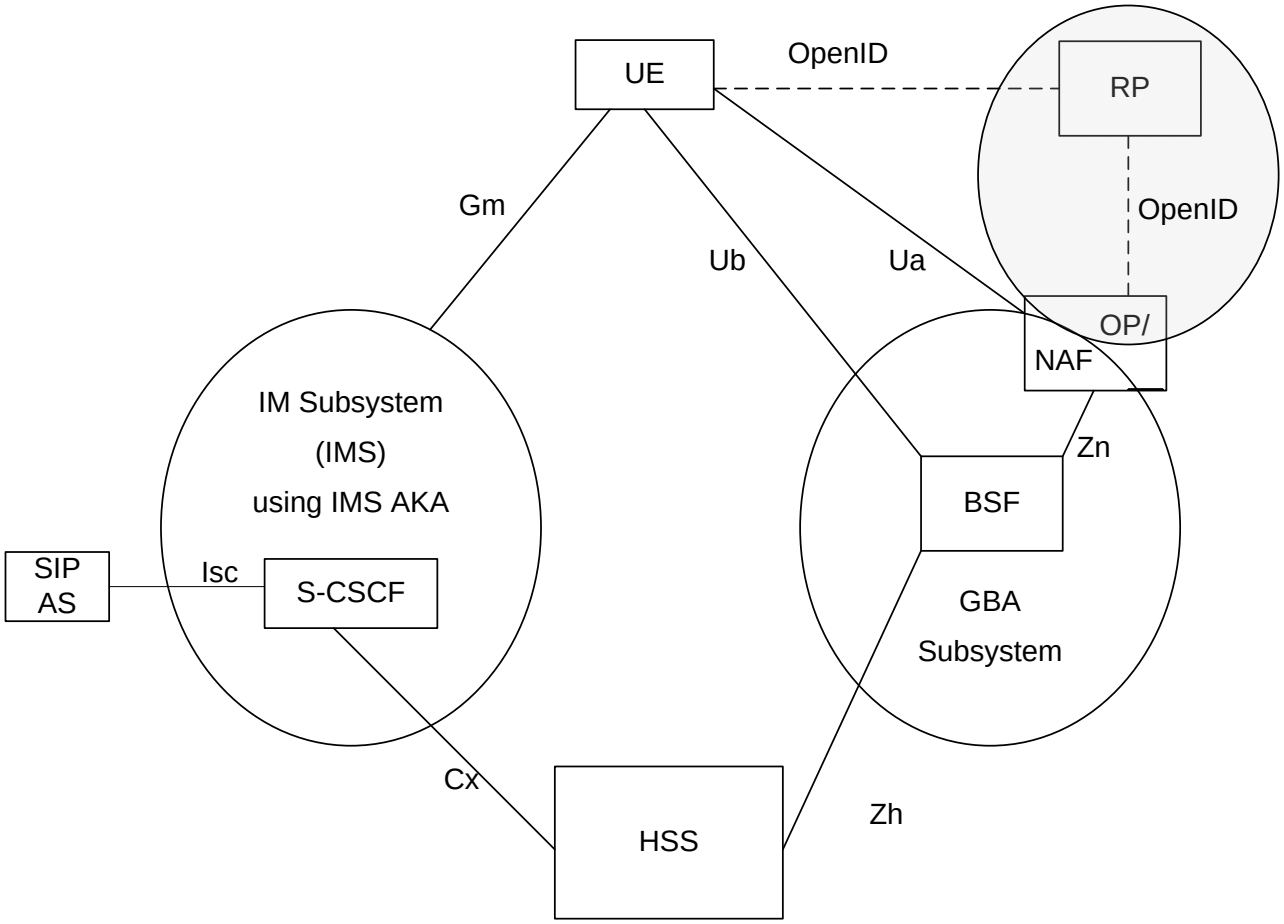


Figure 5.1-4: Relation of GBA with the Common IMS and interworking with OpenID

5.2 High-level architecture for SSO to applications for Common IMS based on SIP Digest

The following figure shows the current use of SIP Digest in IMS on the left hand side. On the right hand side, an application server (AS) is shown that is connected to an SSO subsystem, which in turn is connected to the HSS.

HTTP is certainly the most commonly used protocol with application servers or service providers. The study of other such protocols is not precluded, however.

The SSO subsystem is to contain one or more functional elements that are required to provide the SSO service to application servers based on SIP Digest. It is one of the main tasks of this study to define this SSO subsystem, i.e. to define the functional elements in the SSO subsystem, and their interworking with each other and with the application server and the HSS.

It is clearly shown in the figure that the HSS is the linking element between the IMS subsystem and the SSO subsystem, which results from the fact that HSS stores the permanent credentials used in both subsystems.

The right hand side shows three new reference points linking the UE and the HSS to the SSO subsystem, as well as linking the UE and the AS. The reference points SSOh and SSOb are required to provide the SSO functionality. The reference point SSOa is SSO system specific and may be out of scope of 3GPP.

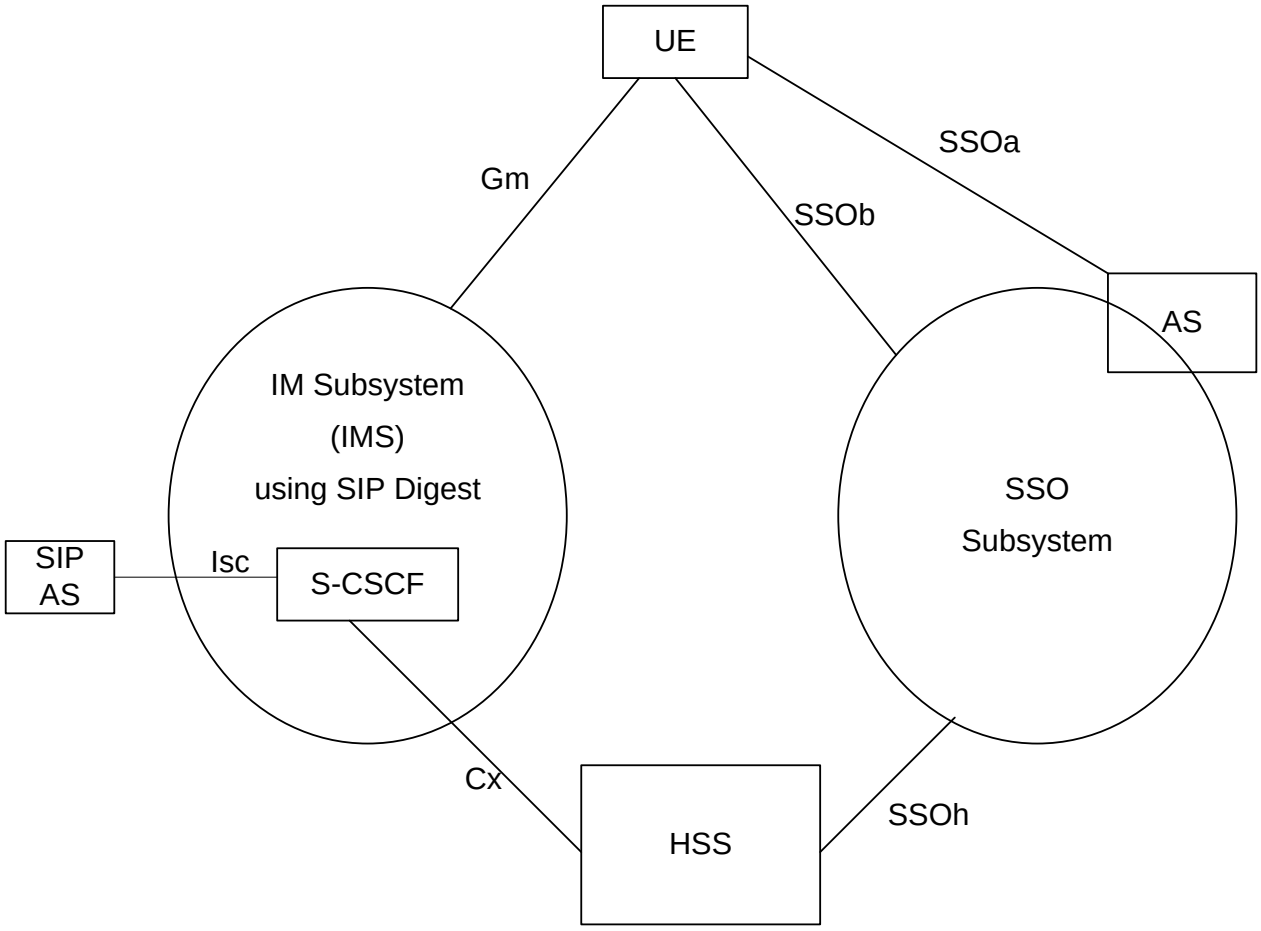


Figure 5.2-1: Relation of SIP Digest-based SSO with the Common IMS

5.3 Support for the Ut reference point

The Ut reference point is between a UE and an application server. TS 33.141 [4] defines the security for the Ut reference point, basically by referring back to TS 33.222 [6]. And TS 33.222 uses GBA, as defined in TS 33.220 [2].

TS 24.623 [7] on "Extensible Markup Language (XML) Configuration Access Protocol (XCAP) over the Ut interface for Manipulating Supplementary Services" defines an additional security mechanism for the Ut reference point, namely HTTP Digest according to RFC 2617 as an alternative to the use of GBA.

By the very definition of GBA, a form of single sign-on to the application server over Ut is provided as the SIM/USIM/ISIM credentials are re-used in GBA. The application server takes the role of a NAF in this case.

No such single sign-on is available when an UE is authenticated over Ut using HTTP Digest according to TS 24.623.

Whether and how single sign-on to an AS over the Ut reference point should be provided when an UE is authenticated using HTTP Digest is assessed in the evaluation sections of the solutions in clause 7.

Note that the user is able to manipulate his data on the application server even when the user is not registered in IMS as the use of GBA with an ISIM does not require the user to be registered in IMS, cf. clause 5.1.

There are two cases to consider, with and without an authentication proxy, as shown in the following two figures taken from TS 24.623.

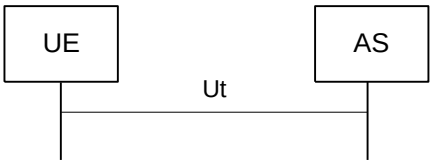


Figure 5.3-1: Ut interface

Quote from TS 24.623: "Authentication of the user with HTTP may take place directly at the AS, such as in figure 1, or with the support of an Authentication Proxy, such as in figure 2. The architecture for authentication is provided in 3GPP TS 33.222 [6].

NOTE: The Network Application Function (NAF) can be an AS."

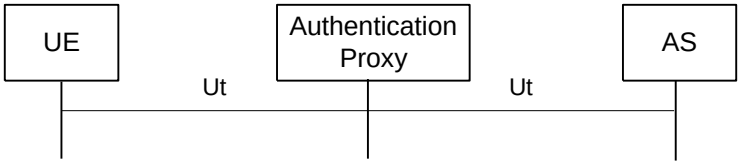


Figure 5.3-2: Authentication proxy in the Ut interface path

5.4 Interworking with Liberty Alliance

The SSO subsystem under study is meant to provide some form of interworking with, or support for, other SSO systems, notably Liberty Alliance, cf. this clause, and OpenID, cf. clause 5.5.

Interworking of GBA with Liberty Alliance systems is described in TR 33.980 [8]. It may be useful to take the approaches in TR 33.980 into account when studying the interworking of the SIP Digest-based SSO subsystem defined in this TR with Liberty Alliance, cf. clause 5.1.2. The results of the study shall not be bound by the approaches taken in TR 33.980 in any way, however.

A high-level architecture for interworking of SIP Digest-based SSO with Liberty Alliance that takes into account the baseline architecture from clause 5.2 is shown in Figure 5.4-1. The interfaces between the UE and the SP and the SP and the IdP are out of scope of 3GPP.

It should be noted that the two different approaches in TR 33.980 for interworking between GBA and Liberty Alliance, cf. clause 5.1.2, map to the same high-level architecture for the SIP-Digest based SSO case at this level of granularity. This is so because the SIP-Digest based SSO subsystem shows no substructure at this level of granularity with roles equivalent to BSF and NAF.

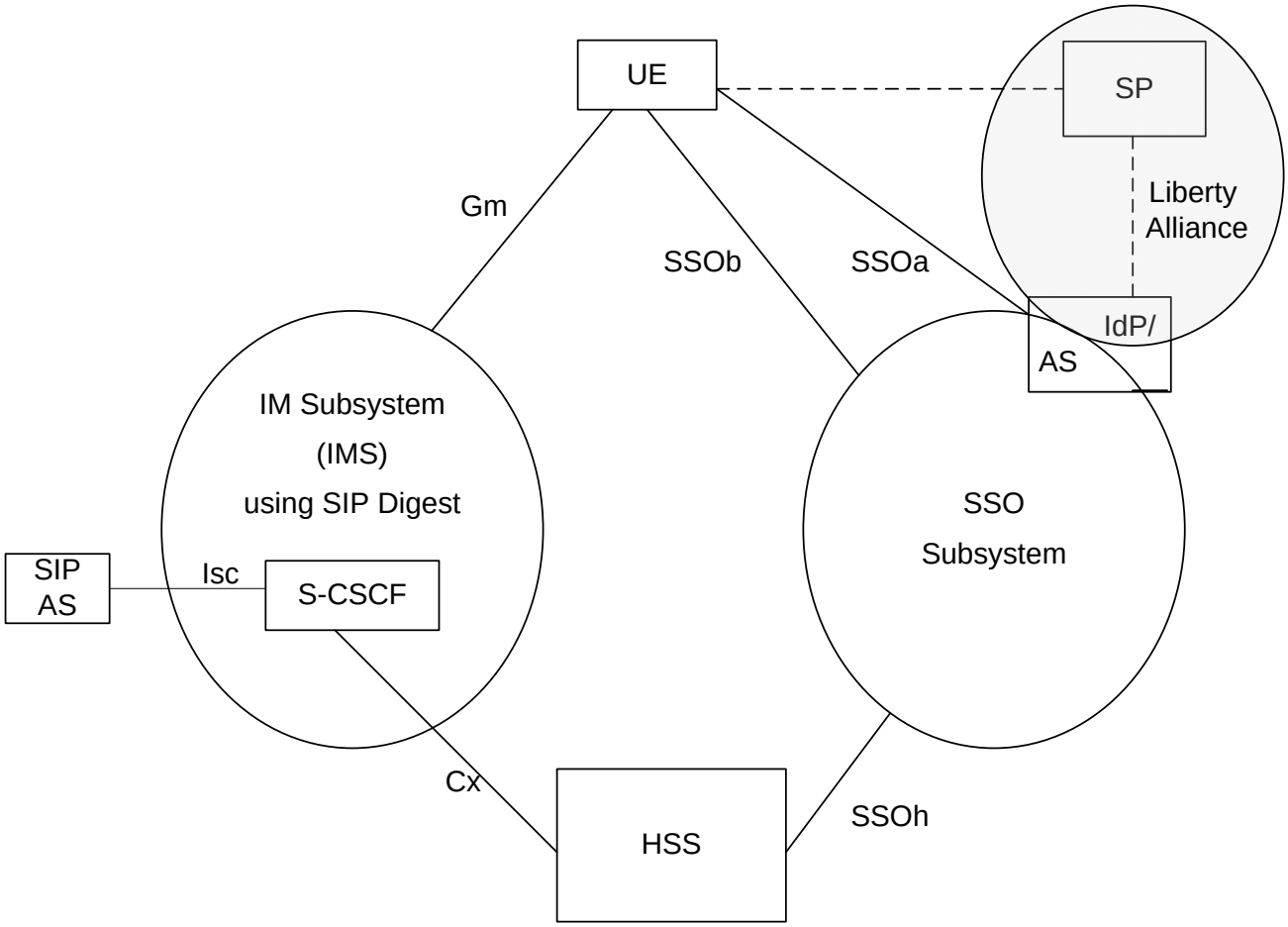


Figure 5.4-1: Relation of SIP Digest-based SSO with the Common IMS and interworking with Liberty Alliance

5.5 Interworking with OpenId

Interworking with OpenID is quite analogous with interworking with Liberty Alliance. The SSO subsystem under study is meant to provide some form of interworking with or support also for, other SSO systems, notably Liberty Alliance, cf. previous clause 5.4.

A high-level architecture for interworking of SIP Digest-based SSO with OpenID that takes into account the baseline architecture from clause 5.2 are shown in Figure 5.5-1 and 5.5-2. The

- Interface between the UE and the RP and the
- Interface between RP and the OP

are out of scope of 3GPP and based on the OpenID protocols.

