Recommendation ITU-T Y.2722

NGN Identity Management Mechanisms

Keywords

Identity management, identity management mechanisms, next generation network, security, and federated identity.

Summary

This Recommendation specifies the mechanisms that can be used to meet the Identity Management (IdM) requirements and deployment needs of Next Generation Networks (NGN).

CONTENTS

[This page intentionally left blank for this distribution version]

1 Scope

ITU-T Recommendation Y.2721, *NGN Identity Management Requirements and Use Cases* [ITU-T Y.2721], specifies identity management (IdM) requirements for the Next Generation Network (NGN). This Recommendation describes the specific IdM mechanisms and suites of options that should be used to meet the requirements specified in [ITU-T Y.2721]. In addition, it provides best practices and guidelines to support interoperability and other needs.

This Recommendation is intended to be used together with [Y.2720] and [ITU-T Y.2721] as the fundamental architectural concepts, requirements and use cases are not repeated in this Recommendation.

Note: Implementers and users of the described mechanisms shall comply with all applicable national and regional laws, regulations and policies. Some specific regulation and legislation may require implementation of mechanisms to protect personally-identifiable information.

2 References

The following ITU-T Recommendations and other references contain provisions, which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All Recommendations and other references are subject to revision; all users of this Recommendation are therefore encouraged to investigate the possibility of applying the most recent edition of the Recommendations and other references listed below. A list of the currently valid ITU-T Recommendations is regularly published.

[ATIS 33102] ATIS.3GPP.33.102V710-2007, *Security Architecture*

[ITU-T X.509] ITU-T Recommendation X.509 (2008), Information technology – Open systems interconnection – The Directory: Public-key and attribute certificate frameworks

[ITU-T X.1252] ITU-T Recommendation X.1252 (2010), Baseline identity management terms and definitions

[ITU-T Y.2701] ITU-T Recommendation Y.2701 (2007), Security requirements for NGN release 1

[ITU-T Y.2720] ITU-T Recommendation Y.2720 (2009), NGN Identity Management Framework.

[ITU-T Y.2721] ITU-T Recommendation Y.2721 (2010), NGN Identity Management Requirements and use cases

[ITU-T Y.2704] ITU-T Recommendation Y.2704 (2010), Security mechanisms and procedures for NGN

[ITU-T Y.2702] ITU-T Recommendation Y.2702 (2008), Authentication and authorization requirements for NGN release 1

[ITU-T Y.2012] Recommendation Y.2012, Functional Requirements and Architecture of the NGN of Release 1, 09/2006

[ITU-T X.1141] ITU-T Recommendation X.1141 (2006), Security Assertion Markup Language (SAML 2.0)

3 Definitions

This Recommendation relies on the terms defined in [ITU-T Y.2720] and [ITU-T X.1252].

Particularly, the following definitions are adopted from [ITU-T X.1252]:

3.1 identity provider (IdP): See identity service provider (IdSP)

3.2 identity service provider (IdSP): An entity that verifies, maintains, manages, and may create and assign identity information of other entities.

4 Abbreviations

This Recommendation uses the following abbreviations and acronyms:

AKA Authentication and Key Agreement
ASP Application Service Provider
AV Authentication Vector
BSF Bootstrapping Server Function
CK Ciphering Key
GBA Generic Bootstrapping Architecture
HSS Home Subscriber System
IdM Identity Management
IdP Identity Provider
IdSP Identity Service Provider
IK Integrity Key
IMPI IP Multimedia Private user Identity
IMPU IP Multimedia Public User identity
IMS IP Multimedia Subsystem
IMSI International Mobile Subscriber Identity
IPTV Internet Protocol Television
ISIM IMS Subscriber Identity Module
LDAP Lightweight Directory Access Protocol
NAF Network Application Function
NGN Next Generation Networks
OASIS Organization for the Advancement of Structured Information Standards
OTP One Time Password
PII Personally Identifiable Information (PII)
PKI Public Key Infrastructure
SAML Security Assertion Markup Language
SIP Session Initiation Protocol
SLF Subscriber Locator Function
SOAP Simple Object Access Protocol
SQL Structured Query Language
SSO Single Sign-On
UE User Equipment
UICC Universal Integrated Circuit Card
UMTS Universal Mobile Telecommunications System

WSS Web Services Security

XML eXtensible Markup Language

5 Conventions

In this document:

The keywords "**is required to"** indicate a requirement which must be strictly followed and from which no deviation is permitted if conformance to this document is to be claimed.

The keywords "**is recommended"** indicate a requirement which is recommended but which is not absolutely required. Thus this requirement need not be present to claim conformance.

The keywords "**is prohibited from"** indicate a requirement which must be strictly followed and from which no deviation is permitted if conformance to this document is to be claimed.

The keywords "**can optionally"** indicate an optional requirement which is permissible, without implying any sense of being recommended. This term is not intended to imply that the vendor's implementation must provide the option and the feature can be optionally enabled by the network operator/service provider. Rather, it means the vendor may optionally provide the feature and still claim conformance with the specification.

In the body of this document and its annexes, the words *shall*, *shall not*, *should*, and *may* sometimes appear, in which case they are to be interpreted, respectively, as *is required to*, *is prohibited from*, *is recommended*, and *can optionally*. The appearance of such phrases or keywords in an appendix or in material explicitly marked as *informative* are to be interpreted as having no normative intent.

6 Mechanisms and Procedures supporting IdM Functions

6.1 Lifecycle Management

Refer to ITU-T Recommendation Y.2720, *NGN Identity Management Framework* for information on identity lifecycle management.

6.2 Authentication and Authentication Assurance

This clause describes mechanisms for authentication and assurance of identities and identity information. It references authentication mechanisms defined elsewhere.

Specific authentication mechanisms such as authentication based on Web Services (WS) Security Assertion Markup Language (SAML) Profile, Certificate-based authentication, or Password-based authentication (including One Time Password (OTP)) can be used by the IdSP for specific applications or services based on context and on the needed level of assurance. The authentication method (or methods) is selected based on the assurance level requirements. The IdSP may request information to determine the authentication methods that satisfy the service provider's assurance level requirements.

6.2.1 Authentication based on WS Security SAML Profile

6.2.1.1 SAML assertions

Security Assertion Markup Language (SAML) [ITU-T X.1141] specifies format of assertions that can be used for exchanging security information for IdM. Among the IdM functions that can be implemented with the use of SAML are authentication, attribute sharing, and authorization, which correspond to three types of the statements about a subject of a SAML assertion:

- Authentication statement – conveys information that the assertion subject was authenticated by a particular means at a particular time.

- Attribute statement – conveys information that the assertion subject is associated with the listed attributes.
- Authorization decision statement – conveys information that access to a specified resource was granted to the assertion subject, or the subject was denied such access.

The content of a SAML assertion can be described at a high level as follows: assertion **A** was issued at time **t** by issuer **R** regarding subject **S** provided conditions **C** are valid.

The SAML assertions used for communicating authentication, attribute and authorization information in are conveyed in Simple Object Access Protocol (SOAP) messages. When SOAP messages are exchanged over an unprotected transport, it is strongly recommended that XML signature [b-W3C XML signature] be used to verify the relationship between the SOAP message and that the statements of the assertions carried in the message. The *Web Services Security (WSS): SAML Token Profile* [b-OASIS SAML token] standard describes how:

- SAML assertions (also referred to as SAML tokens) are carried in and referenced from a SOAP message.
- XML signature is used to bind a subject and the statements of a SAML assertion with a SOAP message.

A typical use of a SAML token with SOAP message constructed according to this Recommendation is depicted by Figure 1 and described below.

In this example, a signed SOAP message contains a SAML assertion with an attribute statement. Based on the information in this statement, the receiver could make access control decisions.

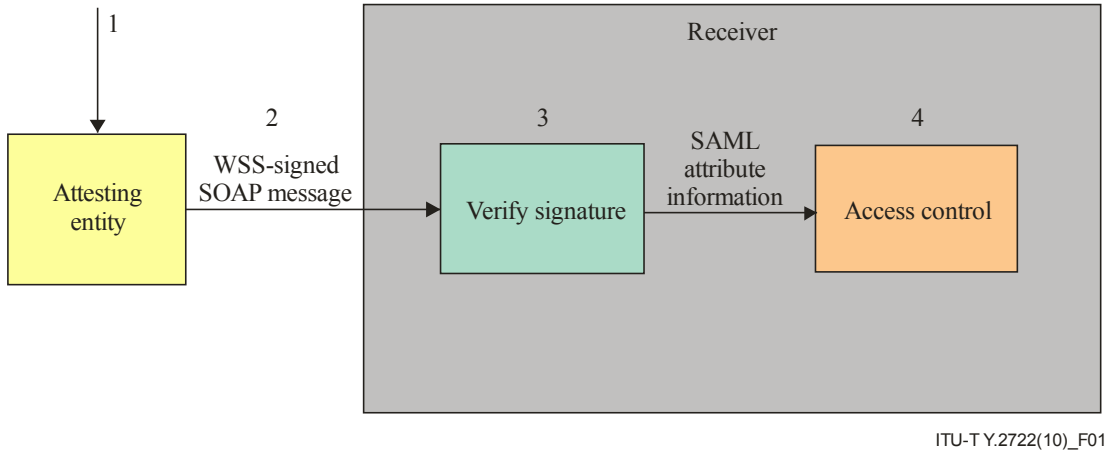


Figure 1 - Typical steps of construction and processing of a SOAP message with a SAML token

1. The Attesting Entity obtains a SAML assertion with an attribute statement, constructs and includes it in a SOAP message according to [b-OASIS SAML token].
2. The Attesting Entity sends the WSS-signed SOAP message to the Receiver.
3. The Receiver verifies the digital signature.
4. The information of the SAML statement may be used for access control decisions.

6.2.1.2 Subject confirmation methods of the SAML tokens

The OASIS Standard, *Web Services Security: SAML Token Profile 1.1* [b-OASIS SAML token] specifies how to attach a SAML assertion to a SOAP message and defines two mandatory subject confirmation methods:

- Holder-of-key
- Sender-vouches

The main XML elements of the SOAP message constructed according to [b-OASIS WSS SOAP] are depicted in Figure 2.

The SAML assertion is placed into the <wsse:Security> header, which also contains the digital signature <ds:Signature>. The digital signature is used by the receiver of the SOAP message to verify that the sender of the message knows the key used for computing the signature over the digest of the SOAP body and for checking its integrity. The digest algorithm is SHA 1 and the signature algorithm is RSA_SHA 1 as specified in [b-OASIS WSS SOAP]. The signature's value is provided in the <ds:SignatureValue> element of the digital signature <ds:Signature>.

The two subject confirmation methods define different ways for conveying information on the key to the receiver.

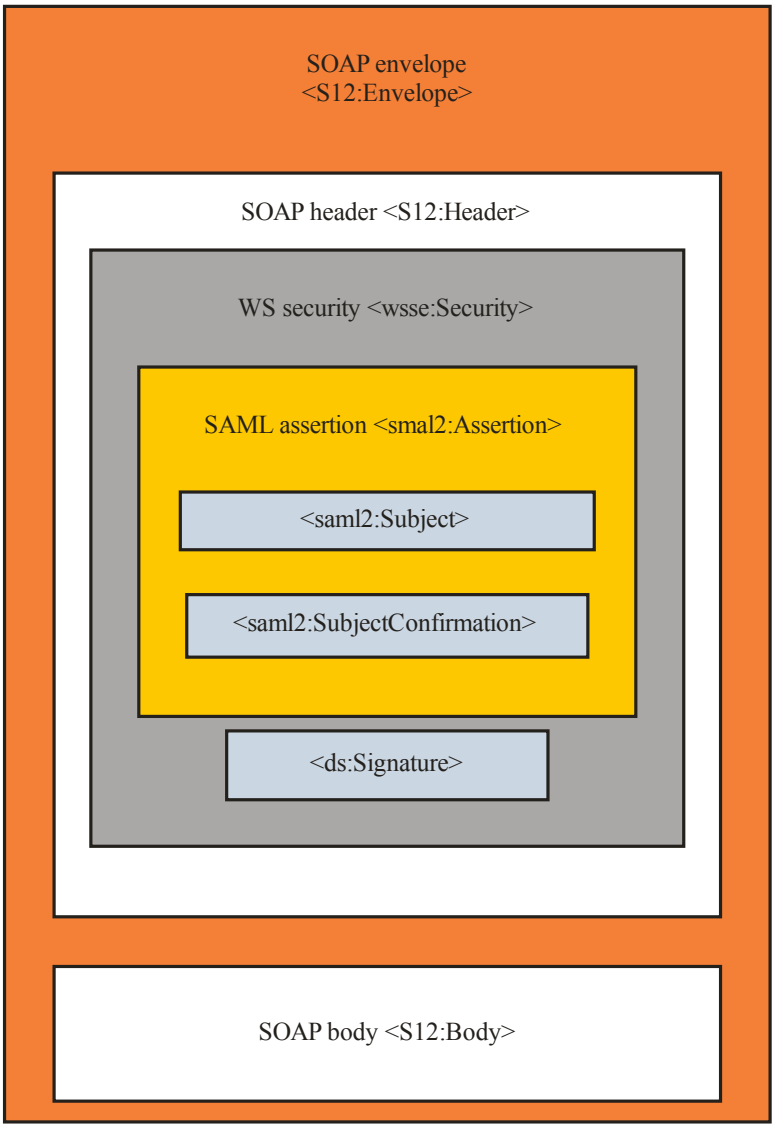


Figure 2 - Structure of the SOAP message with SAML assertion

The following clauses describe the two subject confirmation methods.

6.2.1.2.1 The holder-of-key subject confirmation method

The Figure 3 depicts the structure of the SAML assertion used for the holder-of-key subject confirmation method. The Method attribute of the element <saml2:SubjectConfirmation> indicates the method of the subject confirmation (holder-of-key).

The method specifies that the Sender (also known as Attesting Entity) must prove that it is entitled to make the Statements about the Subject by demonstrating knowledge of the key, which is identified in the <ds:KeyValue> element contained in the <ds:KeyInfo> element of the SAML assertion. The <ds:KeyInfo> element identifies a public or secret key that is used to confirm the identity of the subject. The method further specifies that the sender may do so by signing a digest of the SOAP body with that key. The signature is contained in the element <ds:Signature> of the WS Security header as shown in Figure 2.

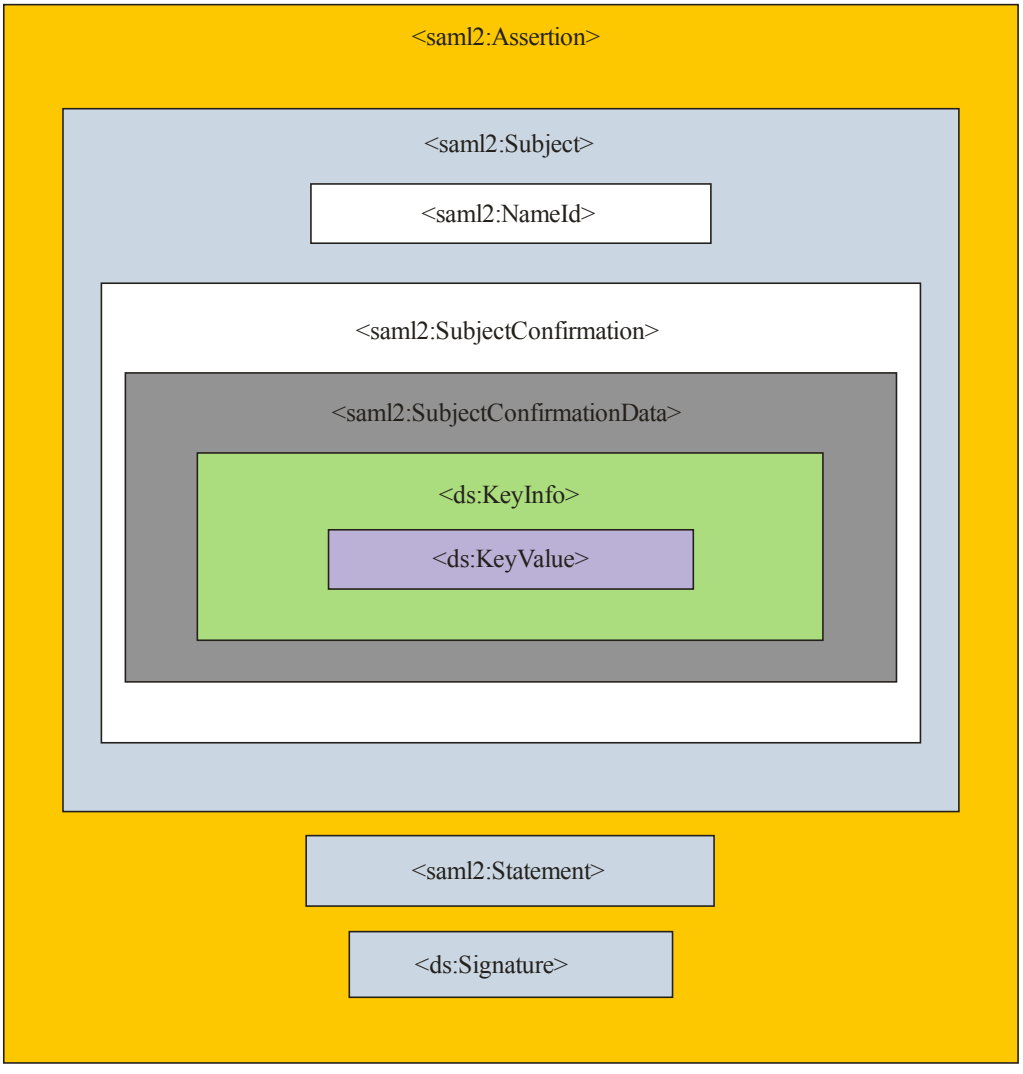


Figure 3 - The structure of the SAML assertion used for the holder-of-key subject confirmation method

The Receiver of the SOAP message obtains the key using information that is provided by the Attesting Entity in the <ds:KeyInfo> element. The Receiver then computes the digital signature of the SOAP body and checks whether it matches the signature provided by the Attesting Entity. If it is the case, then the subject and statements of the SAML assertion may be attributed to the Attesting Entity and the content of the SOAP body whose integrity is protected by the key may be considered as provided by the Attesting Entity.

6.2.1.2.2 The sender-vouches subject confirmation method

The Figure 4 depicts the structure of the SAML assertion used for the sender-vouches subject confirmation method. The Method attribute of the element <saml2:SubjectConfirmation> indicates the method of the subject confirmation (sender-vouches).

The Attesting Entity is trusted by a Receiver to make SAML assertions regarding a subject under condition that value of the Method attribute of the <SubjectConfirmation> element indicates the sender-vouches method.

The Attesting Entity obtains one or more assertions or references to assertions from one or more authorities and includes them in a SOAP message. It then computes a signature of the digest of the SAML assertions and the body of the SOAP message. The signature is contained in the element <ds:Signature> of the WS Security header (shown in

Figure 2).The Attesting Entity optionally provides information to the Receiver on the key that was used to compute the signature. If there is no such information, the Receiver is expected to identify the key by other means.

The Receiver checks validity of the signature. If the signature is valid, the Receiver establishes the fact that the statements have been made about the subject by the Attesting Entity.

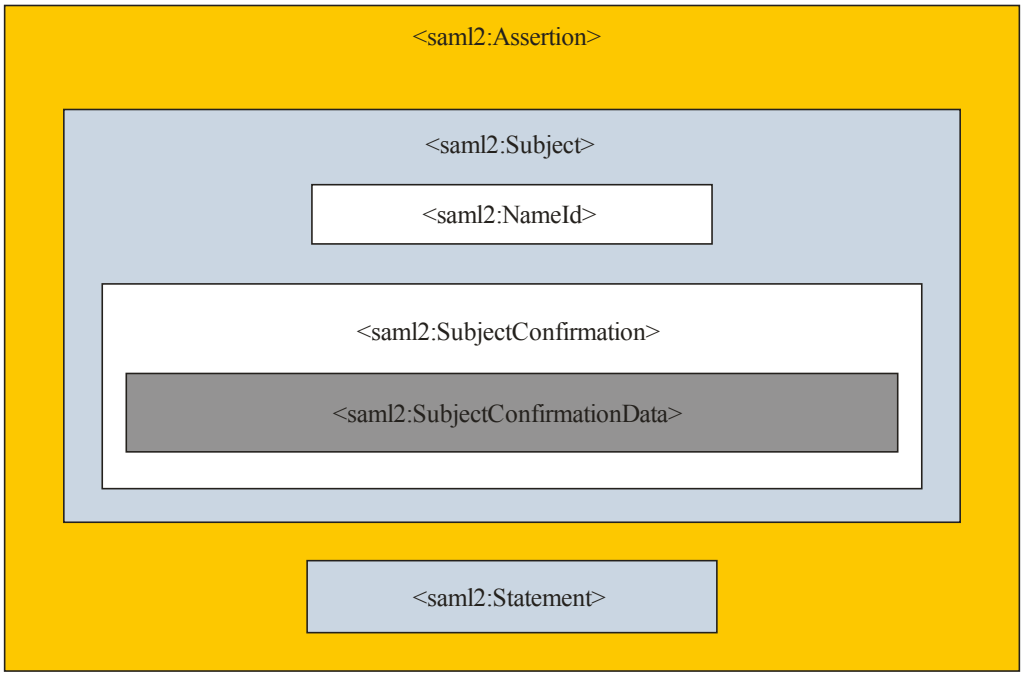


Figure 4 - The structure of the SAML assertion used for the sender-vouches subject confirmation method

6.2.2 Certificate-based authentication

X.509 [ITU-T X.509] certificates may be used for specific application or service based on context and needed level of assurance. The use of X.509 [ITU-T X.509] certificates for authentication is described in ITU-T Recommendation Y.2704, *Security mechanisms and procedures for NGN* [ITU-T Y.2704].