Interworking of GBA with OpenID systems is described in TR 33.924 [9]. It may be useful to take the approaches in TR 33.924 into account when studying the interworking of the SIP Digest-based SSO subsystem defined in this TR with OpenID, cf. clause 5.1.2. The results of the study shall not be bound by the approaches taken in TR 33.924 in any way, however.

TR 33.924 describes two basic scenarios, in the first one the user browses and authenticates with the same device (c.f. 5.5-1) and in the second one the authentication agent (AA) and the browsing agent (BA) are in different entities. The second split terminal scenario has also several variants depending on the actual connection from the BA to the AA and on which entity triggers the authentication.

NOTE: In both figures below the SSOa interface is based on OpenID protocols and uses HTTPS.

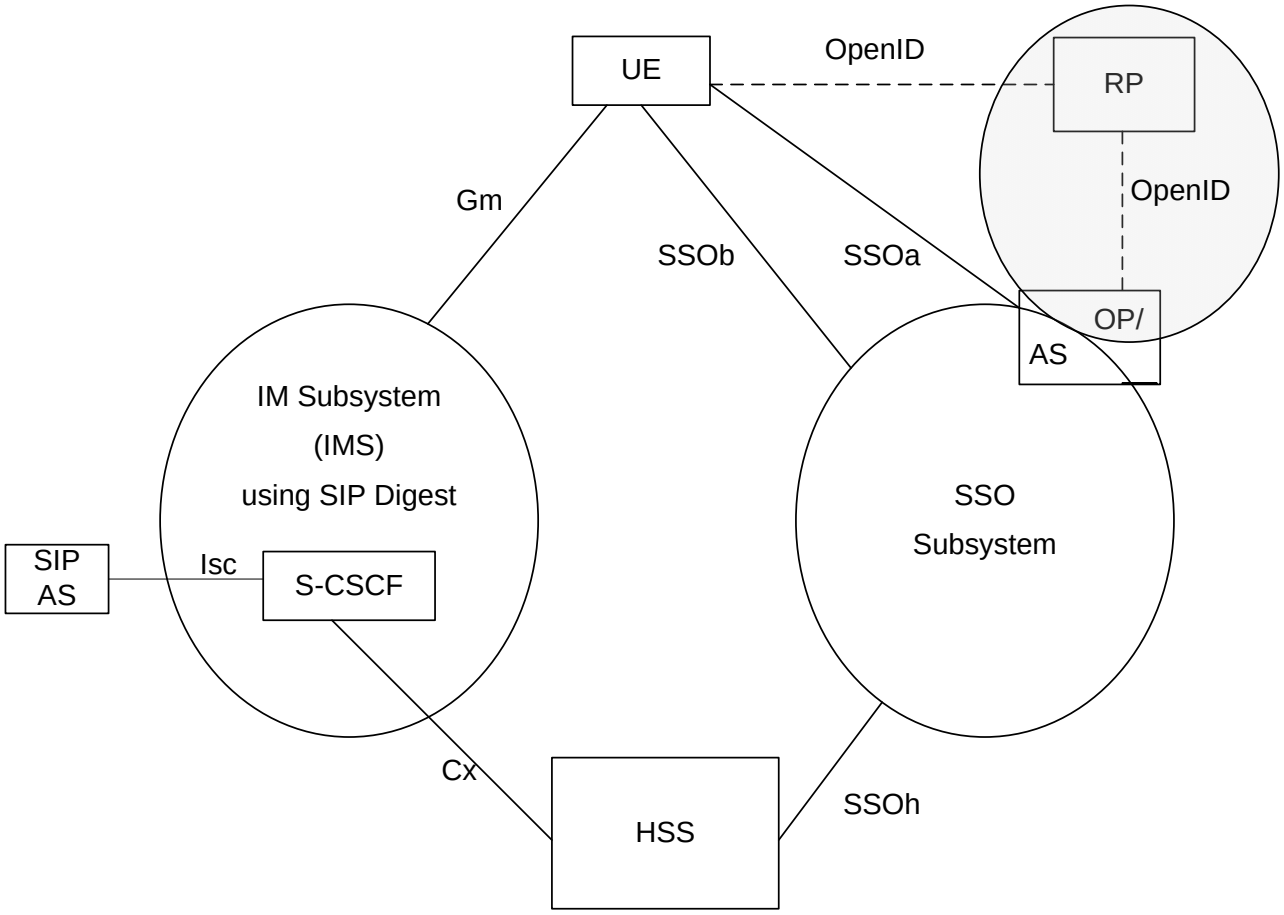


Figure 5.5-1: Relation of SIP Digest-based SSO with the Common IMS and interworking with OpenID

Figure 5.5.-2 covers all the split terminal variants outlined in TR 33.924 [9], but it should be noted that the connection between the BA and AA might not be present depending on the variant.

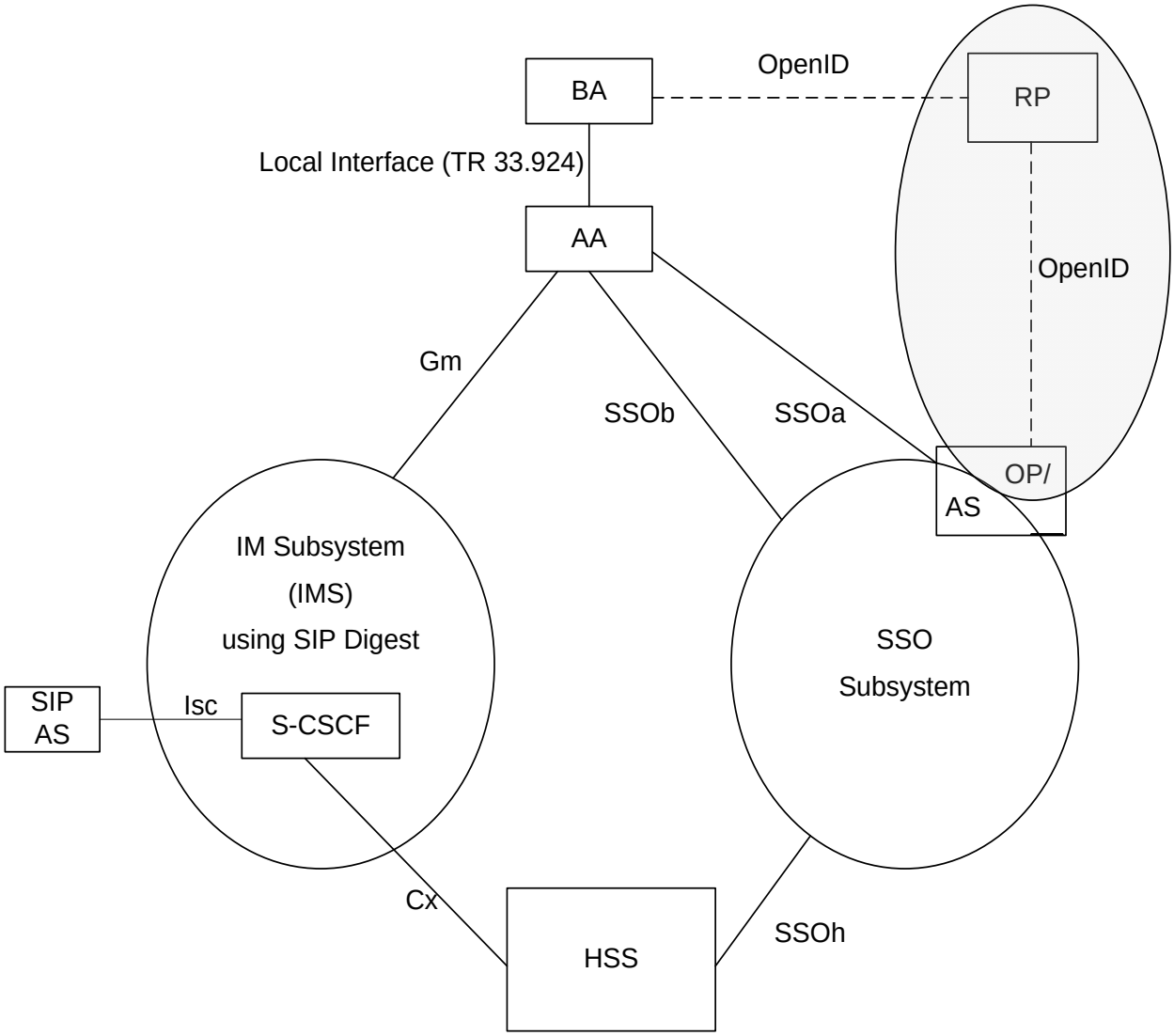


Figure 5.5-2: Relation of SIP Digest-based SSO with the Common IMS and interworking with OpenID for the split terminal scenarios

6 Security Requirements

- The design of a SIP Digest-based SSO system shall allow for the possibility of interworking with a larger range of SSO providers.

- It shall be ensured that the compromise of one SSO service shall not compromise the security of the IMS core or of another SSO service, or an application server.

- It shall be ensured that the compromise of application server or an external SSO server shall not compromise the security of the SSO subsystem.

- Information leakage of the permanent secrets stored in the HSS/HLR due to credential usage for SSO that are available to application servers or entities outside the operator's control shall be prevented.

- The SIP Digest-based SSO service should allow indicating to the application server the strength of the authentication method used.

- IMS core network nodes shall not be modified for supporting the SIP Digest-based SSO service.

- Identity providers and service providers in SSO deployments that were not designed by 3GPP should not need to support additional protocols, besides the protocols already specified for them.

- The approach to utilize SIP Digest authentication for SSO shall maximize commonalities with the already defined 3GPP approaches for interworking with non-3GPP-defined SSO systems as described in TR 33.980 and TR 33.924.

- The operator should be able to control the security level of the SSO system, either by operating the SSO system themselves or by contractual agreements with trusted partners.

- Where user privacy is required, the design of a SIP Digest based SSO system should not allow affiliated non-IMS domain services drawing conclusions about IMS domain identities, e.g., the SSO subsystem should hide IMPIs from application services.

- Any solution should take into account the following design guidelines for HSS-related security:

  - The number of different types of interfaces to the HSS should be minimized in order to keep the complexity of the HSS low. This applies in particular to interfaces over which authentication vectors are retrieved from the HSS as they are highly security-critical.

  - In order to minimize any security risks due to excessive use/abuse of authentication vectors, as well as any performance impact to HSS, the overall number of authentication vectors requested from the authentication centre as well as the number of requests should be kept low. Mechanisms which make economical use of authentication vectors should be preferred. In particular, mechanisms which avoid bursts in authentication vector requests should be preferred.

- The core network load should be minimized and also the SSO subsystem should be designed in an efficient manner to minimize UE authentication and CN interaction to avoid that SSO subsystem is the bottleneck of the application service or other SSO systems.
- Any SSO solution should allow an operator to enforce the condition specified in Annex N of TS 33.203 that SIP Digest authentication shall not apply to access networks defined in 3GPP specifications. According to Annex P of TS 33.203, the enforcement mechanisms may involve implementation-dependent steps.

7 Solutions

7.1 General

This sub clause provides a solution which describes the re-usage of non-UICC credentials to provide security for the access to applications bases on SIP Digest authentication mechanism.

In the High-level architecture for SSO to applications for Common IMS based on SIP Digest, the HSS is the linking element between the IMS subsystem and the SSO subsystem, which results from the fact that HSS stores the permanent credentials used in both subsystems. The credentials used with SIP Digest in the non-UICC are shared secrets or passwords stored in the HSS and in the terminal, or held by the user.

The solution also provides mutual authentication between the application server and the UE. It defines functional elements in the SSO subsystem and their interworking with each other and with the application server and the HSS. These functional elements can provide the SSO service to application servers based on SIP Digest in the UICC-less environment. After the authentication procedure, the UE shares secrets respectively with the SSO and the RP, the SIP Digest credentials can be reused in Common IMS for providing security between a terminal and an application server.

Besides, this solution also supports interworking and exploits commonalities with existing SSO subsystem deployments e.g. OpenID, GBA, Liberty Alliance. It also improves user security by engaging a user-trusted operator in the access control to the applications.

7.2 Solution 1 – SIP Digest based GBA solution

7.2.1 Solution 1 – Architecture for SIP Digest based GBA (GBA_Digest)

In this solution, it is described how SIP Digest authentication can be integrated into the Generic Bootstrapping Architecture (GBA) as defined in TS 33.220 [2]. SIP Digest Authentication in Common IMS is a 3GPP-defined profile of HTTP Digest as defined in RFC 2617 [5].

GBA Digest is an extension of the GBA version currently defined in TS 33.220 [2]. In GBA_Digest, SIP Digest credentials, such as shared secret or password, are re-used instead of the credentials stored in the SIM, USIM or ISIM for authentication.

In GBA_Digest, the function of BSF is extended based on the BSF in TS 33.220 [2].

The architecture of GBA_Digest is described in Figure 7.2-1. For the purpose of GBA_Digest the function of the BSF is extended compared to the BSF as currently defined in TS 33.220 [2].

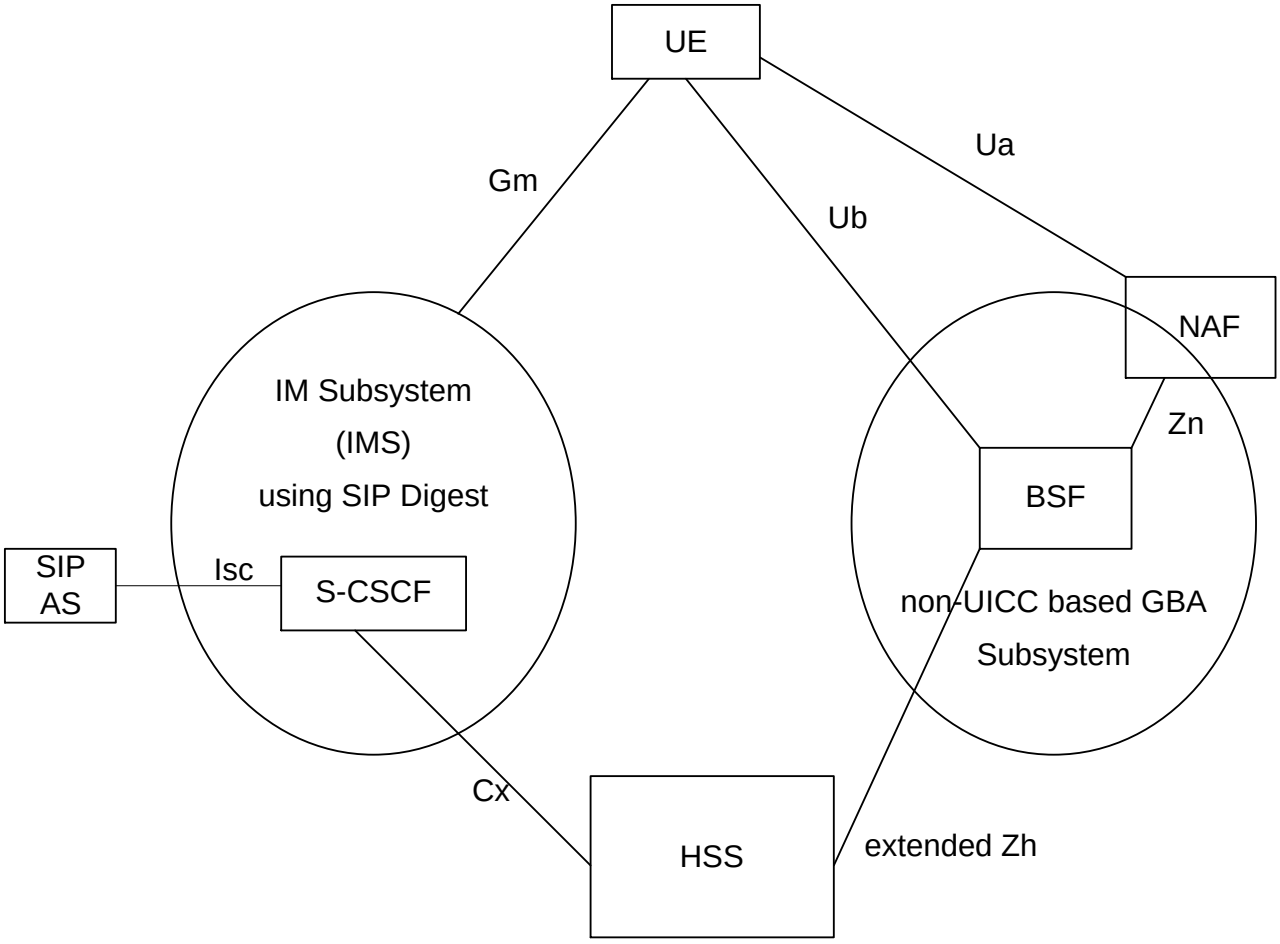


Figure 7.2-1 Architecture of GBA_Digest

Reference points:

- The Ua and Zn [11] interface can be used as defined in TS 33.220, with the extensions for the initiation of bootstrapping mentioned at the start of clause 7.2.2 and for the Zn interface for indications relating to SIP Digest credentials towards the end of clause 7.2.2. The Ua protocol is application specific. It could use e.g. HTTPS, as defined in TS 33.222 [6]. The bootstrapping procedure over the Ub reference point is defined below. It extends the procedure over Ub defined in the main body of TS 33.220 in a way similar to 2G GBA, defined in Annex I of TS 33.220. There is mutual authentication between UE and BSF based on the SIP Digest credential used in the IMS. In addition, there is BSF-to-UE authentication by means of TLS.