6.2.3 Password-based authentication

Password-based authentication may be used for specific applications or services based on context and needed level of assurance. Refer to the ITU-T Recommendation [ITU-T X.1035] for a description of a password-based authentication mechanism.

6.2.4 One-time Password

One-time password (OTP) may be used for specific application or service based on context and needed level of assurance. One method of implementing OTP is described in [b-IETF RFC 2289].

6.2.5 Use of Authentication and Key Agreement (AKA) for Mutual Authentication

The Universal Mobile Telecommunications System (UMTS) Authentication and Key Agreement (AKA) protocol can be used to provide mutual authentication of the Mobile Station (MS) and the network. The UMTS AKA is a challenge-response protocol, which uses a long-term key shared between the Universal Subscriber Identity Module (USIM) and the Authentication Center (AuC). These entities reside on the Universal Integrated Circuit Card (UICC) of the MS and in the mobile station's home network respectively. In certain business arrangements, the functions of the AuC could be provided by an IdSP. The AKA protocol is specified in [ATIS 33102].

6.2.6 Integration of PKI-based authentication with IMS

IP Multimedia Subsystems (IMS) security is based on the AKA mechanism, which uses a shared secret and a challenge-response protocol for user-network authentication. But security of certain NGN services (e.g., IPTV) is based on Public Key Infrastructure (PKI) certificates. To allow blending of NGN services using PKI certificates and IMS security, it may be desirable to integrate PKI-based authentication with the IMS authentication in such a way that leverages the strength of IMS security.

Integration of the IMS authentication with PKI-based authentication allows the user equipment and network to authenticate each other based on respective certificates and to agree on a set of cryptographic keys based on the same key generation algorithms as in AKA. To this end, the user equipment and network need to be provisioned with the respective private keys and certificates, and be able to perform the PKI operations.

With respect to agreement on the Ciphering Key (*CK*) and Integrity Key (*IK*) the described mechanism for integration specifies two options:

1. Establishing agreement on the *CK* and *IK* keys with the use of a shared secret between the End-User Function and the S-5 - Service user profile functional entity (SUP-FE) defined in [ITU-T Y.2012]
2. Establishing agreement on the *CK* and *IK* keys without the use of the shared secret

The generic call flows for the first option and the second option are depicted by **Figure 5** and **Figure 6**, respectively.

6.2.6.1 Conventions

The following connections are used in this section:

"|" designates the string concatenation

CK designates Ciphering Key

IK designates Integrity Key

K() designates a symmetric key encryption

N_{pr} [] designates encryption with the network private key N_{pr}

N_{pu} [] designates encryption with the network public key N_{pu} available from the network certificate

U_{pr} [] designates encryption with the user private key U_{pr}

6.2.6.2 Entities involved in authentication

- S-5 - Service user profile functional entity (SUP-FE)
- End-User Function. The entity is capable of running a SIP client
- S-n call session control functional entity (CSC-FE), where S-n stands for one of the following entities:
    - S-1 Serving call session control functional entity (S-CSC-FE)
    - S-2 Proxy call session control functional entity (P-CSC-FE)
    - S-3 Interrogating call session control functional entity (I-CSC-FE)

The S-n is used to denote one of these entities when there are no differences between them as far as the described authentication procedure is concerned. Refer to [ITU-T Y.2012] for descriptions of the NGN functional entities (S-1, S-2, S-3, S-5 and End-User Function).

6.2.6.3 Establishing agreement on the CK and IK keys with the use of a shared secret between the End-User Function and S-5 (option 1)

The call flow is depicted by **Figure 5**. The basic steps are as follows:

1. End-User Function sends SIP Register request with the user's *IMPU* and *IMPI* to the S-n
2. The S-1 requests a random challenge *RAND, CK, and IK* from the S-5. The values *RAND, CK, and IK* are specified in [ATIS 33102]
3. The S-1 receives *RAND, CK, and IK* from the S-5 for the user
4. The S-n sends to the End-User Function, the SIP 401 Unauthorized message with a challenge *RAND* and its encrypted value $N_{pr}[RAND]$

   The End-User Function:

   - Receives values *A*, which is supposedly equal to *RAND,* and *B*, which is supposedly equal to $N_{pr}[RAND]$
   - Retrieves the network public key N_{pu}
   - Decrypts *B* with N_{pu} and compares the result to *A*. If the values are equal, then the network is authenticated, if not – the authentication procedure is aborted
   - Generates *IK* and *CK* using the shared secret K_s
   - Generates value $U_{pr}[N_{pu}[K]|K(RAND)]$
5. End-User Function sends to the S-n, SIP Register message with the *IMPU* and *IMPI* identifiers and the value $U_{pr}[N_{pu}[K]|K(RAND)]$
6. The S-1 sends to the S-5, data received in step 5 and requests verification and the user's record

   The S-5 performs the following operations:

   - Looks up the user certificate to obtain the user public key U_{pu}

- Decrypts with U_{pu} the received value C, which is supposedly equal to $U_{pr}[N_{pu}[K]|K(RAND)]$ to retrieve value $D|E$, where D is supposedly equal to $N_{pu}[K]$ and E is supposedly equal to $K(RAND)$
- Decrypts with the network private key N_{pr} value D to obtain K'
- Decrypts with K' value E to obtain $RAND'$
- Compares $RAND'$ with $RAND$. If they match, the user has been authenticated

7. The S-5 communicates the authentication result and the user's record to the S-1
8. The S-1 uses the record to check whether the authenticated user is authorized to register and receive the requested service. If that is the case, the S-n notifies the End-User Function that access is granted using a SIP 200 OK message.

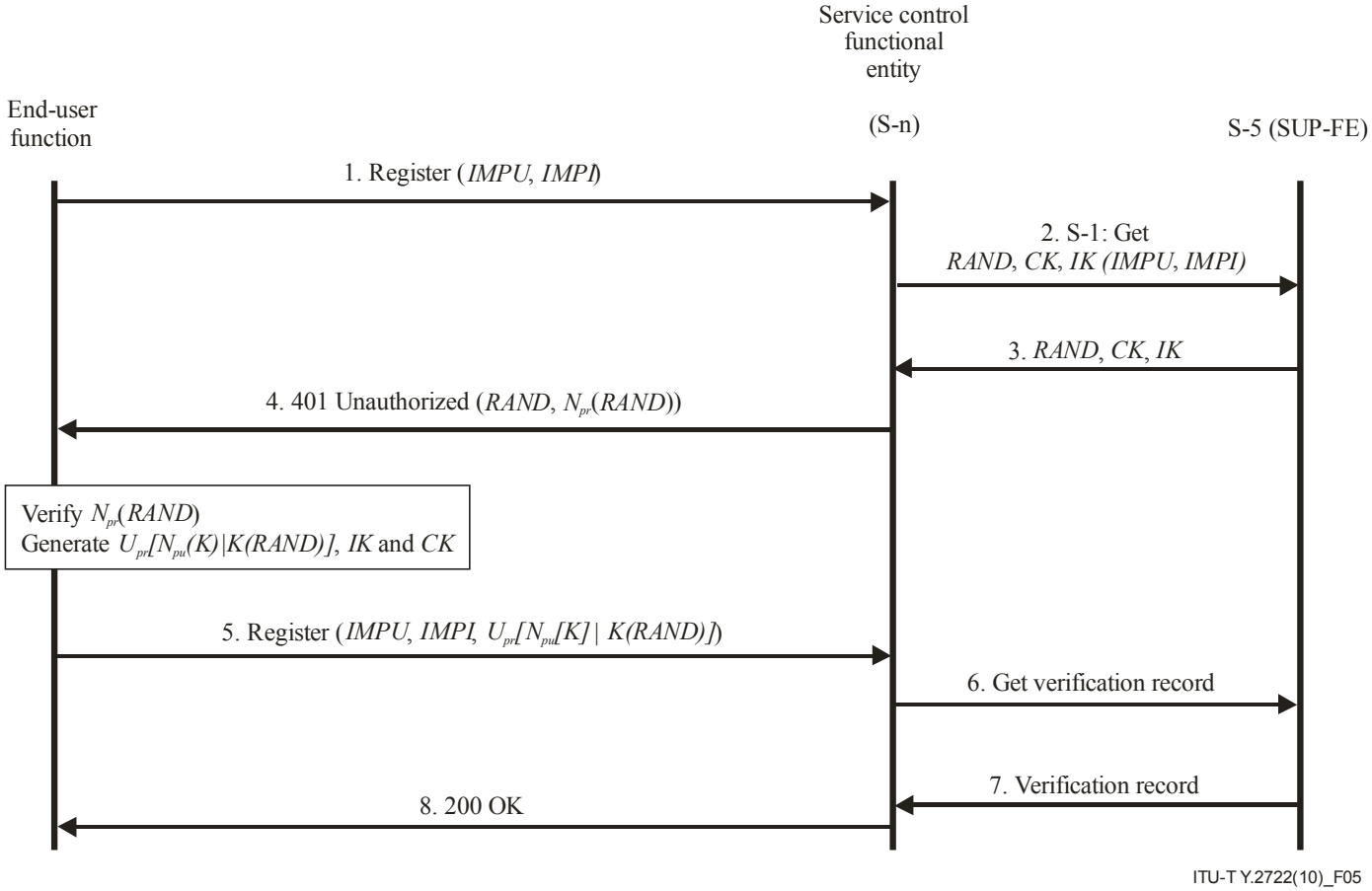


Figure 5 - Integration of the IMS authentication mechanism with PKI-based authentication (option 1)

6.2.6.4 Establishing agreement on the CK and IK keys without the use of a shared secret between the End-User Function and S-5 (option 2)

The call flow is depicted by **Figure 6**. The basic steps are as follows:

1. End-User Function sends SIP Register request with the user's *IMPU* and *IMPI* to the S-n
2. The S-1 requests a random challenge *RAND* from the S-5. The value *RAND* is specified in [ATIS 33102]
3. The S-1 receives *RAND* from the S-5 for the specified user
4. The S-n sends to the End-User Function, the SIP 401Unauthorized message with a challenge *RAND* and its encrypted value $N_{pr}[RAND]$

The End-User Function:

- Receives values *A*, which is supposedly equal to *RAND,* and *B*, which is supposedly equal to $N_{pr}[RAND]$
- Retrieves the network public key N_{pu}
- Decrypts *B* with N_{pu} and compares the result to *A*. If the values are equal, then the network is authenticated, if not – the authentication procedure is aborted
- Generates *IK* and *CK* using the randomly-generated key *K*
- Generates value $U_{pr}[N_{pu}[K]|K(RAND)]$

5. End-User Function sends to the S-n, SIP Register message with the *IMPU* and *IMPI* identifiers and the value $U_{pr}[N_{pu}[K]|K(RAND)]$
6. The S-1 sends to the S-5, data received in step 5 and requests verification, the user's record, and the *CK* and *IK* keys

   The S-5 performs the following operations:

   - Looks up the user certificate to obtain the user public key U_{pu}
   - Decrypts with U_{pu} the received value *C*, which is supposedly equal to $U_{pr}[N_{pu}[K]|K(RAND)]$ to retrieve value *D*|*E*, where *D* is supposedly equal to $N_{pu}[K]$ and *E* is supposedly equal to *K(RAND*)
   - Decrypts with the network private key N_{pr} value *D* to obtain K'
   - Decrypts with *K'* value *E* to obtain *RAND'*
   - Compares *RAND'* with *RAND*. If they match, the user has been authenticated and *K'* = *K*. That is the End-User Function and S-5 now share key *K*
   - Generates the *CK* and *IK* keys using the shared key *K*. For instance, the same functions for generating the *CK* and *IK* that are specified in [ATIS 33102] can be employed with the use of *K* as an input parameter
7. The S-5 communicates the authentication result, the user's record, and the *CK* and *IK* keys to the S-1
8. The S-1 uses the record to check whether the authenticated user is authorized to register and receive the requested service. If that is the case, the S-n notifies the End-User Function that access is granted using a SIP 200 OK message

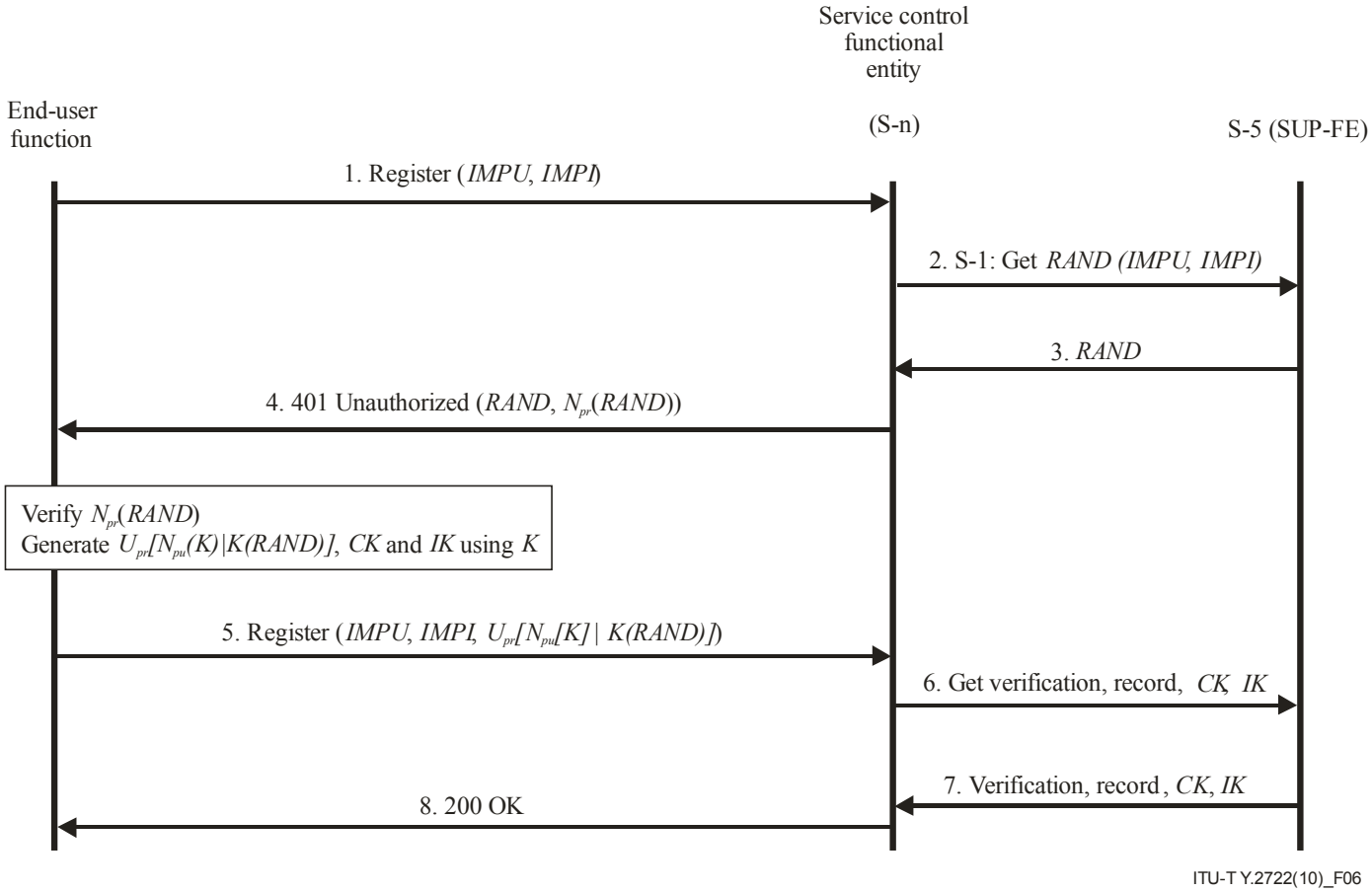


Figure 6 - Integration of the IMS authentication mechanism with PKI-based authentication (option 2)

6.2.6.5 Comparison of the option 1 and option 2

Table 1 provides a comparison between the mechanisms described for options 1 and 2.

Table 1 – Comparison of the option 1 and option 2 for the key agreement between the End-User Function and S-5 on the *CK* and *IK* keys

	Option 1 (with pre-shared secret)	Option 2 (without pre-shared secret)
Advantages	Completely re-uses the AKA mechanism for establishing agreement on the *CK* and *IK* keys	Does not require provisioning of the shared secret between the End-User Function and S-5
Disadvantages	Requires provisioning of the shared secret between the End-User Function and S-5	Requires modifications to the applications running on the End-User Function (e.g., on a smart card) and S-5 for enabling agreement on the *CK* and *IK*

Option 1 should be selected to simplify key agreement on the *CK* and *IK* keys when the End-User Function and the S-5 share a secret. Option 2 should be the choice when the End-User Function and the S-5 do not have a shared secret.

The implementations of this integration mechanism must support both options.

Requirements for the S-5 functional entity

In addition to the capabilities specified in [ATIS 33102], the S-5 must be capable of:

- Storing users' and network's certificates in and retrieving these certificates from the certificate repository
- Performing PKI-based decryption as described in step 6 (for both options)
- Running the Diameter protocol modified to carry information described in step 6 (for both options) and information needed for negotiation with the End-User Function on the PKI-based authentication
- Negotiating with the End-User Function an agreement on the PKI-based authentication method

6.2.6.6 Requirements to the End-User Function

The End-User Function must be capable of:

- Securely storing the user's private key U_{pr}
- Securely storing the shared secret K_s with the network (only for option 1)
- Storing a network X.509 certificate with the network's public key N_{pu}
- Randomly generating one-time session key K and performing the symmetric key encryption with K
- Generating the *CK* and *IK* keys with the use of the shared secret K_s as specified in [ATIS 33102] (only for option 1)
- Generating the *CK* and *IK* keys as described in step 6 for option 2
- Performing PKI-based encryption and decryption described in Steps 4 and 5 for both options
- Running a SIP client with a modified SIP protocol enabling communication of information described in steps 4 and 5
- Negotiating with the S-2 an agreement on the use of the PKI-based authentication

6.2.6.7 Requirements to the S-1

The additional requirements to the S-1 are as follows:

- It must be capable of constructing the SIP messages with information described in step 4 (for both options)
- It must be capable of retrieving from the SIP messages information described in step 5 and repackaging it into the Diameter messages as described in step 6 (for both options)
- It must be capable of performing the PKI-based encryption described in step 4 (for both options)
- It must be able to understand the notification from the S-5 on the use of the PKI-based authentication

6.2.6.8 Requirements to the SIP interfaces between the participating entities

The End-User Function and the S-1 communicate via the S-2 and S-3 functional entities. The S-2 and S-3 entities are not essential to the described authentication and not shown in **Figure 5** and **Figure 6**.

There are SIP interfaces between:

- End-User Function and S-2
- S-2 and S-3
- S-1 and S-3

These interfaces must be able to negotiate the use of the PKI-based authentication (including the specific option for key generation) and to carry information described in steps 4 and 5 (for both options).

6.2.6.9 Requirements to the Diameter interfaces between the participating entities

There are Diameter interfaces between:

- S-1 and S-5
- S-3 and S-5

These interfaces must be able to negotiate the use of the PKI-based authentication (including the specific option for key generation) and to carry information described in step 6 (for both options)

6.2.7 Integration of the PKI-based authentication and the SAML assertion mechanisms

SAML allows having one entity (e.g., IdSP) to perform authentication and another entity (a Relying Party, such as an Application Service Provider) to use the authentication results. In such a scenario, an IdSP may implement multiple authentication methods while the Application Service Provider (ASP) relies on the IdSP's SAML assertions. This scenario is beneficial to both IdSPs and ASPs. Benefits to ASPs are as follows:

- ASP does not have to implement numerous authentication methods.
- ASP could support a wide range of application services with different authentication assurance requirements.

The benefits to the IdSP are as follows:

- It can offer IdM services, particularly authentication, to multiple ASPs
- The IdSP (especially when IdSP is an NGN provider) can utilize its deployed authentication infrastructure to offer IdM services to other providers.

This clause specifies a mechanism for authenticating a client with the use of SAML assertions and PKI-based authentication. This mechanism, along with the one described in clause *6.2.6 Integration of PKI-based authentication with IMS authentication*, allows NGN providers to leverage and utilize their PKI-based infrastructure.

This mechanism is based on the SAML *HTTP redirect binding* specified in [ITU-T X.1141].

6.2.7.1 Entities involved in the authentication and the information flow

- End-User Function. This entity is capable of running a Web client and supporting PKI-based authentication [ITU-T X.509].
- Application Server (AS) — an entity providing a Web service. It plays a role of a Relying Party. It acts as a SAML requestor as defined in [ITU-T X.1141].
- A-2: Application gateway functional entity (APL-GW-FE), which is enabled to perform the PKI-based authentication and act as a SAML responder as defined in [ITU-T X.1141].

- S-5 - Service user profile functional entity (SUP-FE)

The information flow of the authentication procedure is depicted by Figure 7 - The basic steps of data exchange for the PKI-based authentication with SAML-assertion and described below. Refer to [ITU-T Y.2012] for descriptions of NGN functional entities (End-User Function, AS, A-2 and S-5).