- The BSF retrieves a SIP Digest authentication vector (SD-AV) and, if available, the GUSS from the HSS by using an extended Zh interface.

In reference to chapter 5, the SSO Subsystem consists out of the GBA Bootstrapping Server Function (BSF) with SIP digest specific enhancements and the NAF functionality.

NOTE1: GBA only provides an application-specific shared secret, the so-called NAF keys. It is not in itself an authentication mechanism for applications. The derivation of Ks_NAF for GBA_Digest is defined at the end of clause 7.2.2. Clause 7.2.3 explains how Ks_NAF can be used for authentication between a UE and an identity provider (IdP) in an existing Single Sign-On system, such as OpenID or Liberty Alliance.

7.2.2 SIP Digest based GBA (GBA_Digest) bootstrapping procedure and its use

We assume that the UE contacts the NAF and the NAF indicates to the UE that it should use GBA. The NAF demands that bootstrapping is required and gives an indication as described, for the general case, in clause 4.5.1 'Initiation of bootstrapping' of TS 33.220 [2] and, for the special case of an HTTPS-based Ua interface, in clause 5.3 of TS 33.222 [6] and clause 5.2.4 of TS 24.109 [12].

For the special case of an HTTPS-based Ua interface, the messaging contents over Ua reference point to indicate GBA_Digest are described as follows:

1. The request for service message from the UE to the NAF—A new "product" token to the "User Agent" header will indicate that the UE requests the use of GBA_Digest by including a static string "3gpp-gba-sip-digest".

2. The code 401 "Unauthorized" response to the request for service (NAF to UE) contains the WWW-Authenticate header—A new prefix "3GPP-bootstrapping-gba-sip-digest@" in the "realm" attribute in the header will indicate that the UE is allowed to perform non-UICC SIP Digest GBA. If the 401 "Unauthorized" response received by the UE does not include an indication with this new prefix, the UE shall not use GBA_Digest.

This indication shall trigger the UE to run GBA_Digest bootstrapping procedure over the Ub interface (suitably extended for the purposes of GBA_Digest) as is described below:

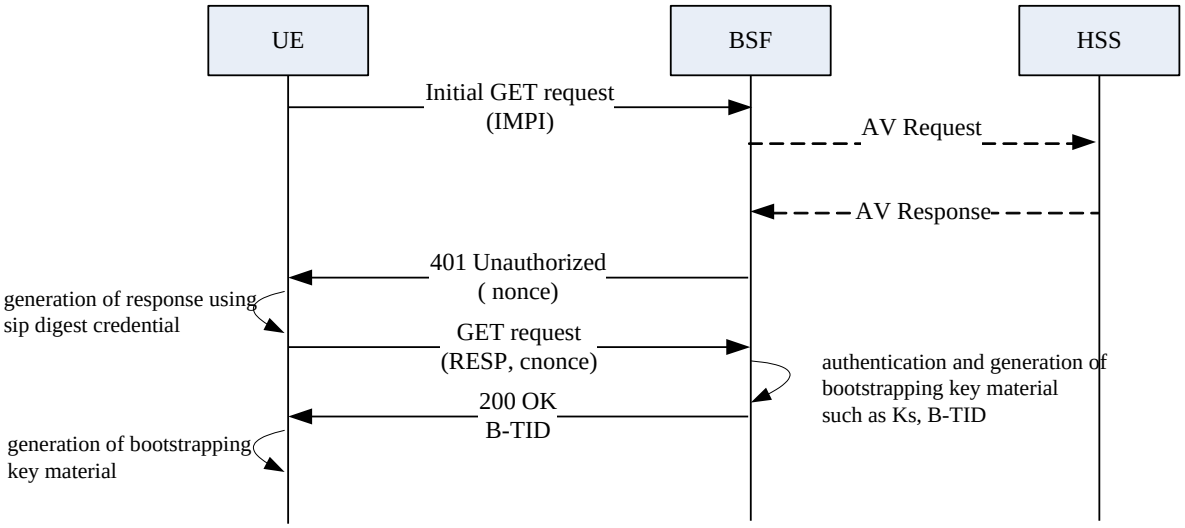


Figure 7.2-2 GBA_Digest bootstrapping procedure

NOTE 1: The figure above only shows an example flow for visualization and not all details are included.

Step 0:

The UE and the BSF establish a TLS tunnel with server authentication using a server certificate. The use of TLS message integrity is mandatory, while the use of TLS encryption is optional. All further messages between the BSF and UE are sent through this tunnel.

NOTE 2: TLS encryption can be useful for protecting the user identity privacy when the TMPI mechanism defined in TS 33.220 [2] is not used.

Step 1:

In this HTTP request message from the UE to the BSF, the UE shall include Authorization header containing the IMPI and a token indicating the use of GBA_Digest.

NOTE 3: It should be noted that it would have been possible to select an alternative information flow as follows: the UE would not convey its IMPI to the BSF in step 1 but only in the second GET request (step 5). In this case the AV request / response exchange between BSF and HSS would be deferred, until this second GET request was received by the BSF. But the information flow presented here was preferred because it maximizes commonality with GBA, as defined in TS 33.220 [2]. This would be possible in GBA_Digest, in contrast to GBA, as the nonce serving as a challenge in GBA_Digest is generated by the BSF while the challenge RAND | AUTN in GBA has to be generated by the HSS. The delayed approach would maximize the similarities with HTTP Digest as used on the web. Note that since this flow requires the realm to be pre-configured in the BSF, all users served by the BSF must use identical realm values in the HSS.

NOTE 4: The token is only required for GBA_Digest, not for AKA-based GBA variants already defined in TS 33.220. This ensures backward compatibility.

Step 2:

The BSF requests a SIP Digest Authentication Vector (SD_AV) and, optionally, inserts into the request the timestamp of the locally stored GUSS for this user, if such is available.

NOTE5: If there is no timestamp the HSS will send the GUSS for this user if it exists if there is a timestamp, then the HSS will only send the GUSS if it is more recent.

The SD-AV is defined in TS 33.203, Annex N. The username field in the Multimedia Auth Request contains the IMPI. If a token indicating the use of GBA_Digest was received by the BSF in step 1 then the BSF shall indicate in the Multimedia Auth Request that the requested Authentication Vectors are for SIP Digest credentials.

NOTE5: If no token indicating the use of GBA_Digest was received by the BSF in step 1 then the BSF knows that the subscription is AKA-based, but the BSF may not be able to determine at this point whether the subscription type is 2G or 3G, hence the authentication scheme is not indicated in the Multimedia Auth Request. Note that the subscription type can be either 2G or 3G, but not both.

Step 3:

The HSS retrieves the SIP Digest authentication vector SD-AV corresponding to the IMPI and sends it to the BSF in a Multimedia Auth Answer. The GUSS is only returned when requested by the BSF, supported by the HSS, and the timestamp in the BSF indicates that the one in the HSS is fresher.

The qop value shall be set to "auth".

NOTE6: The additional protection afforded by qop set to "auth-int" is not needed due to the fact that the messages exchanged between UE and BSF run inside a TLS tunnel. The use of qop set to "auth" is less complex.

Step 4:

In the 401 response from the BSF to the UE, the BSF shall include parameters to WWW-Authenticate header as specified in RFC 2617 [5].

The realm, qop, and algorithm are provided in the SD-AV and the nonce=base64encode(16 byte random value) is generated by the BSF.

The BSF sends the challenge to the UE in an HTTP 401 Unauthorized response.The WWW-Authenticate header is included in the HTTP 401 Unauthorized response.

Step 5:

When responding to a challenge from the BSF, the UE generates a cnonce randomly, and calculates the response RESP. The RESP is put into the Authorization header and sent back to the BSF in the GET request.

RESP is computed as a Digest-response according to RFC 2617 [5] (HTTP Digest) from the most recent GBA_Digest challenge and a password ‘passwd’ that is generated as follows:

passwd = KDF (H(A1), "GBA_Digest_RESP", TLS_MK_Extr)

where H(A1) is the hash of the following three parameters: the user name and password used by the user in IMS for SIP Digest according to TS 33.203, Annex N, and the realm, cf. also RFC 2617 [5]. "GBA_Digest_RESP" is a character string. TLS_MK_Extr is extracted from the TLS master key according to RFC5705 [15] with the optional context value being omitted and the label set to "EXPORTER_GBA_Digest", and the length set equal to the length of the TLS master secret (48 bytes). KDF is the key derivation function as specified in Annex B in TS 33.220 [2].

NOTE7: A cautionary note on notation: According to RFC 2617 [5], the computation of RESP from the password ‘passwd’ defined above entails again a parameter called H(A1). This parameter will differ from the value of H(A1) that is input to the above formula because the passwords from which these two H(A1) values are derived differ. But no new notation is deemed necessary here as the notation H(A1), when H(A1) is derived from ‘passwd’, is not explicitely used in the text of the present document.

NOTE8: The label for the exporter function EXPORTER_GBA_Digest, cf. [15], needs to be registered with IANA when the work proceeds to normative stage.

Step 6:

Upon receiving a GET request carrying the authentication response RESP, the BSF checks that the expected RESP (calculated by the BSF in the same way as the UE in step 5) matches the received RESP. If the check is successful then the user has been authenticated.

The BSF then derives Ks as follows:

Ks = KDF (H(A1), "GBA_Digest_Ks", TLS_MK_Extr, RESP)

where H(A1), RESP, and TLS_MK_Extr are defined as in step 5, and "GBA_Digest_Ks" is a character string.

The BSF generates the bootstrapping transaction identifier (B-TID) for the IMPI and stores the tuple <B-TID,IMPI,Ks>. The B-TID shall be constructed in the format of a NAI by taking the nonce from step 4, and the BSF server name, i.e. nonce@BSF_server_domain_name.

The BSF sends 200 OK response to the UE to indicate the success of the authentication.

In this message from the BSF to the UE, the BSF shall include bootstrapping transaction identifier (B-TID) and the key lifetime in an XML document in the response payload. The BSF may also include additional server specific data to the XML document.

An Authentication-Info header according to RFC 2617 [5] shall be included into the 200 OK response.

The UE aborts the procedure if the server authentication according to [5] fails. Otherwise, the UE derives Ks in the same way as the BSF did above.

NOTE9: The B-TID construction above is almost identical to the one used in TS 33.220. The difference is that in TS 33.220 the username part is constructed from the (base64 encoded) RAND value.

After successful bootstrapping procedure the UE and the BSF contain the key Ks and the B-TID. In addition, the BSF stores an indication of the underlying security quality, i.e. SIP Digest, like it is doing for 2G GBA. The key Ks is then used in the BSF and in the UE to derive NAF specific key(s) Ks_NAF to secure Ua reference points in the following way:

- Ks_NAF is computed as Ks_NAF = KDF (Ks, "gba-digest", nonce, IMPI, NAF_Id), where KDF is the key derivation function as specified in Annex B of TS 33.220 [2], and the input parameters consist of the user's IMPI, the NAF_Id and 'nonce'. 'nonce' is the nonce that was used for computing the RESP that was input to the derivation of Ks . The NAF_Id is constructed as in TS 33.220 [2], clause 4.5.2. The "gba-digest" parameter is a static character string.

1) NOTE10: The above derivation of Ks_NAF is analagous to the derivation in TS 33.220 [2], clause 4.5.2, step 9 and the same KDF can be utilized.

When the BSF delivers a Ks_NAF to a NAF it shall indicate that this Ks_NAF was derived from SIP Digest credentials. The BSF shall deliver a Ks_NAF to the NAF only after the NAF has indicated its willingness to accept SIP Digest credentials.

The KDF shall be implemented in the BSF and in the terminal.

The UE and the BSF shall store the key Ks with the associated B-TID and the nonce for further use, until the key Ks is updated or deleted.

7.2.3 Interworking of SIP digest based GBA with other SSO systems

The approach outlined above for the non-UICC based GBA solution could interwork with existing SSO subsystems, such as OpenID and Liberty Alliance.

The Ks_NAF derived from the Ks by the UE and the BSF can be used for authentication between UE and Identity Provider in OpenID as outlined in TR 33.924 or for Liberty Interworking as outlined in TR 33.980. Also for other services the derived Ks_NAF can be used.

The following figures 7.2-3 depict the interworking between non-UICC based GBA, IMS subsystem and Liberty Alliance.

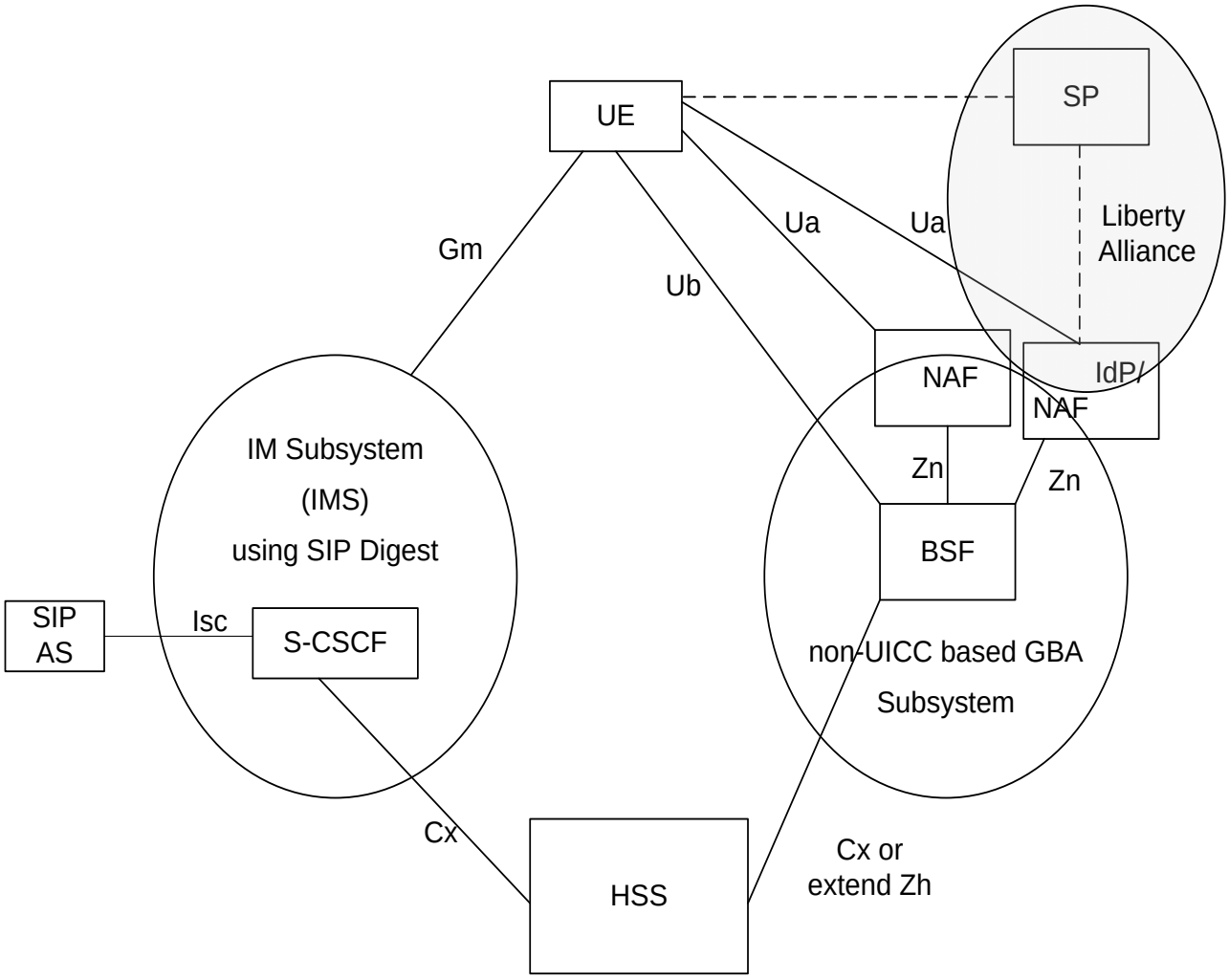


Figure 7.2-3 interworking with Liberty Alliance

NOTE: The figure 7.2.-3 contains a potential set-up where the BSF supports an IdP/NAF, but also delivers NAF specific keys directly to an application server, which is not part of the service supported by the IdP.