6.2.7.2 Conventions

The description uses the following conventions:

"|" designates the string concatenation

K() designates a symmetric key encryption

K_s designates a secret shared between A-2 and AS

N_{pr} *[]* designates encryption with the network private key N_{pr}

N_{pu} *[]* designates encryption with the network public key N_{pu} available from the network certificate

U_{pr} *[]* designates encryption with the user private key U_{pr}

RAND designates randomly-generated challenge

6.2.7.3 Mechanism's parameters

This clause specifies the mechanism-specific parameters. The list of the parameters is as follows:

`pki-auth-challenge` – parameter for transmitting the value of *RAND*

`pki-auth-challenge-encrypted` – parameter for transmitting the value of $N_{pr}[RAND]$

`pki-auth-user-signature` – parameter for transmitting the value of $U_{pr}[N_{pu}[K]|K(RAND)]$

`pki-auth-keyinfo` – parameter for transmitting the value of $K_s(K)$

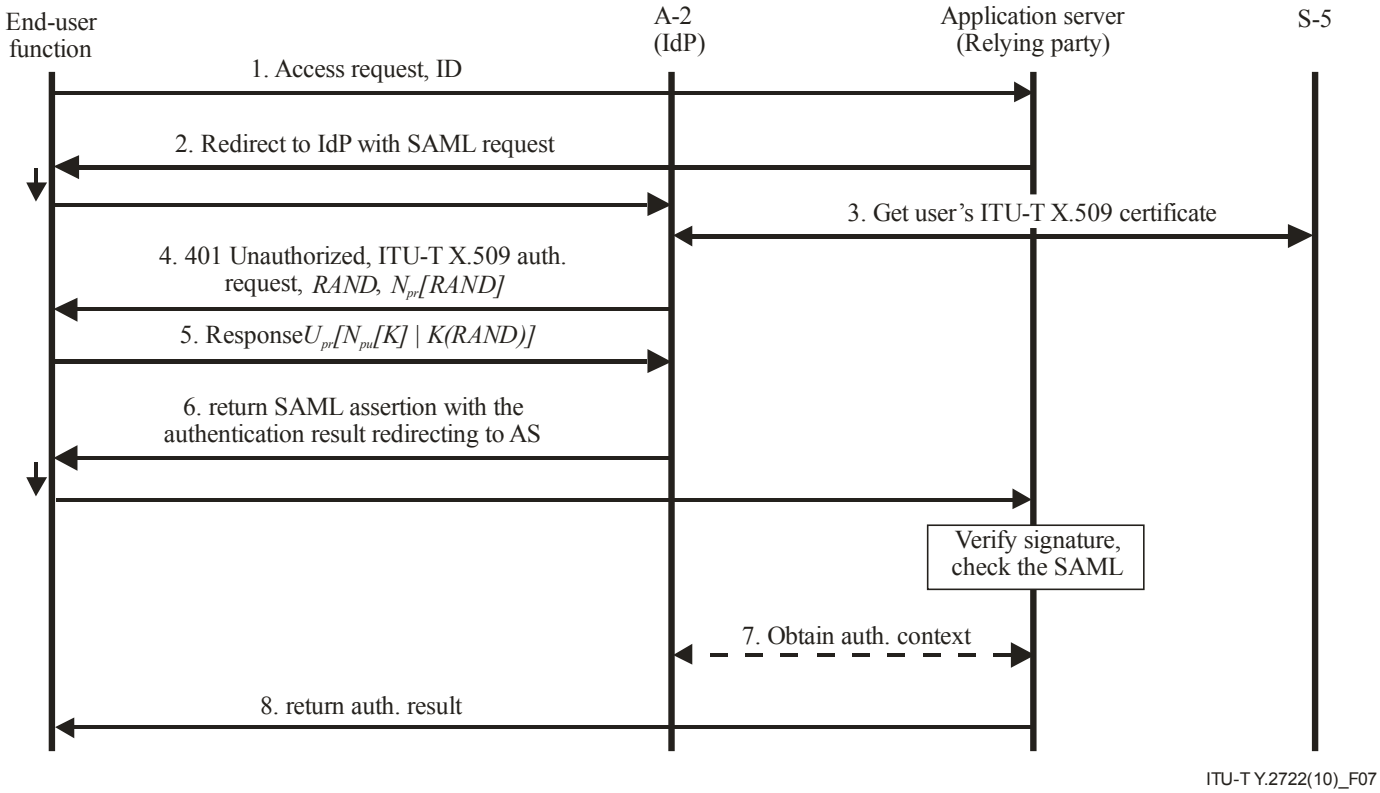


Figure 7 - The basic steps of data exchange for the PKI-based authentication with SAML-assertion

The mutual authentication of the End-User Function and A-2 is similar to the procedure used by the mechanism for integration of PKI-based authentication with IMS authentication, which is described in clause 6.2.6.

The basic steps of the procedure that relies on the PKI-based authentication and SAML assertions are as follows:

1. A Web client of the End-User Function issues an HTTP Access request to the Application Server (AS). The request includes a user identifier and the URL of the A-2.
2. The Application server acting as a SAML requester responds to the HTTP request by sending a SAML request. The SAML request is encoded into the HTTP response's Location header with the HTTP status set to either 302 or 303. The agent of the End-User Function delivers the SAML request by issuing HTTP GET request to A-2, which acts as a SAML responder. This HTTP redirection procedure, known as *HTTP redirect binding*, is specified in [ITU-T X.1141]. To ensure authentication and integrity of the URL-encoded message, it should be signed as specified in clause *10.2.4.5.2 Security Considerations* of [ITU-T X.1141]. The shared secret K_s must be used for signing.
3. After signature validation, A-2 obtains from the S-5, the End-User's certificate and checks whether it is valid. The certificate contains the End-User Function's public key.
4. A-2 responds to the End-User Function with an HTTP response message indicating that authentication with the use of X.509 [ITU-T X.509] certificate is required. This is accomplished by setting the value of the response header `WWW-Authenticate` [b-IETF RFC 2616] to "`pki-auth`". The body of the message includes the `pki-auth-challenge` and `pki-auth-challenge-encrypted` parameters that carry the values

of the randomly-generated challenge *RAND* and its encryption $N_{pr}[RAND]$ respectively. The header `Content-Type` must be set to `application/x-www-form-urlencoded.`

5. The End-User Function

   - Retrieves values *A*, which is supposedly equal to *RAND,* and *B*, which is supposedly equal to $N_{pr}[RAND]$
   - Retrieves the network public key N_{pu}
   - Decrypts *B* with N_{pu} and compares the result to *A*. If the values are equal, then the network is authenticated, if not – the authentication procedure is halted
   - Generates a secret key *K*
   - Generates value $U_{pr}[N_{pu}[K]|K(RAND)]$, sets the parameter `pki-auth-user-signature` to that value, and sends it in the body of an HTTP POST message to A-2. The header `Content-Type` of the message must be set to the value `application/x-www-form-urlencoded`

   After this step the A-2 checks whether the response is valid. To that end A-2 performs the following operations:

   - Looks up the user certificate to obtain the user public key U_{pu}
   - Decrypts with U_{pu} the received value *C*, which is supposedly equal to $U_{pr}[N_{pu}[K]|K(RAND)]$ to retrieve value *D*|*E*, where *D* is supposedly equal to $N_{pu}[K]$ and *E* is supposedly equal to *K(RAND*)
   - Decrypts with the network private key N_{pr} value *D* to obtain *K'*
   - Decrypts with *K'* value *E* to obtain *RAND'*
   - Compares *RAND'* with *RAND*. If they match, the user has been authenticated and *K'* = *K*. That is the End-User Function and A-2 now share key *K*

6. If all the above steps were successful, the A-2 performs the following operations

- Generates a SAML assertion setting the attribute Method of the element <SubjectConfirmation> to the value sender-vouches.
- Computes value $K_s(K)$.
- Includes the assertion in a SAML response. It then delivers the SAML response and the computed value $K_s(K)$ over HTTP in the same manner as described for the SAML request in step 2 (i.e., as part of a query string). The value $K_s(K)$ is carried by the parameter `pki-auth-keyinfo`
- To ensure origin authentication and integrity of the URL-encoded message, A-2 signs it as specified in clause *10.2.4.5.2 Security Considerations* of [ITU-T X.1141]. The shared secret K_s should be used for signing.

  After validating the signed URL, AS is assured that the SAML assertion is made by A-2. The AS checks the assertion itself (e.g., to ensure that *conditions* are met). After that, the AS retrieves the value $K_s(K)$ and decrypts it using shared K_s to obtain *K*. At this point AS has authenticated the End-User Function and both entities share the key *K,* which can be used for securing communications between them.

7. The AS, if it is required by policy for making an authorization decision, obtains information about the authentication context. In that case A-2 responds with information specified by the *Public key – X.509* authentication context class [ITU-T X.1141].
8. AS sends to the End-User Function the result of the authorization decision.

6.2.7.4 Additional requirements for the entities participating in the authentication

In order to support the described mechanism, the participating entities must meet the following requirements:

6.2.7.4.1 Requirements for the End-User Function

The End-User Function must be capable of:

- Running HTTP client
- Securely storing its private key U_{pr} (e.g., on a smart card)
- Obtaining the network public key N_{pu}
- Performing encryption and decryption
- Generating a key K

6.2.7.4.2 Requirements for the Application Server (AS)

- AS must support SAML [ITU-T X.1141]
- AS must have a shared secret (K_s) with A-2

6.2.7.4.3 Requirements for the A-2 Functional Entity

The A-2 functional entity must be able to:

- Support HTTP protocol
- Securely store its private key N_{pr}
- Obtain the user public key U_{pu}
- Perform encryption and decryption
- Generate a random challenge *RAND*
- Support SAML [ITU-T X.1141]
- Have a shared secret (K_s) with AS

6.2.7.4.4 Requirements for the S-5 Functional Entity

The S-5 Functional Entity should be capable of storing the users' X.509 certificates or retrieving the certificates from the repository (or both).

6.2.7.5 Additional requirements for the interfaces between the participating entities

The requirements for the interfaces are as follows:

- The interface between the End-User Function and the Application Server must support HTTP protocol [b-IETF RFC 2616]
- The interfaces between the End-User Function and A-2 Functional Entities must support HTTP protocol [b-IETF RFC 2616]

- The interface between A-2 and Application Server must support SAML [ITU-T X.1141]
- The interface between the A-2 and S-5 Functional Entities must support a query-response mechanism that allows A-2 to obtain the users' X.509 certificates from S-5

6.2.8 Integration of *OpenID*-based authentication with the AKA authentication

Integration of AKA authentication with OpenID-based authentication allows combination of network-centric and user-centric IdM capabilities. The integration mechanism:

- Enables the network providers to provide identity services to users accessing Web applications
- Can be used to provide users with single sign-on (SSO) across the IMS network and web services environment with an existing ISIM application as well as with other SIM applications that rely on AKA
- Allows users to control their public identifiers on the Web as specified in [b-OpenID v.2] while leveraging NGN services
- Improves user security by engaging a user-trusted network provider in the access control to the Web applications.