7.2.4 Evaluation

Compatibility with other GBA-related specifications:

GBA_Digest can be used with TSs 33.221 [16] and 33.222 [6] because GBA_Digest provides as output a key Ks_NAF (cf. clause 7.2.3) and an identifier B-TID (cf. clause 7.2.2) as well as information in the BSF about the underlying security quality, i.e. SIP digest authentication, (cf. clause 7.2.2). This is the input required for TSs 33.221 and 33.222 to become applicable.

Single sign-on to an AS over the Ut reference point can, hence, also be provided using GBA_Digest, as TS 33.141 [4] defines the security for the Ut reference point by referring back to TS 33.222 [6].

Advantages of solution 1:

The non-UICC based GBA solution has the following advantages:

- Re-use of the SIP Digest subscriber credentials stored in HSS.
- Re-use of Ua interface and Zn interface (with adding one field for indicating GBA_Digest) as specified in TS 33.220 [2].

- Support for the existing GBA based applications.
- Enhanced security over plain SIP Digest (without TLS)
- Seamless evolution and migration to GBA variants using a SIM, USIM, or ISIM possible on the network side when clients are upgraded to contain a UICC

Security considerations on solution 1:

The role of TLS:

The use of TLS in GBA_Digest provides the following advantages:

- Parameters derived from the SIP Digest password and exchanged between UE and BSF can be observed by an attacker when the TLS connection is not encrypted. In order to make it much more difficult for the attacker to infer (e.g. by codebook attacks) the SIP Digest password from the observed parameters GBA_Digest was constructed such that the computation of these parameters additionally requires the knowledge of the TLS master key.

  It is true that the attacker could just as well attempt to infer the SIP Digest password from parameters exchanged during a SIP Digest run in IMS, as defined in TS 33.203, Annex N. However, this is not possible in IMS when these parameters are sent over an encrypted TLS connection, as defined in TS 33.203, Annex O. In order to achieve at least the same the level of protection against password cracking in GBA_Digest as that afforded in IMS by TS 33.203, Annex O, the TLS master key is used in the construction of the GBA_Digest parameters.

  But it should be noted that the use of TLS differs between IMS and GBA_Digest: it is not required in GBA_Digest to use TLS with encryption, cf. clause 7.2.2, step 0. Mandating TLS encryption was avoided as it might conflict with national regulation.

- TLS provides 'perfect forward secrecy'. This property means that, when a compromise of a permanent secret has occurred, session keys agreed prior to the time of the compromise cannot be reconstructed by the attacker knowing the permanent secret. As the TLS master key is input to the computation of the key Ks resulting from a GBA_Digest run this property is inherited by GBA_Digest. This property is especially desirable for a password-based scheme as passwords are more prone to compromise than UICC-based credentials.

- The role of TLS in protecting against man-in-the-middle attacks is described below.

Protection against man-in-the-middle attacks:

Two types of man-in-the-middle attacks need to be considered:

Man-in-the-middle attacks through BSF impersonation

- An attacker could set up a server and trick a terminal into believing that it was talking to a genuine BSF and sending the authentication response to the attacker's server. The attacker could then use this response to impersonate the user towards the genuine BSF.

- This can be prevented by requiring the use of TLS between UE and BSF and ensuring that the UE authenticates the TLS server before sending anything over the TLS connection. This authentication is provided by TLS server certificates and the fact that the UE checks as part of the authentication process that the name in the certificate coincides with the BSF name configured in the UE.

- It should be noted, though, that even a successful BSF impersonation attack would be of limited value to the attacker as he could still not obtain the key Ks. However, the attack would not be detected during the GBA_Digest run, but only later when the key Ks would be used.

Man-in-the-middle attacks in tunnelled authentication protocols

- When authenticating the client end point of a TLS session by means of a shared key protocol the so-called 'man-in-the-middle-attacks in tunnelled authentication protocols' [17] need to be considered. The man-in-the-middle-attacks exploit a situation where the same authentication protocol and credentials are used by the client in two different contexts, and the authentication responses may be sent both inside and outside the TLS

connection. In the context of GBA_Digest, the shared key protocol used for client authentication is HTTP Digest. An attacker could establish a TLS connection with the BSF and at the same time trick the UE into running HTTP Digest with him outside the TLS connection, thus obtaining the desired authentication response. The attacker would then send the obtained response over the TLS connection to the BSF. In this way, the attacker could impersonate the UE towards the BSF without knowing the user's password.

- In GBA_Digest, this attack is thwarted by making the authentication response different from any parameter used outside the context of the GBA_Digest protocol through the use of different input parameters that, in particular, binds the authentication response to the TLS master secret.

- It should be noted, though, that even a successful man-in-the-middle-attack would be of limited value to the attacker as he could still not obtain the key Ks. However, the attack would not be detected during the GBA_Digest run, but only later when the key Ks would be used.

In the SSO solution described in 7.2. the BSF server has to support the following 3GPP specific functionalities:

- Interface to HSS to obtain the SD-AV, user profile and related attributes
- Storage and handling of data related to (nonce, H(A1))

7.3 Solution 2 – SIP Digest based Authentication and Lightweight Security (SDALS) solution

7.3.1 Architecture and Interworking for SDALS

7.3.1.1 Solution 2 – High-level Architecture

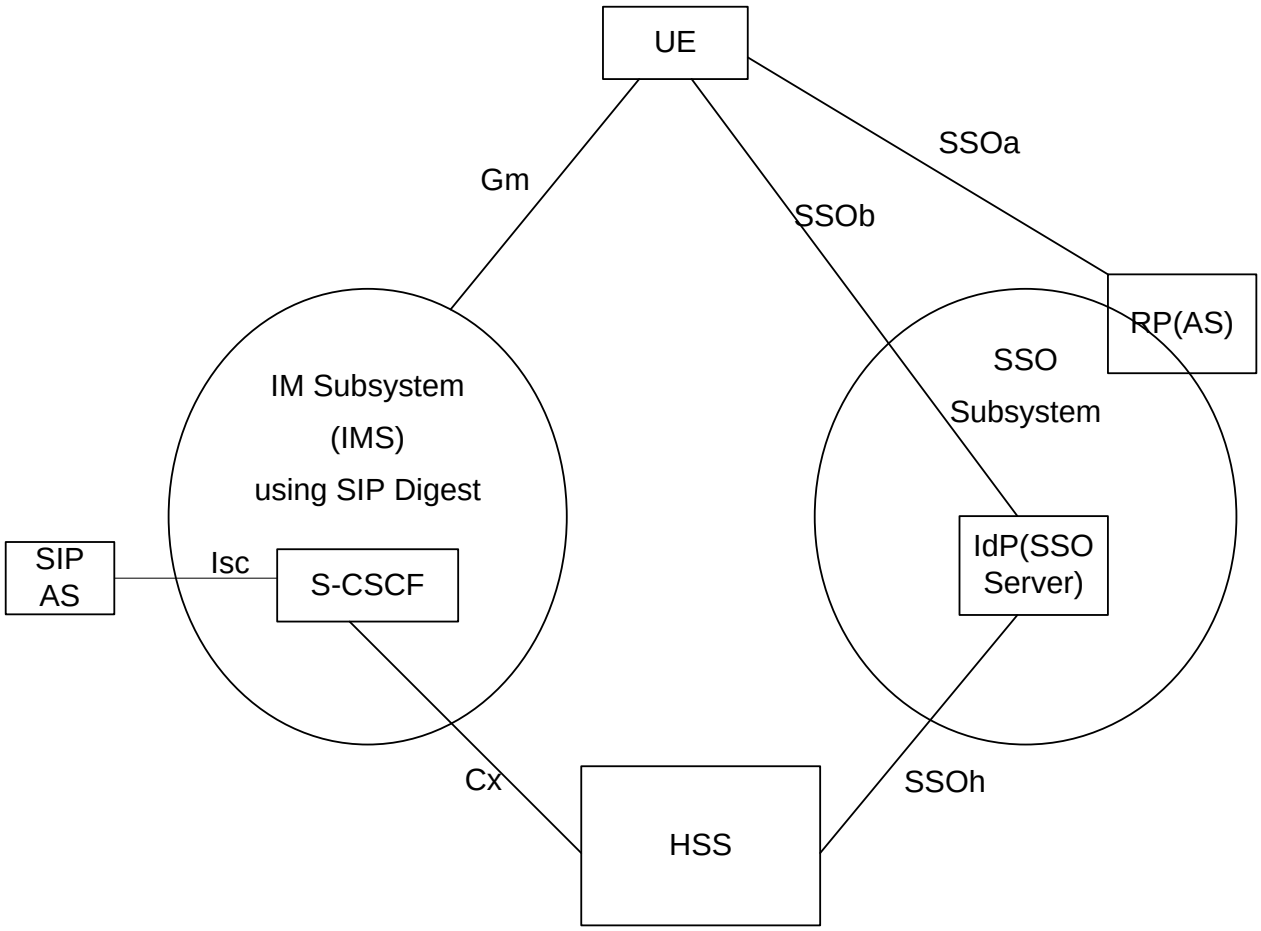


Figure 7.3-1: Architecture for SIP Digest based Authentication and Lightweight Security (SDALS)

Figure 7.3-1 shows a high-level architecture for SIP Digest based Authentication and Lightweight Security (SDALS). The authentication between UE and IdP (SSO Server) is based on SIP Digest, and the solution creates two keys, one key (k_0) used between UE and IdP (SSO Server) is negotiated and generated based on challenge, and the other key (K_1) used between UE and RP is generated by IdP (SSO Server) and delivered to UE and RP via credential.The IdP (SSO Server) contained in the substructure of the SSO subsystem which could be leveraged to enable application servers and on the user side to establish shared keys could provide the SSO service to application servers based on SIP Digest. There are four new reference points linking the UE、 the HSS and the RP (AS) to the IdP (SSO Server), as well as linking the UE and the RP. The reference points SSOh and SSOb could provide the SSO functionality.

General requirements for the functionality of the IdP (SSO Server) are:

- It shall be able to communicate securely with the HSS.
- It shall be able to perform authentication based on SIP digest based credentials.
- It shall be able to generate a shared key and a session key with a UE.
- It shall be able to manage the Identity-related security keys and credentials.

The required functionalities for a UE are:

- Supporting authentication using SIP digest credentials.
- Generating a shared key and a session key with the IdP.

The requirements for the reference point SSOa are:

- The reference point SSOa carries the application protocol,which is secured using a session key between a UE and the IdP

The requirements for the reference point SSOb are:

- The reference SSOb shall provide mutual anthentication beween the IdP and a UE based on the SIP Digest method
- The IdP and a UE shall be able to generate shared keys
- The IdP shall be able to identify the UE
- The IdP shall be able to indicate to the UE the lifetime of the shared key

The requirements for the reference point SSOh are:

- The IdP shall be able to retrieve SD-AV from the HSS, so we can re-use the Cx interface as the SSOh reference point.

7.3.1.2 Interworking of SDALS (solution 2) with other SSO systems

7.3.1.2.1 Background

The solution of SDALS has the following advantages:

The approach outline above for the non-UICC solution could provide some forms of interworking with other existing SSO subsystem, notably OpenID and Liberty Alliance.

7.3.1.2.2 Co-hosting AS and OP

The architecture for interworking of the SDALS with OpenID that co-hosts AS and OP, which takes into account the baseline architecture from clause 7.3.1.1 is shown in Figure 7.3-2. It makes the Application Server (AS) in the SSO subsystem and the OpenID Provider (OP) in the OpenID as an entity, and the interface between the UE and the RP and the interface between RP and the OP are out of scope of 3GPP and based on the OpenID protocols.

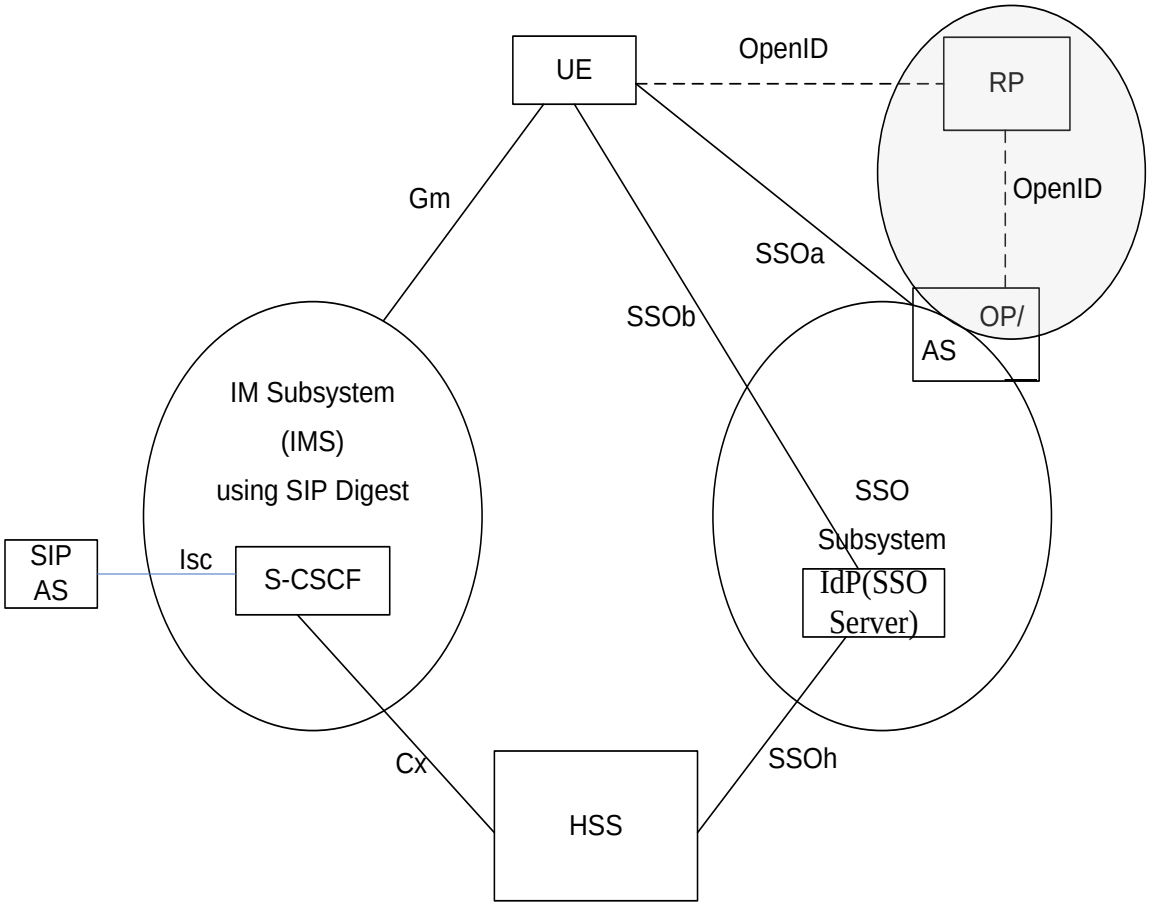


Figure 7.3-2: Interworking with OpenID that co-hosts AS and OP

7.3.1.2.3 Co-hosting AS and IdP (Liberty Alliance)

Interworking with Liberty Alliance which co-hosts AS and IdP (Liberty Alliance) is quite analogous with interworking with OpenID that co-hosts AS and OP.

The approach outline above for the non-UICC solution could provide some forms of interworking with other existing SSO subsystem, Interworking of GBA with Liberty Alliance systems is described in TR 33.980 [8]. It may be useful to take the approaches in TR 33.980 into account when studying the interworking of the SIP Digest-based SSO subsystem defined in this TR with Liberty Alliance. The results of the study shall not be bound by the approaches taken in TR 33.980 in any way, however.

The approach outline above for the non-UICC solution could provide some forms of interworking with Liberty Alliance, The architecture for interworking of the SDALS with Liberty Alliance that co-hosts AS and IdP (Liberty Alliance), which takes into account the baseline architecture from clause 7.3.1.1 is shown in Figure 7.3-3. It makes the Application Server (AS) in the SSO subsystem and the Identity Provider (L_IdP) in the Liberty Alliance as an entity. The interface between the UE and the SP, and the interface between the SP and the L_IdP are out of scope of 3GPP, both of them are based on the Liberty Alliance protocols.