Technical Report [b-3GPP TR 33.924] describes several solutions for integration of *OpenID* with AKA that rely on the use of the Bootstrapping Server Function (BSF).

This section describes an additional mechanism for integration of *OpenID* and AKA. To this end, the *OpenID* specification calls for a variety of authentication mechanisms.

OpenID can interwork with other technologies, such as *OAuth*, as shown in the Appendix III.

6.2.8.1 Entities involved in the authentication and the information flow

- End-User Function. This entity is capable of running a Web client and communicating with the appropriate SIM application
- Application server — an entity providing a Web service. It plays a role of a Relying Party
- A-2: Application gateway functional entity (APL-GW-FE), which is enabled to serve as an *OpenID* [b-OpenID v.2] identity provider. (The A-2 optionally shares a short-term secret with the Application server as specified in [b-OpenID v.2])
- S-5 - Service user profile functional entity (SUP-FE)

The information flow of the authentication procedure is depicted by

Figure 8. The procedure of establishing the short-term signing key between the Application server and A-2 is not shown. The figure shows the basic steps of the procedure for two *OpenID* options:

a. The A-2 and the Application server share a secret

b. The A-2 and the Application server do not share a secret

The common steps for both options are 1 through 6. The step 7a is for the option *a* only.

The steps 7b, 8b, and 9b are for the option *b* only.

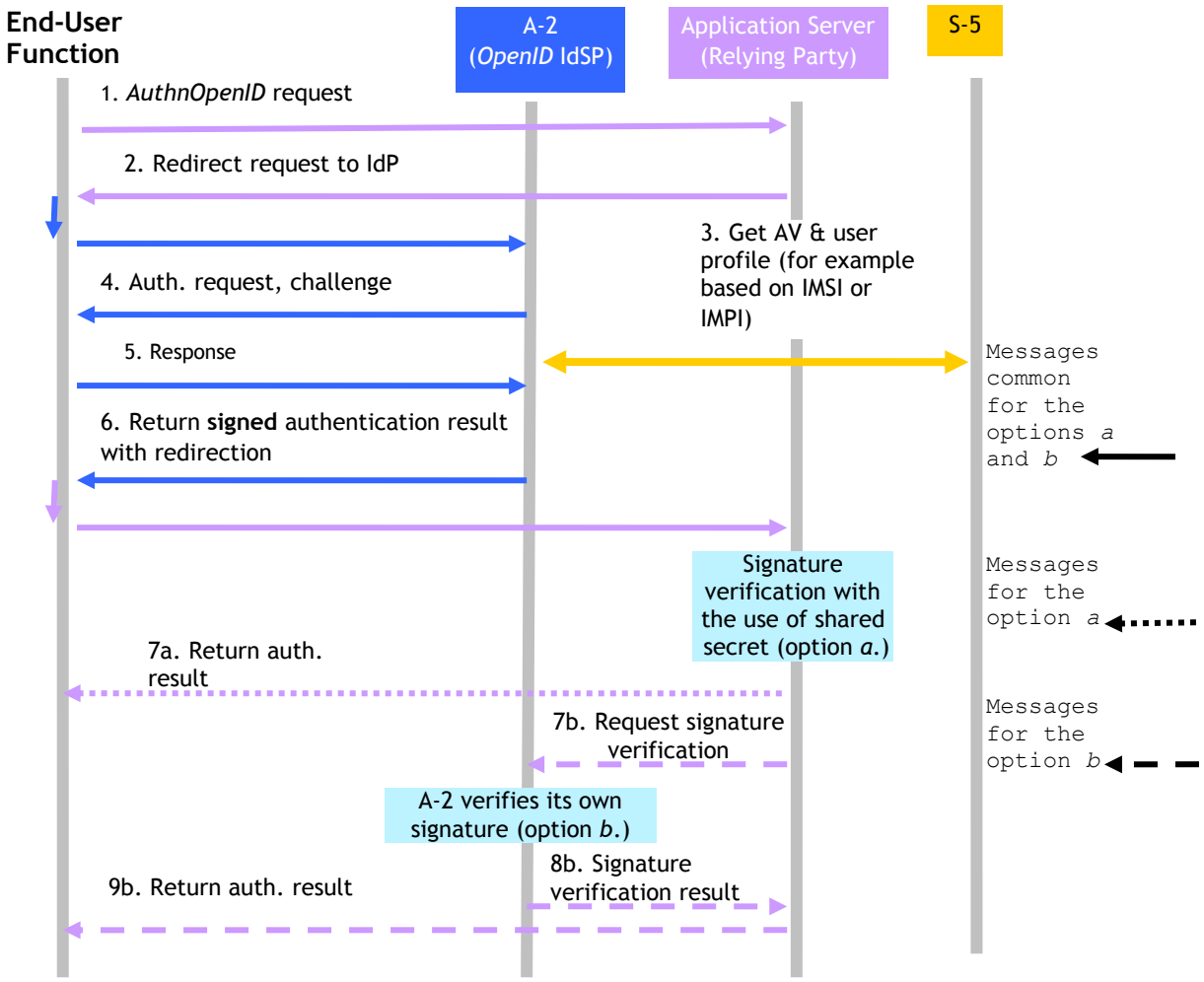


Figure 8 - Integration of the AKA authentication mechanism with OpenID

The basic steps are as follows:

1. A Web client of the End-User Function issues an authentication request *AuthnOpenID* to the Application server. The request includes an *OpenID* identifier.
2. The Application server, using the presented *OpenID* identifier, discovers the URL of A-2, which serves as an *OpenID* identity provider, and redirects the user authentication request to that URL.

   After this step, A-2 correlates the user identifier with the appropriate identity (such as IMSI or IMPI)
3. A-2 obtains from the S-5, the AKA authentication vector AV and the user profile based on the IMPI.
4. A-2 sends to the End-User Function, authentication request using the HTTP Digest AKA method [b-IETF RFC 4169] or [b-IETF RFC 3310]. The request includes a challenge and a quantity that enables the End-User Function to authenticate the network.

After this step the End-User Function authenticates the network as specified in [b-IETF RFC 4169] or [b-IETF RFC 3310].

5. End-User Function sends to the A-2, response to the challenge as specified in [b-IETF RFC 4169] or [b-IETF RFC 3310].

   After this step the A-2 authenticates the End-User Function as specified in [b-IETF RFC 4169] or [b-IETF RFC 3310].

6. The A-2 sends to the End-User Function, a signed message asserting that the claimed *OpenID* identifier belongs to the user. The message is signed with the use of a secret shared with the Application server for the option a. For the option b, the message is signed with the A-2 secret key. The message includes a request to redirect the Web client of the End-User Function to the Application server. The details of the signing and redirection procedures are described in [b-OpenID v.2]. [b-OpenIDv2] also specifies measures to prevent attacks based on the reuse of the signed authentication assertion.

Steps that are specific to option *a*:

7a. After verifying the signature of the response received in step 6, the Application server notifies the End-User Function of the authentication result. The Application server uses the secret shared with the A-2 for such verification.

If there is a failure in one of the following steps: 1 through 6, or 7a – the authentication procedure stops.

Steps that are specific to option *b*:

7b. The Application server sends a copy of the message received in step 6 to the A-2 with a request to verify the signature.

8b. After verifying its own signature, the A-2 reports the verification result to the application server.

9b. The Application server reports the authentication result to the End-User Function.

If there is a failure in one of the following steps: 1 through 6, 7b, 8b, or 9b – the authentication procedure stops.

6.2.8.2 Additional requirements for the entities participating in the authentication

In order to support the described mechanism, the participating entities must meet the following requirements:

- 6.2.8.2.1 Requirements for the End-User Function

  The End-User Function must be capable of:

  - Authenticating with the use of the HTTP Digest AKA method
  - Communicating with the appropriate SIM application

- 6.2.8.2.2 Requirements for the Application server

  The Application server must be able to support *OpenID* specification version 2.0 [b-OpenID v.2]

- 6.2.8.2.3 Requirements for the A-2 Functional Entity

  The A-2 functional entity must be able to:

- Perform the HTTP Digest AKA authentication
- Correlate the user *OpenID* identifier with the appropriate identifier (such as IMSI or IMPI)
- Serve as an *OpenID* identity provider

- 6.2.8.2.4 Requirements for the S-5 Functional Entity

  There are no other requirements to the S-5 Functional Entity than those specified in [ITU-T Y.2012]

6.2.8.3 Additional requirements for the interfaces between the participating entities

The requirements for the interfaces are as follows:

- The interface between the End-User Function and the Application server must support the *OpenID* authentication as specified in specification version 2.0 [b-OpenID v.2]
- The interfaces between the End-User Function and A-2 Functional Entities must support the HTTP Digest AKA protocol [b-IETF RFC 4169] or [b-IETF RFC 3310]
- The interface between the A-2 and S-5 Functional Entities does not have any mechanism-specific requirements

6.2.8.4 Mechanism for interworking of *OpenID* and AKA for the split user terminal scenario

The mechanism described in this section also supports the *split terminal scenario*, described in [b-3GPP TR 33.924]. The *split user terminal* scenario refers to a situation where an *authenticating agent* (an entity with access to the UICC card) and the *browsing agent* are not located on the same user terminal.

Considering that in the direct AKA solution specified in this section, IdSP corresponds to a collapsed NAF/BSF, the scenarios described in [b-3GPP TR 33.924] are completely supported by the solution. The mechanism relies on the direct AKA authentication instead of the GBA-based authentication.

6.2.9 GBA

The Generic Bootstrapping Architecture (GBA) specifies a framework for bootstrapping authentication and establishing key agreement leveraging the 3GPP Authentication and Key Agreement (AKA) mechanism. The GBA facilitates authentication of the End-Users to Network Application Function (NAF) and can be used in NGN Identity Management for enabling:

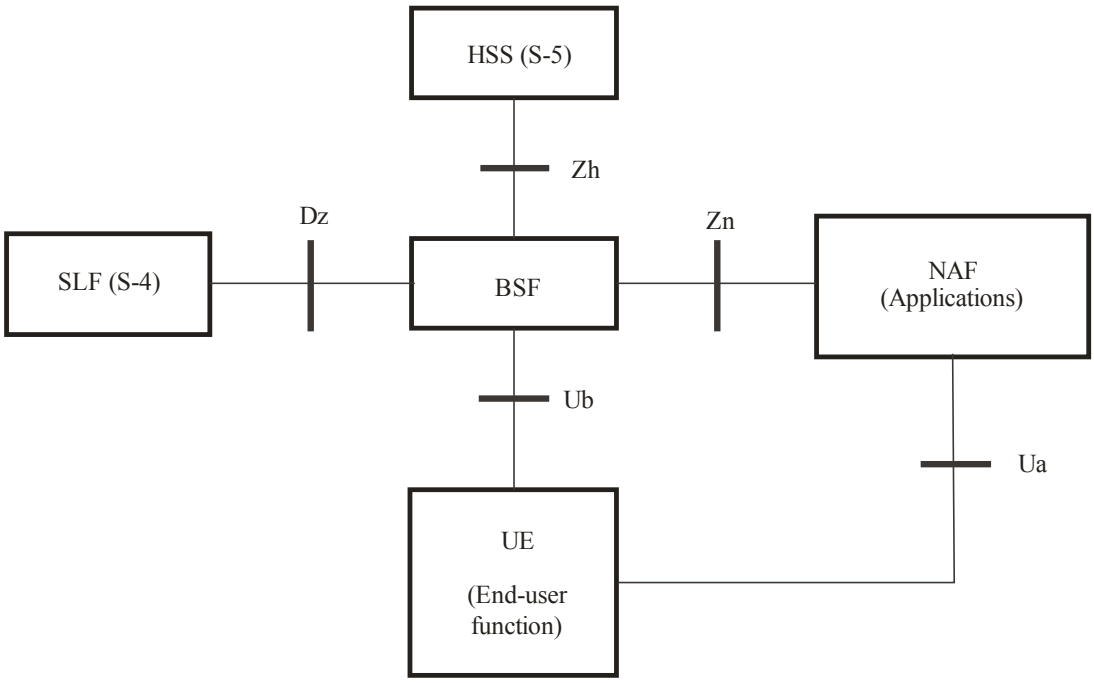
- Authentication and key agreement
- Privacy protection
- Single Sign On

The GBA is an authentication system that includes three parties:

- An end-user who is trying to obtain network services using User Equipment (UE)
- Application server (called Network Application Function or NAF)
- A trusted entity (called Bootstrapping Server Function or BSF), which is involved in authentication and key exchange between two other entities.

The GBA enables authentication of the End-User, who is using UE, to an application server (NAF) without revealing the End-User's long-term credentials and secrets to the NAF by using a trusted entity BSF.

The following figure depicts the GBA *Simple network model for bootstrapping* [b-ETSI TS 133 220] and provides mapping of the 3GPP-defined entities to the functional entities specified in [ITU-T Y.2012]



Note - Labels in parentheses denote entities specified in [ITU-T Y.2012].

ITU-T Y.2722(10)_F09

Figure 9 – Simple network model for bootstrapping

These are the basic steps of the GBA procedure:

1. NAF requests authentication and negotiates the use of GBA over the Ua reference point.
2. The BSF client that runs on the UE initiates bootstrapping procedure over the reference point Ub. The BSF fetches authentication information and the GBA user security settings from the HSS over Zh. The UE and the BSF mutually authenticate using http Digest AKA. The procedure results in the UE receiving bootstrapping transaction identifier (B-TID) from the BSF and establishing a shared key (Ks) between the UE and the BSF.
3. UE derives Ks_NAF from Ks and sends B-TID (along with the application-specific data) to the NAF.
4. The NAF sends B-TID to the BSF over Zn reference point.
5. The BSF based on B-TID determines the Ks that should be used, derives Ks_NAF from it and sends Ks_NAF to the NAF.
6. Finally, UE and NAF can authenticate each other using the shared key Ks_NAF. The exact authentication procedure depends on the protocol between the UE and NAF. For instance, GBA specifies that HTTP-based applications can use either HTTP Digest authentication [b-IETF RFC 2617] or TLS pre-shared key ciphersuites [b-IETF RFC 4279].

Note: The BSF queries the SLF over the Dz reference point to obtain the name of the HSS containing the subscriber-specific data. The SLF is not needed when the BSF is configured to use a pre-defined HSS.

Mapping of the GBA entities to the NGN entities specified in Y.2012, *Functional requirements and architecture of the NGN of Release 1*[ITU-T Y.2012] are as follows*:*

- NAF corresponds to *Applications* entity of the Y.2012 Figure 3: NGN generalized functional architecture.
- HSS corresponds to S-5 Service User Profile FE
- SLF corresponds to S-4 Subscription Locator FE
- UE corresponds to the End-User Function

6.2.10 IMSI-based authentication

Depending on the required level of assurance, the IMSI (International Mobile Subscriber Identity) may be used in WAP network for authentication providing backward compatibility. As the IMSI is a unique alphabetic string, it can be used as an entity-identity for a particular service.

The approach is as follows

- Uses the IMSI code as entity-identity in wireless application;
  Provides reliable service channel on condition that the endpoint has legitimate IMSI, when the endpoint requests for authentication;
- Assumes that all of the systems trust in the WAP gateway’ s authentication result, and provide services for that entity;
- Can be used to provide a Single Sign-On function by virtue of the uniqueness of the IMSI for the same endpoint between GPRS/CDMA 1x and wireless application (e.g., Wireless-mailbox etc.);
- Makes the IMSI code security protected.

6.3 Correlation and Binding

ITU-T Recommendation Y.2720 *NGN Identity Management Framework* [ITU-T Y.2720] states that *identity information (e.g., identifiers, credential and attributes) may be correlated to establish a binding to assure the identity of an entity.*

An objective of a solution that enables correlation is to gather various types of identity information from different sources and present it to the applications in a unified format that they would understand.

The concept of such a solution is illustrated by **Error! Reference source not found.**. The figure depicts three example sources of identity information: HSS, presence server, and a database of the application-specific user data. An application may need all three types of information for making authentication and authorization decisions. In the depicted example, the correlation mechanism employs the Diameter, LDAP, and SQL protocols for obtaining data from the respective sources. This data then presented to the application in a format that it understands. Thus, the correlation mechanism relieves applications from a burden to support multiple protocols for communicating with different sources of identity information.

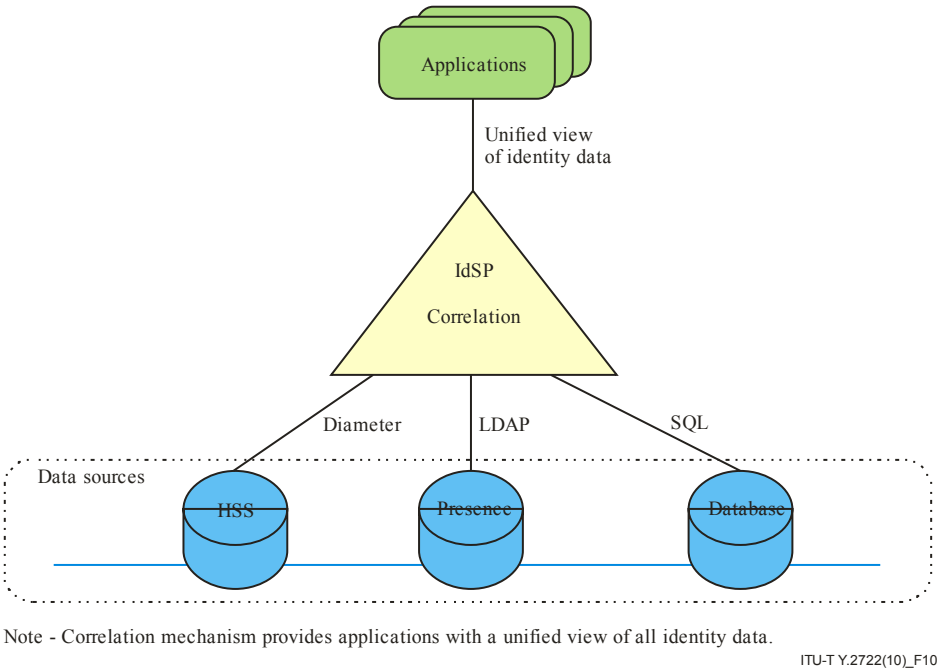


Figure 10 - Correlation of identity information

6.4 Discovery

The ITU-T draft Recommendation *NGN Identity Management requirements and use cases* [Y.2721] specifies that *NGN/IdSP is required to support functions and capabilities to discover sources of identity information* within an NGN/IdSP domain and across different NGN/IdSP domains.

This section provides the examples of standard mechanisms that support these requirements and the references to the relevant specifications.

6.4.1 Intra-network Discovery

The ITU-T Recommendation *Functional Requirements and Architecture of the NGN* [ITU-T Y.2012] defines a special entity – the subscription locator functional entity (SL-FE) – that provides an address of the service user profile functional entity (SUP-FE) that stores identity information of a particular subscriber. SL-FE enables discovery of SUP-FE, which is responsible for storing user profiles, subscriber-related location data, and presence status data. Through querying SUP-FE the network entities can obtain this identity information. As specified in [ITU-T Y.2012], the following network entities may query SL-FE for the address of the appropriate SUP-FE:

- Application support functional entity (AS-FE)
- Interrogating call session control functional entity (I-CSC-FE)
- Serving call session control functional entity (S-CSC-FE)

A mechanism, which enables these entities to find in an operator's network the address of SUP-FE that stores the identity information for a given user is specified in [3GPP TS 23.228]. Note that the mapping of the entities of [ITU-T Y.2012] to the entities of [3GPP TS 23.228] is as follows:

- AS-FE corresponds to AS
- I-CSC-FE corresponds to I-CSCF
- S-CSC-FE corresponds to S-CSCF
- SL-FE corresponds to SLF

6.4.2 Inter-network Discovery

Examples of the mechanisms for inter-network discovery of an IdSP include those specified in SAML [ITU-T X.1141] and ID-WSF [b-LA WSF]. These mechanisms depend on the pre-established agreements among the involved entities (e.g., IdSP and Relying Party) or members of a federation.

Another example is OpenID [b-OpenID v.2] which specifies a discovery mechanism that enables a relying party to locate a user's IdSP based on the user-supplied OpenID identifier.

6.5 IdM Communications and Information Exchange

This section recommends protocols and mechanisms to communicate and exchange identity information

6.5.1 Security of IdM Communications and Exchange

This section recommends mechanisms to provide integrity and confidentiality protection of IdM communications

6.5.1.1 Solutions based on SAML 2.0 [ITU-T X.1141]

For both integrity and confidentiality protection, SAML 2.0 recommends the use of a secure channel or secure network protocol such as TLS or IPsec to be configured to protect the packets transmitted via the network connection.

For message level integrity protection in addition to the secured communication channel, XML Signature can be used. The section "8.4 SAML and XML signature syntax and processing" of ITU-T Recommendation X.1141 [ITU-T X.1141] is required to be followed when XML signature is used.

For message level confidentiality protection in addition to the secured communication channel, XML Encryption can be used. The section "8.4 SAML and XML signature syntax and processing" of ITU-T Recommendation X.1141 [ITU-T X.1141] is required to be followed when XML Encryption is used.