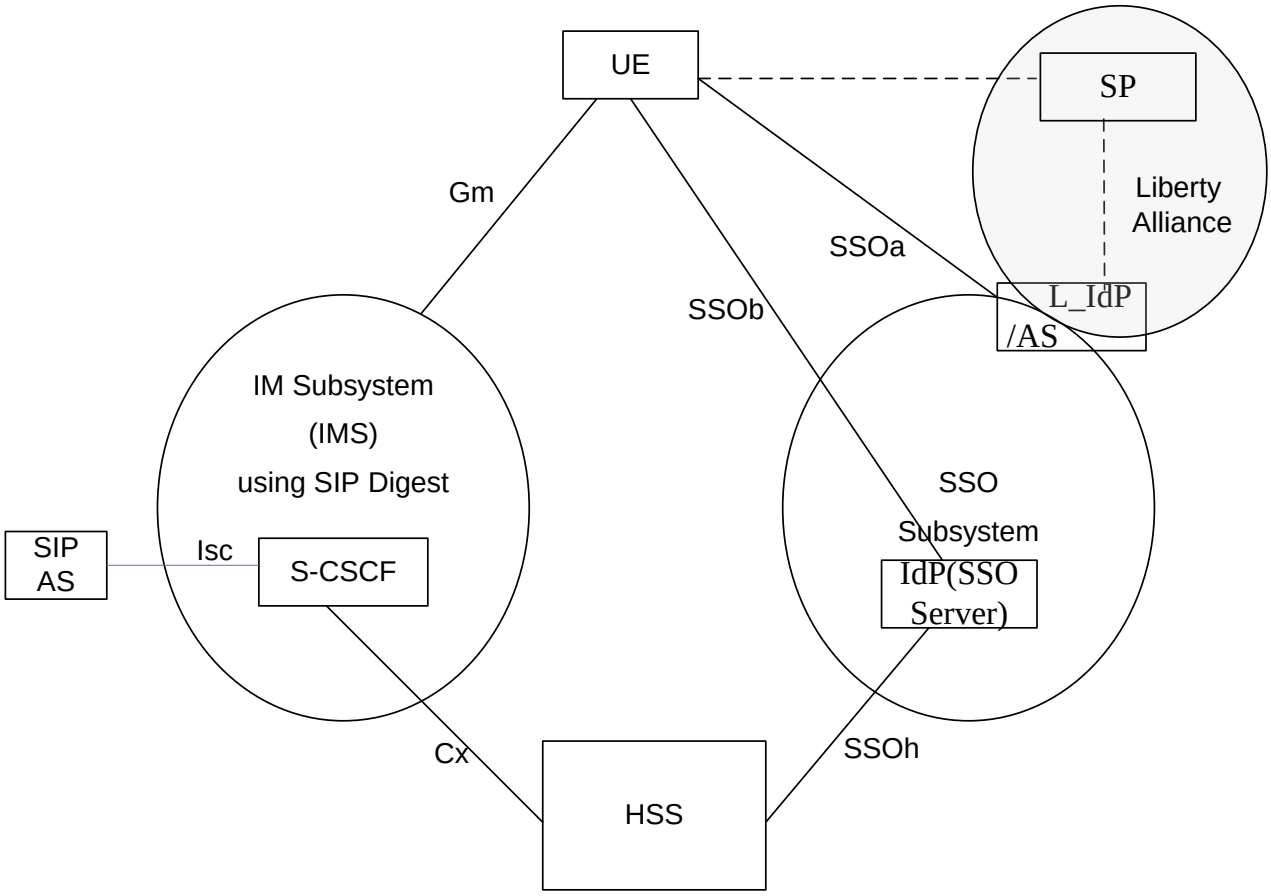


Figure 7.3-3: Interworking with Liberty Alliance that co-hosts AS and IdP (Liberty Alliance)

7.3.1.2.4 Co-hosting IdP (SSO Server) and OP

The architecture for interworking of the DT-Digest with OpenID that co-hosts IdP (SSO Server) and OP, which takes into account the baseline architecture from clause 7.3.1, is shown in Figure 7.3-4. It makes the IdP in the SSO subsystem and the OpenID Provider (OP) in the OpenID as an entity.

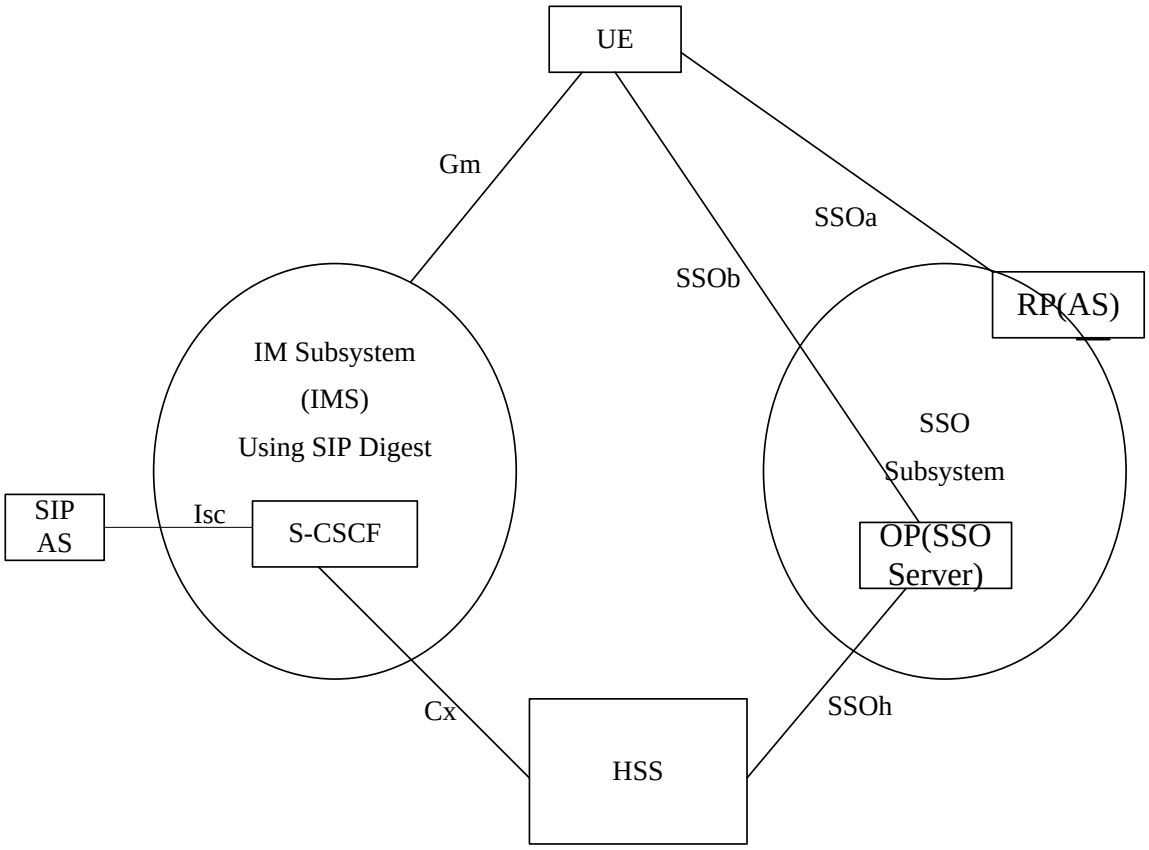


Figure 7.3-4: Interworking with OpenID that co-hosts IdP (SSO Server) and OP

7.3.2 Message Flows for Solution 2 SDALS

7.3.2.1 Basic Message Flow

Editor's Note: The solution above and below have some overlap and need to be sorted out, how is for further study.

The solution realizes a SSO function that is available when an IMS UE is authenticated over SIP Digest authentication mechanism. Figure 7.3-5 shows the message flow of the authentication process to realize SIP Digest-based SSO with the Common IMS in the UICC-less environment.

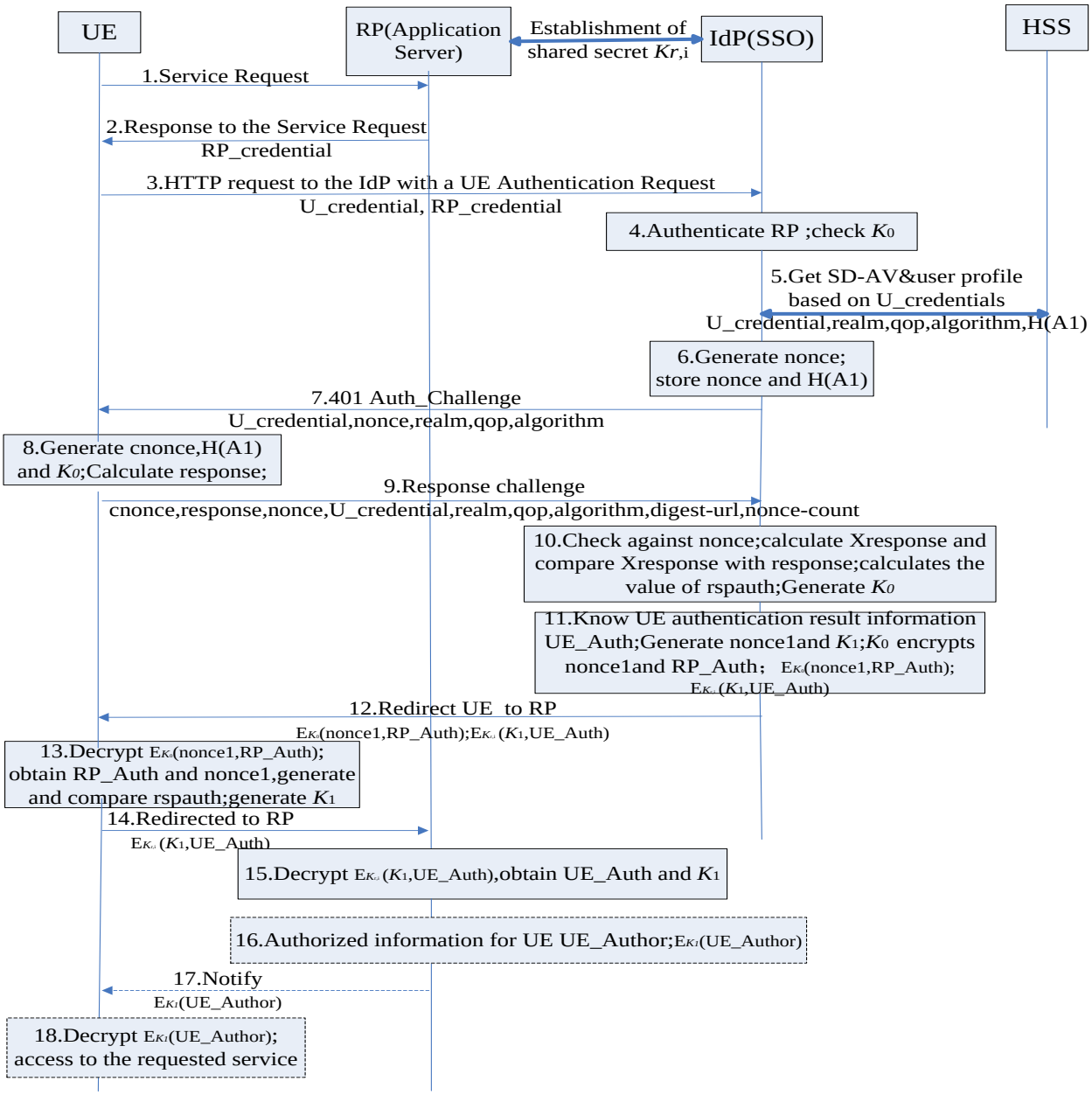


Figure 7.3-5 SDALS Procedure

The basic steps are as follows:

1. The UE issues a service request to RP without user identifier.
2. The RP sends a response to the UE the RP Authentication Request. The response includes the RP identifier (RP_credential). The RP identifier may be FQDN.
3. The UE sends a HTTP request to the IdP known by the UE with the UE authentication request and RP Authentication Request. The request includes the UE identifier (U_credential). The UE identifier may be IMPI or TMPI.

   NOTE 1: The address of the IdP (SSO Server) can be configured in UE, or, can be generated by UE based on UE identifier, e.g. if the user identifier is user_name@domain, then the address of IdP (SSO Server) can be idp_sso.domain.

4. The IdP authenticates the RP based on the RP_credential and generates related authentication result RP_Auth. According to the U_credential, the IdP first checks whether there is already a shared secret K_0 between the UE and IdP. If K_0 exists, the process jumps to step 11; otherwise, the process goes on to the next step.

NOTE2: The RP and the IdP shall have a shared secret ($K_{r,i}$) using existing mechanism, for example, using the Diffie-Hellman Key Exchange Protocol or pre-shared secret, the details of shared key establishment between the RP and IdP are out of scope. With this shared secret the IdP can sign subsequent messages and the RP can verify those messages.

5. The IdP sends authentication request to the HSS, it then obtains the SIP Digest authentication vector SD-AV and the user profile based on the U_credential from the HSS. The SD-AV consists of the qop (quality of protection) value, the authentication algorithm, realm, and a hash, called H (A1), of the U_credential, realm, and password. Refer to RFC 2617[5] for additional information on the values in the authentication vector for SIP Digest based authentication. In a multiple HSS environment, the IdP may have to obtain the address of the HSS where the UE is stored by querying the SLF.
6. The IdP generates a random nonce, stores H(A1) and the nonce against the U_credential.
7. The IdP sends a 401 Auth_Challenge to the UE which includes the nonce, the realm, qop, algorithm and U_credential.
8. Upon receiving the challenge, the UE generates a random cnonce and the H(A1), and then generates the shared secret K_0 based on the H(A1), the cnonce, etc. It then uses the cnonce as well as parameters provided in the 401 Auth_Challenge such as nonce, U_credential and qop to calculate an authentication response according to RFC 2617[5].
9. The UE sends a response to the IdP which includes the cnonce, the nonce, the response, the realm, the U_credential, qop, algorithm, nonce-count and digest-url.
10. Upon receiving the response, the IdP uses the previously stored nonce to check against the nonce included in the response. If the check is successful, the IdP calculates the expected response (Xresponse) using the previously stored H (A1) and the nonce together with other parameters contained in the response (e.g.cnonce, nonce-count, qop, as specified in RFC 2617[5]) and uses this to check against the response sent by the UE. If the check is successful, the authentication of the UE is succeeded, else the authentication fails. If the UE is successfully authenticated, the IdP calculates the value of rspauth based on SIP Digest as specified in RFC 2617 [5] and generates the shared secret K0 based on the H(A1), the cnonce, etc.
11. The IdP knows the User authentication conclusion (UE_Auth); and then the IdP generates a random nonce1 and generates a shared secret K_1 based on K_0 and nonce1. The IdP encrypts the nonce1 and RP_Auth using K_0, i.e. E_{K_0}(nonce1,RP_Auth); and encrypts the K_1 and UE_Auth using $K_{r,i}$, i.e. $E_{K_{r,i}}$ (K_1,UE_Auth).
12. The IdP sends the UE an message including E_{K_0}(nonce1,RP_Auth) and $E_{K_{r,i}}$ (K_1,UE_Auth) ,the value of rspauth with redirection.
13. The UE decrypts the E_{K_0}(nonce1,RP_Auth) and then obtains RP_Auth and nonce1. Based on the RP_Auth the UE knows the legitimacy of the requested RP. If the authentication result indicates that the RP is not valid, the UE will stop visiting the RP. The UE calculates the rspauth in the same way as the IdP did in step 9, and uses it to check against the rspauth sent by the IdP. If the check is successful, the authentication of the Network is succeeded, else the authentication fails. If the Network is successfully authenticated, and then the UE will generates the shared secret K_1 based on K_0, nonce1.
14. The message sent by the IdP is redirected to the RP including $E_{K_{r,i}}$ (K_1,UE_Auth).
15. The RP decrypts the $E_{K_{r,i}}$ (K_1,UE_Auth), and obtains UE_Auth and K_1.
16. After verifying the UE_Auth, the RP generates authorization information for the UE, i.e. UE_Author and encrypts UE_Author using K_1 E_{K_1}(UE_Author).
17. The RP notifies the UE about the authorization information.
18. The UE decrypts the E_{K_1}(UE_Author) and then accesses to the requested service.

NOTE3: The last 3 steps 16, 17 and 18 are application specific, they are optional steps and not required for SSO authentication purpose.

If there is a failure in steps 1 through 15 – the authentication procedure stops.

7.3.2.2 Message Flow with IdP (SSO Server) and OP co-hosting

The SSO subsystem under the solution can provide some forms of interworking with, or support for, other SSO systems, notably OpenID and Liberty Alliance. In the following a message flow of the authentication process is defined to describe business cases where while an operator wishes to be OpenID provider. It makes the operator a critical part of OpenID framework in this case, and allows the operator to leverage their valuable assets, such as subscription credentials and their customers' trust, effectively enabling operators to become OpenID providers.

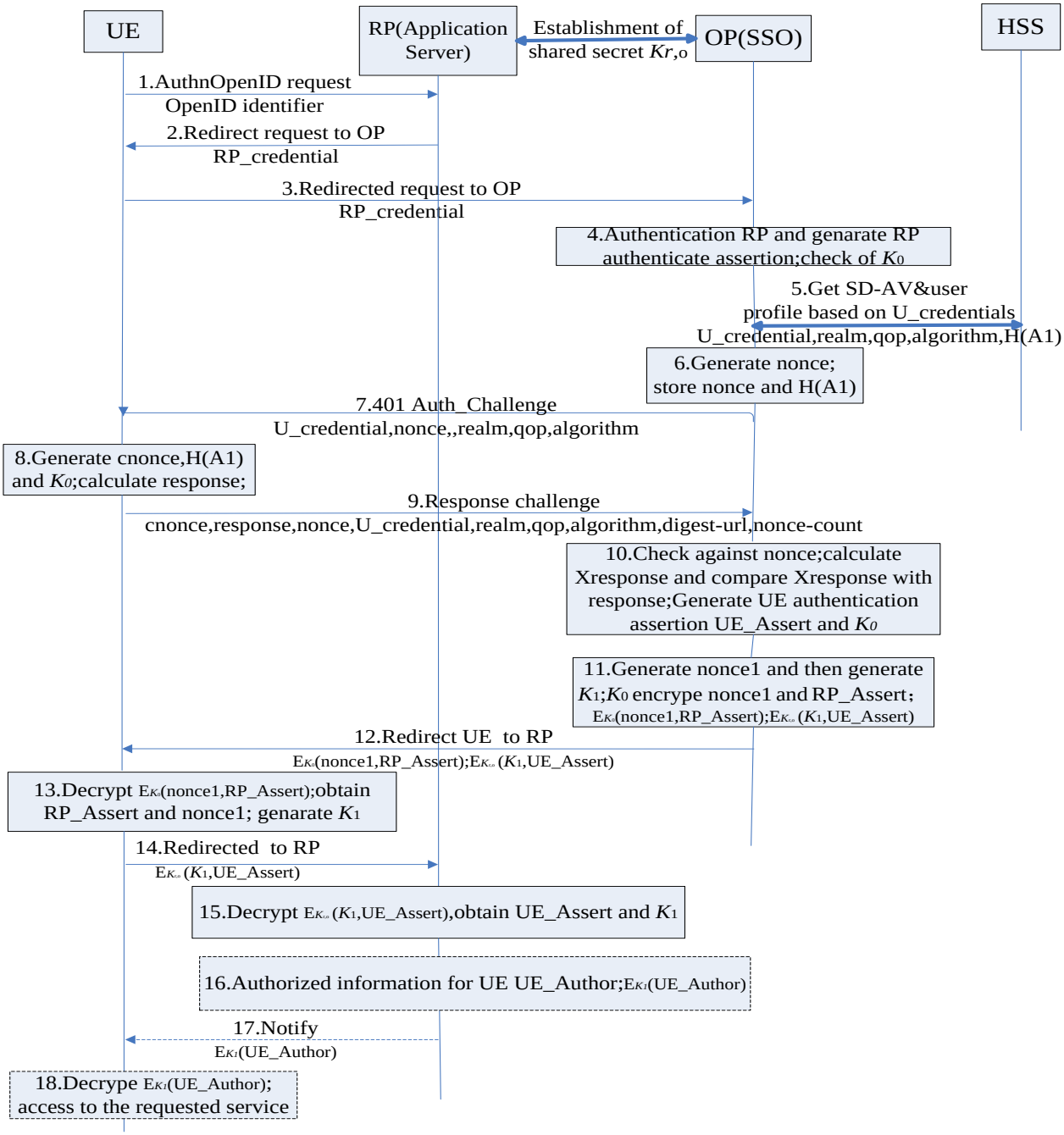


Figure 7.3-6 Interworking of SDALS with the OpenID that co-hosts IdP (SSO Server) and OP

The basic steps are as follows:

1. The UE issues an authentication request AuthnOpenID to the RP which includes an OpenID identifier.
2. The OpenID Identifier is normalized as described in Appendix A.1 of [14].The RP (Application server), using the presented OpenID identifier, discovers the URL of the OpenID identity provider OP, and redirects the user authentication request to that URL. The request includes the RP identifier (RP_credential).

3. The authentication request is redirected to the OpenID identity provider (OP). After this step the OP correlates the OpenID identifier with the UE identifiers.
4. The OP authenticates the RP based on the RP identifier. Assuming RP authentication success, the OP checks whether there is already a shared secret K_0 between the UE and the OP according to the OpenID identifier. If K_0 exists, the process jumps to step 11; otherwise, the process goes on to the next step.

   NOTE1: The RP and the OP shall have a shared secret ($K_{r,o}$) using existing mechanism, for example, using the Diffie-Hellman Key Exchange Protocol or pre-shared secret, the details of shared key establishment between the RP and OP are out of scope. With this shared secret the OP can sign subsequent messages and the RP can verify those messages.

   NOTE2: The OP is the sole decision point for RP's authenticity, and this means that any explicit messaging, e.g. to the UE, regarding the OP's decision on the authenticity of the RP, is redundant and unnecessary.

   NOTE3: There may be security concerns if this message (about OP notifying the UE about failure of OP authentication of the RP) is sent unprotected.

5. The OP sends authentication request to the HSS, then it obtains the SIP Digest authentication vector SD-AV and the user profile based on the U_credential from the HSS. The SD-AV consists of the qop (quality of protection) value, the authentication algorithm, realm, and a hash, called H (A1), of the U_credential, realm, and password. Refer to RFC 2617 [5] for additional information on the values in the authentication vector for SIP Digest based authentication. In a multiple HSS environment, the OP may have to obtain the address of the HSS where the UE is stored by querying the SLF.
6. The OP generates a random nonce, stores H(A1) and the nonce against the U_credential.
7. The OP sends a 401 Auth_Challenge to the UE which includes the nonce, the realm, qop, algorithm and U_credential.
8. Upon receiving the challenge, the UE generates a random cnonce and the H(A1), and then generates the shared secret K_0 based on the H(A1), the cnonce, etc. It then uses the cnonce as well as parameters provided in the 401 Auth_Challenge such as nonce, U_credential and qop to calculate an authentication response according to RFC 2617[5].
9. The UE sends a response to the OP which includes the cnonce, the nonce, the response, the realm, the U_credential, qop, algorithm, nonce-count and Digest-url.
10. Upon receiving the response, The OP uses the previously stored nonce to check against the nonce included in the response. If the check is successful, the OP calculates the expected response (Xresponse) using the previously stored H(A1) and the nonce together with other parameters contained in the response (e.g.cnonce, nonce-count, qop, as specified in RFC 2617[5]) and uses this to check against the response sent by the UE. If the check is successful the authentication of the UE is succeeded, else the authentication fails. The OP stores an authentication assertion (UE_Assert). If the UE is successfully authenticated, the OP generates the shared secret K_0 based on the H(A1), the cnonce, etc.
11. The OP generates a random nonce1 and generates a shared secret K_1 based on K_0, nonce1. The OP encrypts the nonce1 using K_0, i.e. E_{K_0}(nonce1); and encrypts the K_1 and UE_Assert using $K_{r,o}$, i.e. $E_{K_{r,o}}$ (K_1,UE_Assert).
12. The OP sends the UE an message including E_{K_0}(nonce1) and $E_{K_{r,o}}$ (K_1,UE_Assert) with redirection.
13. The UE decrypts the E_{K_0}(nonce1); and then obtains the nonce1; The UE will generates the shared secret K_1 based on K_0, nonce1.
14. The message sent by the OP is redirected to the RP including $E_{K_{r,o}}$ (K_1,UE_Assert).
15. The RP decrypts the $E_{K_{r,o}}$ (K_1,UE_Assert), and obtains UE_Assert and K_1.
16. After verifying the UE_Assert, the RP generates authorization information for the UE, i.e. UE_Author and encrypts UE_Author using K_1 E_{K_1}(UE_Author).
17. The RP notifies the UE about the authorization information.
18. The UE decrypts the E_{K_1}(UE_Author) and then accesses to the requested service.

   NOTE4: The last 3 steps 16, 17 and 18 are application specific, they are optional steps and not required for SSO authentication purpose.

If there is a failure in steps 1 through 18 – the authentication procedure stops.

NOTE 5: In the message flow of the authentication process above, steps 1 through 3 and steps 12 through 15 comply with OpenID specification, step 4 authenticates RP, which is similar to the association establishment process in OpenID specification, steps 5 through 11 belong to SIP Digest-based authentication, steps 16 through 18 provide application security.

Editor's Note: The aspects of providing keys for general application security between a terminal and application server, not only for interwoking with OpenID, should be also taken into account in the solution.

NOTE6: The interworking with the Liberty Alliance is similar to the interworking with the OpenID.

7.3.2.3 Message Flow with AS and OP co-hosting

The solution to utilize SIP Digest authentication for SSO can maximize commonality with the already defined 3GPP approaches for interworking with non-3GPP-defined SSO system as described in TR33.924 [9] and TR33.980 [8]. In the following a message flow is defined to allow the interworking with OpenID and the specific message flow is depicted as shown in Figure 7.3-7.

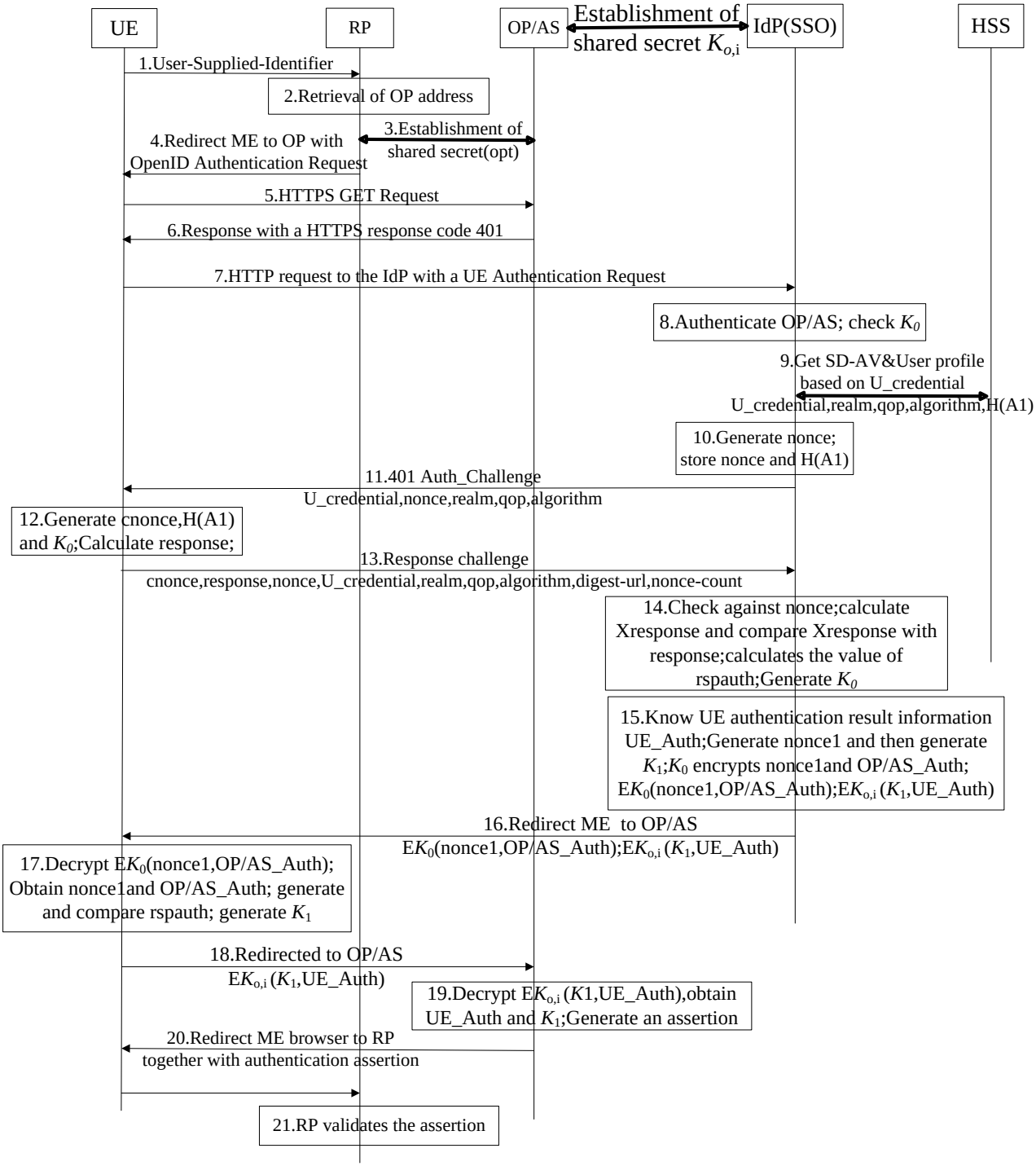


Figure 7.3-7 Interworking of SDALS with OpenID that co-hosts AS and OP

The basic message flow is as follows:

To initiate OpenID Authentication, the Relying Party should present the end user with a form that has a field for entering a User-Supplied Identifier. The form field's "name" attribute should have the value "openid_identifier".

1. The browser in the ME sends a User-Supplied Identifier to the Relying Party
2. The User-Supplied Identifier is normalized as described in Appendix A.1 of [14]. The RP retrieves the address of the OP and performs a discovery of the OP Endpoint URL (based on the User-Supplied Identifier) that the end user wishes to use for authentication.

3. The RP and the OP may then establish a shared secret (called association) using the Diffie-Hellman Key Exchange Protocol. The purpose of this shared secret is that the OP can sign subsequent messages and the RP can verify those messages.

   NOTE1: The Diffie-Hellman Key Exchange Protocol between the OP and the RP lies outside of the 3GPP specifications, this association using Diffie Hellman is an optional feature and not required for interworking purposes. If the OP and RP do not both reside under the control of the same MNO, the usage of this option seems strongly advisable。

4. The RP redirects the ME's browser to the OP with an OpenID Authentication Request as defined in chapter 9 in [14]. The RP inserts into the openid.claimed_id and into the openid.identity fields the user supplied identifier of step 1.

5. Following this redirection the ME sends a HTTPS GET request to the OP.

6. The OP/AS initiates the ME authentication and responds with a HTTPS response code 401 "Unauthorized", which contains a WWW Authenticate header carrying a challenge requesting the UE to use SIP Digest Authentication with SSO_APS. The response message also includes the OP/AS credential (OP/AS_credential).

   NOTE2: The OP/AS and the IdP shall have a shared secret (K_{0i}) using existing mechanism, for example, using the Diffie-Hellman Key Exchange Protocol or pre-shared secret, the details of shared key establishment between the OP/AS and IdP are out of scope.

7. If no valid K_0 is available, then the ME sends a second HTTP request to the IdP with a UE Authentication Request carrying the UE credential (U_credential) and OP/AS_credential.

8. The IdP obtains the OP/AS_credential; The IdP authenticates the RP based on the OP/AS_credential; then generates and stores related authentication result OP/AS_Auth. According to the U_credential, the IdP first checks whether there is already a shared secret K_0 between the UE and IdP. If K_0 exists, the process jumps to step 15; otherwise, the process goes on to the next step.

9. The IdP sends authentication request to the HSS, it then obtains the SIP Digest authentication vector SD-AV and the user profile based on the U_credential from the HSS. The SD-AV consists of the qop (quality of protection) value, the authentication algorithm, realm, and a hash, called H (A1), of the U_credential, realm, and password. Refer to RFC 2617[5] for additional information on the values in the authentication vector for SIP Digest based authentication. In a multiple HSS environment, the IdP may have to obtain the address of the HSS where the UE is stored by querying the SLF.

10. The IdP generates a random nonce, stores H(A1) and the nonce against the U_credential.

11. The IdP sends a 401 Auth_Challenge to the UE which includes the nonce, the realm, qop, algorithm and U_credential.

12. Upon receiving the challenge, the UE generates a random cnonce and the H(A1), and then generates the shared secret K_0 based on the H(A1), the cnonce, etc. It then uses the cnonce as well as parameters provided in the 401 Auth_Challenge such as nonce, U_credential and qop to calculate an authentication response according to RFC 2617[5].

13. The UE sends a response to the IdP which includes the cnonce, the nonce, the response, the realm, the U_credential, qop, algorithm, nonce-count and digest-url.

14. Upon receiving the response, the IdP uses the previously stored nonce to check against the nonce included in the response. If the check is successful, the IdP calculates the expected response (Xresponse) using the previously stored H (A1) and the nonce together with other parameters contained in the response (e.g.cnonce, nonce-count, qop, as specified in RFC 2617[5]) and uses this to check against the response sent by the UE. If the check is successful, the authentication of the UE is succeeded, else the authentication fails. If the UE is successfully authenticated, the IdP calculates the value of rspauth based on SIP Digest as specified in RFC 2617 [5] and generates the shared secret K_0 based on the H(A1), the cnonce, etc.

15. The IdP knows the User authentication conclusion (UE_Auth); and then the IdP generates a random nonce1 and generates a shared secret K_1 based on K_0 and nonce1. The IdP encrypts the nonce1 and OP/AS_Auth using K_0, i.e. E_{K_0}(nonce1,OP/AS_Auth); and encrypts the K_1 and UE_Auth using $K_{0,i}$, i.e. $E_{K_{0i}}$ (K_1,UE_Auth).

16. The IdP sends the UE an message including E_{K_0}(nonce1,OP/AS_Auth), $E_{K_{0i}}$ (K_1,UE_Auth) ,and the value of rspauth with redirection.

17. The UE decrypts the E_{K_0}(nonce1,OP/AS_Auth) and then obtains OP/AS_Auth and nonce1. Based on the OP/AS_Auth the UE knows the legitimacy of the requested OP/AS. If the authentication result indicates that the OP/AS is not valid, the UE will stop visiting the OP/AS. The UE calculates the rspauth in the same way as the IdP

did in step 14, and uses it to check against the rspauth sent by the IdP. If the check is successful, the authentication of the Network is succeeded, else the authentication fails. If the Network is successfully authenticated, and then the UE will generates the shared secret K_1 based on K_0, nonce1.

18. The message sent by the IdP is redirected to the OP/AS including $E_{K_{o,i}}(K_1, \text{UE\_Auth})$.
19. The OP/AS decrypts the $E_{K_{o,i}}(K_1, \text{UE\_Auth})$, and obtains UE_Auth and K_1. The OP/AS establishes whether the end user is authorized to perform OpenID Authentication and wishes to do so based on the authorization information stored in the UE_Auth. In particular, the UE_Auth may contain information about the type of information which is allowed to be shared with the RP. The OP/AS authenticates the user of OpenID using SSOa reference point, and generates an assertion based on the authorization information.
20. The OP/AS redirects the browser to the return OpenID address i.e. the OP/AS redirects the ME's browser back to the RP with either an assertion that authentication is approved or a message that authentication failed. The response header contains a number of fields defining the authentication assertion which may be protected by the shared secret between OP/AS and RP. The protection is especially important if the OP/AS and the RP do not reside both in the same MNO network.
21. The RP validates the assertion i.e. checks if the authentication was approved. The authenticated identity of the user is provided in the response message towards the RP. If a shared secret was established in step 3, then it is now used to verify the message from the OP/AS. If the validation of the assertion and the verification of the message are successful, then the user is logged in to the service of the RP.

If there is a failure in steps 1 through 21 – the authentication procedure stops.

7.3.2.4 Solution 2 (SDALS) – Improvements with RP authentication for IdP (SSO Server) and OP co-hosting case

Editor's note: It is FFS whether this solution with RP authentication is in scope of 3GPP's responsibility.

Achieving RP authentication at the OP

The protocol depicted in Figure 7.3-6 does not show how RP authentication can be achieved. Particularly, it is not clear whether and how RP uses any secret in the steps 1-4 towards the OP. In order to achieve RP authentication, additional steps need to be considered. If we assume that the OP and RP already share a secret Kr,o this secret has to be used for RP authentication with the OP. Not shown in the protocol figure 7.3-6 are the association creation steps of the OpenID protocol.

Note that the inclusion of RP authentication in the OpenID protocol requires changes to the OpenID protocol itself as well as to the implementations of the OP and RP. Thus, any such change should be carefully weighed for benefits versus the cost. Moreover, it is desired that any such RP authentication method should NOT affect implementations on the UE.

In general RP authentication will consist in a challenge response step between OP and RP, where the OP has to send a challenge with a proof of freshness to the RP, e.g. via a nonce. The RP can then use the pre-established shared secret Kr,o to sign this nonce and return the answer to the OP. The response to the authentication challenge can be as a direct response to the OP authentication challenge, or can be integrated in the redirect message, which sends the UE to the OP. In either case it needs to be ensured that the OP can have reliable evidence on the authentication of the RP before engaging in actual UE authentication. This allows to stop the protocol in the case of a failed RP authentication, and saves communication effort between UE and OP in the case of such a failed RP authentication. The OP can then directly convey the information on the failed RP authentication to the UE.

Using Associations for RP authentication

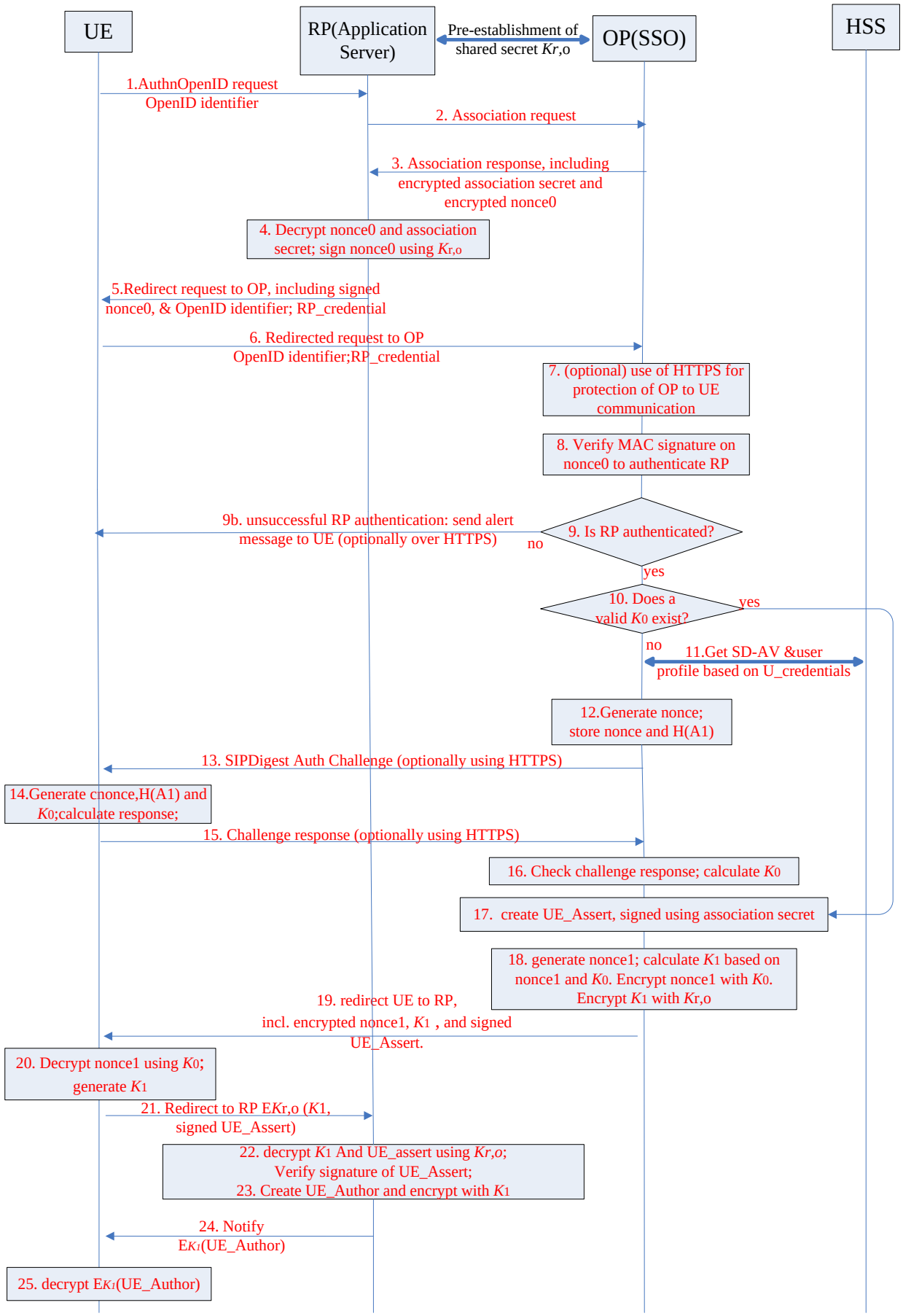
If the RP establishes an association with the OP, the OpenID association protocol can be slightly modified to incorporate a challenge from the OP to be conveyed to the RP for the purpose of RP authentication.

During the association establishment the OP and RP can set up a message authentication code (MAC) key which is later used to sign the assertion message. This key is sent encrypted using a temporary secret key which is negotiated between OP and RP using Diffie-Hellman (DH). In addition to that key, deviating from the OpenID specs, the OP can include a nonce, also encrypted with the DH-key, in the response to the RP.

The RP can then decrypt the nonce and the MAC key based on the negotiated DH-key. The RP then uses its own pre-established Kr,o key to sign (or encrypt) the nonce as received from the OP and adds it as an additional parameter to the redirect message which is sent to the UE. Since the UE follows the redirect to the OP, the OP receives the signed (or encrypted) nonce, and can use the shared key Kr,o to authenticate the RP. In the case of a failed authentication, the OP can send an alert message to the UE to protect it from unauthenticated RPs. In the case of a successful RP authentication, the OP can proceed with the protocol.

Simplified Variant Protocol with RP authentication and HTTPS protection

The following Figure 7.3-8 depicts a revised protocol where the RP is authenticated by the OP using a modified OpenID association messaging


UE
RP(Application Server)
Pre-establishment of shared secret Kr,o
OP(SSO)
HSS
1.AuthnOpenID request OpenID identifier
2. Association request
3. Association response, including encrypted association secret and encrypted nonce0
4. Decrypt nonce0 and association secret; sign nonce0 using Kr,o
5.Redirect request to OP, including signed nonce0, & OpenID identifier; RP_credential
6. Redirected request to OP OpenID identifier;RP_credential
7. (optional) use of HTTPS for protection of OP to UE communication
8. Verify MAC signature on nonce0 to authenticate RP
9. Is RP authenticated?
9b. unsuccessful RP authentication: send alert message to UE (optionally over HTTPS)
no
yes
10. Does a valid K0 exist?
yes
no
11.Get SD-AV &user profile based on U_credentials
12.Generate nonce; store nonce and H(A1)
13. SIPDigest Auth Challenge (optionally using HTTPS)
14.Generate cnonce,H(A1) and K0;calculate response;
15. Challenge response (optionally using HTTPS)
16. Check challenge response; calculate K0
17. create UE_Assert, signed using association secret
18. generate nonce1; calculate K1 based on nonce1 and K0. Encrypt nonce1 with K0. Encrypt K1 with Kr,o
19. redirect UE to RP, incl. encrypted nonce1, K1 , and signed UE_Assert.
20. Decrypt nonce1 using K0; generate K1
21. Redirect to RP EKr,o (K1, signed UE_Assert)
22. decrypt K1 And UE_assert using Kr,o; Verify signature of UE_Assert;
23. Create UE_Author and encrypt with K1
24. Notify EK1(UE_Author)
25. decrypt EK1(UE_Author)

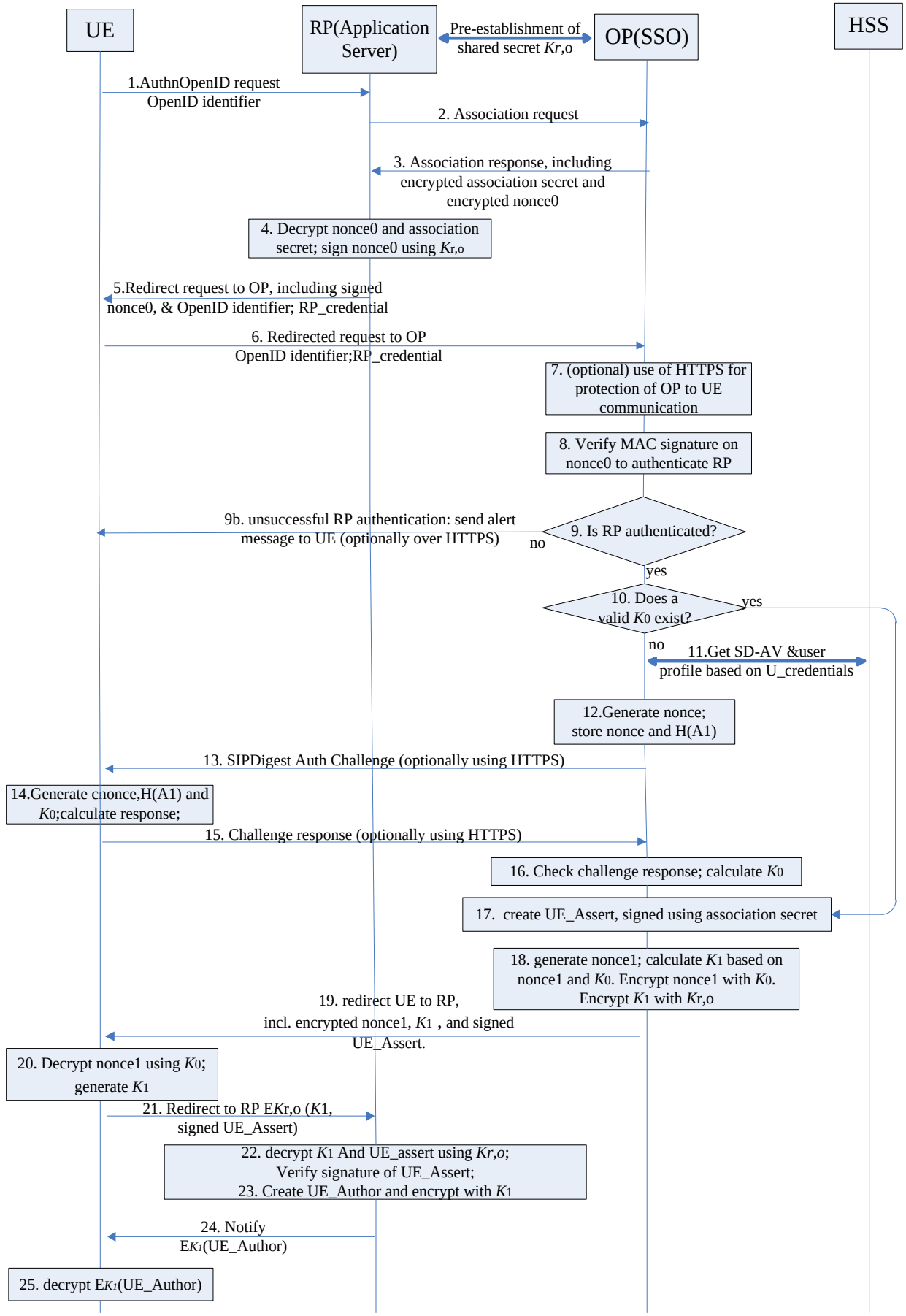


Figure 7.3-8 Interworking of SDALS with IdP (SSO Server) and OP co-hosting, with RP authentication

The steps of this variant protocol are as follows:

1. The UE issues an authentication request AuthnOpenID to the RP which includes an OpenID identifier.
2. The RP sends an (OpenID) association request to the OP. The RP and the OP then establish a Diffie-Hellman Key D-H, according to Section 8 of [14]. OP generates an association secret and association handle (together called association according to [14]).
3. The OP sends RP an association response according to [14], Section 8.2, including association secret and nonce0, both encrypted with the D-H key established in step 2.
4. The RP decrypts the received encrypted nonce0 and encrypted association secret, and signs nonce0 with *Kr,o*.

   NOTE1: HMAC or another suitable symmetrical signature algorithm may be used to sign nonce0

   NOTE2: The RP and the OP shall have a shared secret ($K_{r,o}$) using existing mechanism, for example, using the Diffie-Hellman Key Exchange Protocol or pre-shared secret, the details of shared key establishment between the RP and OP are out of scope. With this shared secret the OP can sign subsequent messages and the RP can verify those messages.
5. The RP redirects the authentication request sent by the UE which includes the OpenID identifier, the RP identifier (RP_credential), and the signed nonce0.
6. The authentication request is redirected to the OP. The redirection includes OpenID identifier and RP_credential.
7. (Optionally) The OP may set up TLS and initiates HTTPS communiation with UE. This could be done by configuration of the OP web server.
8. The OP verifies the signature of nonce0 to authenticate the RP.
9. If RP authentication of step 9 failed, the OP sends an alert message 10b, (optionally protected by HTTPS if step 7 is taken), to indicate RP authentication failure, to the UE. If RP authentication in step 9 succeeded, proceed to step 11.
10. The OP determines if a valid key K_0 exists. If the answer is no, proceed to step 11. If yes, proceed to step 17.
11. The OP sends authentication request to the HSS, then it obtains the SIP Digest authentication vector SD-AV and the user profile based on the U_credential from the HSS. The SD-AV consists of the qop (quality of protection) value, the authentication algorithm, realm, and a hash, called H (A1), of the U_credential, realm, and password. Refer to RFC 2617 [5] for additional information on the values in the authentication vector for SIP Digest based authentication. In a multiple HSS environment, the OP may have to obtain the address of the HSS where the UE is stored by querying the SLF.
12. The OP generates a random nonce, stores H(A1) and the nonce against the U_credential.
13. The OP sends, optionally protected by HTTPS, a 401 Auth_Challenge (as a SIP-Digest Auth challenge) to the UE which includes the nonce, the realm, qop, algorithm and U_credential.
14. Upon receiving the challenge, the UE generates a random cnonce and the H(A1), and then generates the shared secret K_0 based on the H(A1), the cnonce, etc. It then uses the cnonce as well as parameters provided in the 401 Auth_Challenge such as nonce, U_credential and qop to calculate an authentication response according to RFC 2617[5].
15. The UE sends a challenge response, optionally protected by HTTPS, to the OP which includes the cnonce, the nonce, the response, the realm, the U_credential, qop, algorithm, nonce-count and Digest-url.
16. Upon receiving the response, The OP uses the previously stored nonce to check against the nonce included in the response. If the check is successful, the OP calculates the expected response (Xresponse) using the previously stored H(A1) and the nonce together with other parameters contained in the response (e.g.cnonce, nonce-count, qop, as specified in RFC 2617[5]) and uses this to check against the response sent by the UE. If the check is successful the authentication of the UE is succeeded, else the authentication fails. If the UE is successfully authenticated, the OP generates the shared secret K_0 based on the H(A1), the cnonce, etc.
17. The OP creates an authentication assertion (UE_Assert) signed using the association secret (from step 3 above).
18. The OP generates a random nonce1 and generates a shared secret K_1 based on K_0, nonce1. The OP encrypts the nonce1 using K_0, i.e. E_{K_0}(nonce1); and encrypts the K_1 and the signed UE_Assert using $K_{r,o}$, i.e. $E_{K_{r,o}}$ (K_1, signed(UE_Assert)).
19. The OP sends the UE an message including E_{K_0}(nonce1) and $E_{K_{r,o}}$ (K_1, signed(UE_Assert)) with redirection to RP.
20. The UE decrypts the E_{K_0}(nonce1); and then obtains the nonce1; The UE will generates the shared secret K_1 based on K_0, nonce1.

21. The message sent by the OP is redirected to the RP including E_{K_0} (K_1,signed UE_Assert).

22. The RP decrypts the E_{K_0} (K_1, signed UE_Assert), and obtains UE_Assert and K_1. RP verifies signature of UE_Assert using the association secret shared with OP.

23. After verifying the UE_Assert, the RP generates an authorization information for the UE, i.e. UE_Author and encrypt UE_Author using K_1 E_{K_1}(UE_Author).

24. The RP notifies the UE about the authorization information, which is contained in this message, encrypted with K_1.

25. The UE decrypts the E_{K_1}(UE_Author) and then accesses to the requested service.

7.3.3 Solution 2 SDLAS - Evaluation

In the variant described in 7.3.2.1 the OpenID Identity Provider server (figure 7.3-6) has to support the following 3GPP specific functionalities:

- Establishment of a shared secret between RP and OP
- Validation of K0
- Interface to HSS to obtain the SD-AV, user profile and related attributes
- Handling of User credentials in addition to OpenID identifier
- Storage and handling of data related to U_credential (nonce, H(A1))
- Extensive generation and handling of nonces (nonce, nonce1, cnonce) compared to OpenID which only uses one nonce to prevent replay attacks against the RP

In addition the RP needs to support non-OpenID specific extensions with regard to the incoming redirection (step 14, figure 7.3-6), where the assertion and a key are together encrypted.

If solution 2 is used in conjunction with Liberty Alliance as outlined in 7.3.2.1 in the text above figure 7.3-7, then also 3GPP specific extensions to the IdP and the service provider similar to the one above for OpenID are needed (figure 7.3-7 where the IdP is a Liberty Alliance compliant IdP). The variant of solution 2 in 7.3.2.3 improves the RP security by adding authentication and usage of associations. This improvement is 3GPP specific and not part of OpenID and requires additional functionality from the RP and OP.

For all variants of solution 2 the OpenID is collocated with the IdP and Liberty Alliance identity management specifications and their extensions are not part of the 3GPP specification family. All variants of solution 2 introduce some extensions to the non-3GPP specific protocols and either they create a 3GPP specific extension which non 3GPP nodes and services have to follow or they create a dependency to an external standardization body with a different release schedule.

7.4 The Use of Protocol Binding for SIP Digest Over TLS to Prevent MitM Attacks

The solution described in this section assumes the same architectural framework given in figure 7.2-1. The new solution improves upon the solution depicted in Figure 7.2-1 in two aspects:

1. The solution uses TLS as an underlying security protocol for SIP Digest, providing basic protection against simple MitM attacker.
2. The solution further uses binding between the TLS tunnel and the SIP digest protocol, thereby providing protection against man-in-the-middle (MitM) attackers by preventing such an attacker from running a TLS session with the BSF and interacting with the UE via HTTP outside of the TLS tunnel.

The new solution, which is a non-UICC based (SIP Digest) protocol on top of a TLS tunnel, employs a generic form of GBA based on the solution provided in TS 33.220. The binding solution makes explicit use of the shared SIP Digest credentials between the network and the UE. As a result of the binding between the TLS and SIP-Digest, this solution prevents a MitM attacker from taking advantage of the dual authentication aspects of the protocol. The following Figure 7.4-1 and the step-wise description below the figure explain the protocol flow of the new solution.

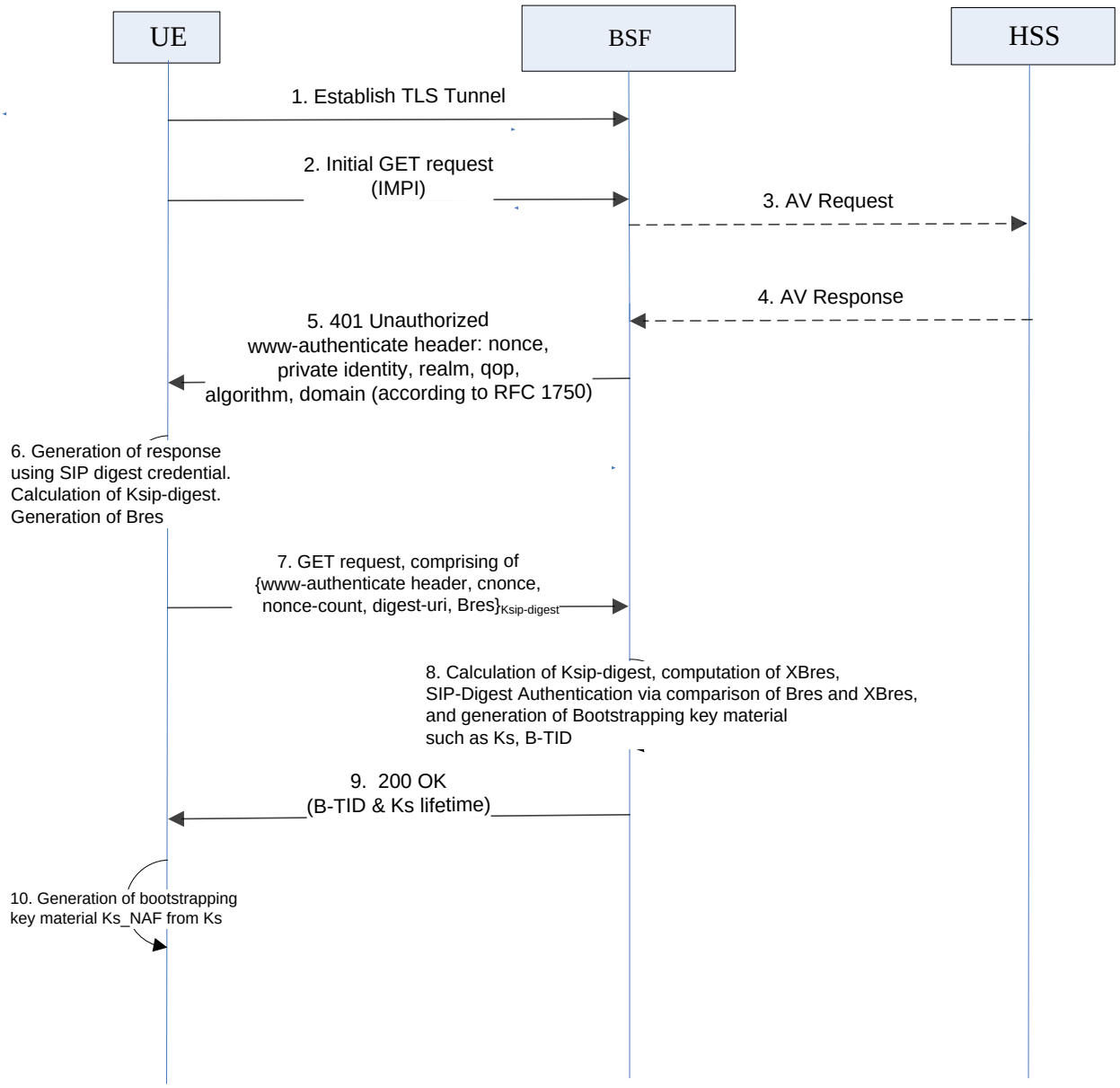


Figure 7.4-1 procedure of non-UICC based GBA solution, using TLS and binding to SIP-Digest

The proposed binding solution is illustrated in Fig. 7.4-1 above and described as follows:

1. The UE starts the bootstrapping procedure by initiating a TLS session with the BSF. The UE and BSF. The UE authenticates the BSF by the certificate presented by the BSF. The BSF does not require authentication from the UE at this point.

2. Following the establishment of the TLS tunnel the UE sends an HTTP Request message (which includes the IMPI) to the BSF.

3. The BSF requests an authentication vector (AV) from the HSS.

4. The HSS sends the AV to the BSF.

5. The BSF sends the UE an authentication challenge in a HTTP 401 Unauthorized response. This challenge includes a www-authenticate header and a randomly generated nonce.

6. The UE calculates a cnonce and generates an HTTP request message which is calculated according RFC 2617. While calculating this response, the UE also generates a SIP-Digest response session key, denoted Ksip-digest, from the SIP Digest credential, H(A1), and cnonce. The UE then generates a message authentication code (MAC) value

Bres using both the TLS tunnel session key and Ksip-digest as inputs for the MAC algorithm. This binds the TLS tunnel authentication with the SIP-Digest authentication within GBA.

Editor's note: Definition and derivation of Ksip-digest is needed.

7. Both the SIP-Digest authentication challenge response and Bres are sent back to the BSF as part of the Authorization header in a GET request message. The Get request message is integrity protected using the key Ksip-digest.

   Editor's Note: The solution proposes to use the same Digest Response as used in IMS registration. A solution for protocol separation is ffs.

   NOTE1: Bres may be calculated by the same algorithm as the authentication response, however with different input parameters.

8. The BSF receives the GET request which contains the authentication challenge response and checks that the expected response, i.e., the integrity value which is to be calculated by the BSF, matches the integrity value of the received challenge response. The BSF calculates Ksip-digest from the SIP-Digest credential H(A1) it already has from the AV and the cnonce received from the GET request. The BSF then checks Bres against its own, computed expected value (called XBres). It can do so because it now knows both keys (Ksip-digest and the TLS tunnel session key) used in the calculation of Bres. If the received Bres matches XBres, and the received authentication challenge response integrity value matches its expected integrity value the BSF determines that the UE has been authenticated and also assures itself that, because of the binding effect verified from the matching of the Bres and XBres, the entity (UE) that was authenticated in the formation of the tunnel is the same entity it (the BSF) just authenticated in the GBA aspect of the protocol. If these checks are successful, the user is considered authenticated and the user's private user identity is registered to the BSF.

   Editor's note: The consistency of this step with the lack of authentication step in Step 1 needs to be explained.

   Editor's note: The channel binding requires a new RFC.

   Editor's note: Channel binding for the UE to verify the BSF is FFS.

   Editor's note: Protocol separation from ordinary IMS usage is FFS.

9. The BSF indicates to the UE the successful authentication of the user by sending the UE a 200 OK response which includes B-TID, Ks_lifetime, and rspauth (generated according to RFC 2617).

Annex A:
Use of the Key Derivation Function

A.1 Derivation of passwd and Ks

Derivation of passwd and Ks for GBA_Digest follows the same procedure as NAF specific key derivation in GBA and GBA_U (TS 33.220, Annex B.3).

The input parameters for the key derivation function to derive passwd and Ks shall be the following:

- FC = 0x01,
- P1 = TLS_MK_Extr,
- L1 = length of TLS_MK_Extr is 48 octets (i.e. 0x00 0x30),

In the derivation of passwd as specified in Clause 7.2.2, Step 5,

- P0 = "GBA_Digest_RESP"
  (i.e. 0x47 0x42 0x41 0x5F 0x44 0x69 0x67 0x65 0x73 0x74 0x5F 0x52 0x45 0x53 0x50), and
- L0 = length of P0 is 15 octets (i.e., 0x00 0x0F).

In the key derivation of Ks as specified in Clause 7.2.2, Step 6,

- P0 = "GBA_Digest_Ks"
  (i.e. 0x47 0x42 0x41 0x5F 0x44 0x69 0x67 0x65 0x73 0x74 0x5F 0x4B 0x73),
- L0 = length of P0 is 13 octets (i.e., 0x00 0x0D),
- P2 = RESP, and
- L2 = length of RESP is variable and depends on the algorithm used in HTTP Digest (e.g., 32 if MD5 is used)

The Key to be used in the key derivation function shall be:

- H(A1) as specified in Clause 7.2.2, step 5

NOTE: In the specification this function is denoted as:
passwd = KDF (H(A1), "GBA_Digest_RESP", TLS_MK_Extr), and
Ks = KDF (H(A1), " GBA_Digest_Ks", TLS_MK_Extr, RESP).

A.2 NAF specific key derivation in GBA_Digest

In GBA_Digest, the input parameters for the key derivation function to derive Ks_NAF shall be the following:

- FC = 0x01,
- P0 = "gba-digest" (i.e. 0x67 0x62 0x61 0x2d 0x64 0x69 0x67 0x65 0x73 0x74), and
- L0 = length of P0 is 10 octets (i.e., 0x00 0x0a).
- P1 = nonce,
- L1 = length of nonce is variable (not greater than 65535),
- P2 = IMPI encoded to an octet string using UTF-8 encoding (see clause B.2.1 of TS 33.220 [2]),
- L2 = length of IMPI is variable (not greater than 65535),
- P3 = NAF_ID with the FQDN part of the NAF_ID encoded to an octet string using UTF-8 encoding (see clause B.2.1 of TS 33.220 [2]), and

- L3 = length of NAF_ID is variable (not greater that 65535).

The Key to be used in key derivation shall be:

- Ks as specified in clause A.1 of the present document,

NOTE: In clause 7.2.2 this function is denoted as:

Ks_NAF = KDF (Ks, "gba-digest", nonce, IMPI, NAF_Id).

Annex B:
Change history

Change history							
Date	**TSG #**	**TSG Doc.**	**CR**	**Rev**	**Subject/Comment**	**Old**	**New**
2011-05	SA-52	SP-110255	--	--	Presentation to SA for Information	---	1.0.0
2012-02	SA-55	SP-120032	--	--	Presentation to SA for Approval	1.1.0	2.0.0