6.5.1.2 Identity Web Services Framework (known as ID-WSF)

In order to use ID-WSF, its communications and its messages between the sender and recipient are expected to have their integrity and confidentiality protected. Like SAML 2.0, it recommends the use of a secure channel or secure network protocol such as TLS or IPsec to be configured to protect the packets transmitted via the network connection [b-LA ID-WSF security].

(1) Transport Layer Channel Protection

In case of using SSL or TLS as secure network protocol for ID-WSF, it is required to use either SSL 3.0, TLS 1.0 or higher. An entity that terminates an SSL (3.0) or TLS (1.0) connection is required to offer or accept suitable cipher suites during the handshake. Recommended TLS 1.0 cipher suites (or their SSL 3.0 equivalent) are as follows, although they are not exhaustive.

- TLS_RSA_WITH_RC4_128_SHA
- TLS_RSA_WITH_3DES_EDE_CBC_SHA
- TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA
- TLS_RSA_WITH_AES_CBC_SHA
- TLS_DHE_DSS_WITH_AES_CBC_SHA

For signing and verification of protocol messages, communicating entities is recommended to use certificates and private keys that are distinct from the certificates and private keys applied for SSL or TLS channel protection.

Other security protocols such as *IPsec* or *Kerberos* may be used as long as they implement equivalent security measures.

(2) Message Confidentiality Protection

In the presence of intermediaries, communicating entities are required to ensure that sensitive information is not disclosed to unauthorized entities. In this case, these entities are required to use the confidentiality mechanisms specified in Web Services Security (WSS) SOAP Message Security by OASIS [b-OASIS WSS SOAP], to encrypt the SOAP envelope <S:Body> Content.

(3) Message Integrity Rules

Message Integrity Rules in this section only applies if Web Services Security (WSS) SOAP Message Security by OASIS [b-OASIS WSS SOAP] is used for a ID-WSF protocol message bound to SOAP according to the Liberty SOAP Binding Version 2.0 [b-LA SOAP binding].

In this case, the sender is required to create a single <ds:Signature> contained in the <wsse:Security> header and this signature is required to reference all of the message components required to be signed.

In particular, this signature is required to reference the SOAP Body element (the element itself), the security token associated with the signature, and all headers in the message that have been defined in the Liberty SOAP Binding Version 2.0 [b-LA SOAP binding], including both required and optional header blocks.

An example security token is a <saml2:Assertion> element conveyed in the <wsse:Security> header. The wsu:Timestamp header in the wsse:Security header block, the wsa:MessageID, wsa:RelatesTo, sb:Framework, sb:Sender and sb:InvocationIdentity header blocks are examples of header elements that would be referenced in a signature.

Note that care is required to be taken when constructing elements contained in Reference Parameters in Endpoint References, as these will be promoted to SOAP header blocks. Appropriate measures should be taken to avoid conflicting or duplicate id attributes, for example by using techniques to generate unique ids.

If the message is signed, the sender is required to include the resultant XML signature in a <ds:Signature> element as a child of the <wsse:Security> header.

The <ds:Signature> element is required to refer to the subject confirmation key with a <ds:KeyInfo> element. The <ds:KeyInfo> element is required to include a <wsse:SecurityTokenReference> element so that the subject confirmation key can be located within the <wsse:Security> header. The inclusion of the reference is recommended to adhere to the guidance specified in section 3.4.2 of Web Services Security: SAML Token Profile 1.1. by OASIS [b-OASIS SAML token].

i) Sender Processing Rules

The construction and decoration of the <wsse:Security> header element is required to adhere to the rules specified in the Web Services Security: SAML Token Profile 1.1 by OASIS [b-OASIS SAML token] .

The <wsse:Security> header element is required to have a mustUnderstand attribute with logical value true.

The sender is required to place the message authentication security token as a direct child of the <wsse:Security> element.

The sender is required to follow the Message Integrity rules outlined for senders and recipients when message authentication mechanisms are used.

The following considerations do not apply to Bearer tokens:

For deployment settings which require independent message authentication, the obligation is required to be accomplished by signing the message body and portions of the header and placing the <ds:Signature> as a direct child of the <wsse:Security> header.

For deployment settings which do not require independent message authentication, the subject confirmation obligation may be accomplished by correlating the certificate and key used to affect peer entity authentication with the certificate and key described by the message authentication token. To accommodate this, the assertion issuing authority is required to construct the assertion such that the confirmation key can be unambiguously verified to be the same certificate and key used in establishing peer entity authentication. This is necessary to mitigate the threat of a certificate substitution attack. It is recommended that the certificate or certificate chain be bound to the subject confirmation key.

ii) Recipient Processing Rules

The recipient is required to locate the <wsse:Security> element for which it is the target. This MUST adhere to the rules specified in Web Services Security (WSS) SOAP Message Security by OASIS [b-OASIS WSS SOAP] and the applicable WSS token profiles (e.g., Web Services Security: SAML Token Profile 1.1. by OASIS [b-OASIS SAML token] for SAML tokens).

The <wsse:Security> header element is required to have a mustUnderstand attribute with logical value true and the recipient must be able to process this header block according to Web Services Security (WSS) SOAP Message Security by OASIS [b-OASIS WSS SOAP] and the appropriate WSS token profiles (e.g., Web Services Security: SAML Token Profile 1.1. by OASIS [b-OASIS SAML token] for SAML tokens).

The recipient is required to locate the security token and the recipient is required to determine that it trusts the authority which issued the token.

The recipient is required to validate the issuer's signature over the token. This validation is required to conform to the core validation rules described in XML Signature Syntax and Processing (Second Edition) by World Wide Web Consortium (W3C) [b-W3C XML signature]. The recipient is recommended to validate the trust semantics of the signing key, as appropriate to the risk of incorrect authentication.

If the message has been signed, then the recipient is required to locate the <ds:Signature> element carried inside the <wsse:Security> header.

Unless the security mechanism is peerSAMLV2, the recipient is required to resolve the contents of the <ds:KeyInfo> element carried within the <ds:Signature> and use the key it describes for validating the signed elements. When the security mechanism is peerSAMLV2, the key is the client key used in SSL/TLS client authentication.

The recipient is required to follow the Message Integrity rules outlined for senders and recipients when message authentication mechanisms are used.

(4) Processing messages with WSS X.509 token

The semantics and processing rules for mechanisms with MESSAGE having the value of X509 are described in this section. An example can be found in Appendix I.

These URIs support unilateral (sender) message authentication and are of the form:

- *urn:liberty:security:2003-08:PEER:X509* where PEER may vary depending on the peer authentication mechanism deployed (e.g., may be null, TLS etc).

The WSS X509 message authentication mechanism uses the Web Services Security X.509 Certificate Token Profile [b-OASIS WSS X.509 profile] as the means by which the message sender authenticates to the recipient. These message authentication mechanisms are unilateral. That is, only the sender of the message is authenticated. It is not in the scope of this document to suggest when response messages should be authenticated but it is worth noting that this mechanism could be relied upon to authenticate the response message as well. It is recommended to recognize, however, that independent authentication of response messages does not provide the same message stream protection semantics as a mutual peer entity authentication mechanism would offer.

For deployment settings that require message authentication independent of peer entity authentication, the sending peer is required to perform message authentication by demonstrating proof of possession of the key associated with the X.509 token. This key is required to be recognized by the recipient as belonging to the sending peer.

When the sender wields the subject confirmation key to sign elements of the message the signature ensures the authenticity and integrity of the elements covered by the signature. However, this alone does not mitigate the threat of replay, insertion and certain classes of message modification attacks. To secure the message from such threats, one of the mechanisms which support peer entity authentication can be used or the underlying SOAP binding request processing model is required to address these threats.

i) Sender Processing Rules

The rules in this section are in addition to the generic message authentication processing rules specified in this document.

The sender is required to demonstrate possession of the private key associated with the signature generated in conjunction with the WSS X509 token profile.

For deployment settings which REQUIRE independent message authentication, the obligation is required to be accomplished by signing portions of the message as appropriate and recording information in the <wsse:Security> header (as outlined in [b-OASIS WSS SOAP]).

For deployment settings which DO NOT REQUIRE independent message authentication, the sender is required to accomplish this obligation by decorating the security header with a <ds:KeyInfo> element bearing the certificate.

This is required to be unambiguously verified to be the same certificate and key used in establishing peer entity authentication. This is necessary to mitigate the threat of a certificate substitution attack. Also, note that this optimization only applies to *ClientTLS:X509* mechanisms.

ii) Recipient Processing Rules

If the validation policy regards peer entity authentication is sufficient for purposes of authentication, then the recipient is required to establish the correspondence of the certificate and key used to establish peer authentication with the corresponding key information conveyed in the message. This allows the message recipient to determine that the message sender intended a particular transport authenticated identity to be used. Information relating the SSL/TLS key to the message MAY be conveyed in the message using an OASIS SOAP Message Security X.509 security token.

6.6 Protection of Personally-Identifiable Information (PII)

According to ITU-T Recommendation Y.2720, *NGN Identity Management* Framework [ITU-T Y.2720] protection of PII is a subject to national and regional regulations. While the mechanisms and procedures employed for supporting the PII protection may vary depending on such regulations, they are based on common principles.

This section provides a brief overview of the procedures for protection of PII that are specified by the NIST Special Report *Guide to Protecting the Confidentiality of Personally Identifiable Information (PII)* [b-NIST-SP 800-122]. The document's specifications of the *PII confidentiality safeguards* could be used as guidance by the designers a of the IdM systems. The following categories of the safeguards are defined:

- Operational Safeguards
    - Policy and Procedure Creation
    - Awareness, Training, and Education
- Privacy-Specific Safeguards
    - Minimizing the Use, Collection, and Retention of PII
    - Conducting Privacy Impact Assessments
    - De-Identifying Information
    - Anonymizing Information
- Security Controls

  The *Security Controls* section provides guidance on the security mechanisms and procedures that are not PII-specific, but can be employed for the PII protection. Similarly, the non-specific to PII security mechanisms specified in [ITU-T Y.2704] can be used for protection of PII.

6.7 Federated Identity Functions

The ITU-T draft Recommendation *NGN Identity Management Requirements and Use Cases* [Y.2721] explains that *the general concept of federation is to allow each federation member to remain independent while facilitating sharing of specific identity information to allow federated services.*

This section recommends the use of two widely implemented standard mechanisms that allow a user to access multiple services without subscribing to each service individually.

The SAML specification [ITU-T X.1141] provides a standard solution for federation. It is used mostly by the businesses, government organizations and their service providers.

OpenID [b-OpenID v.2] specifies a user-centric solution, which is popular for accessing Web services on the Internet.

6.7.1 Bridging and Interworking

This Recommendation describes a number of the mechanisms that support bridging and interworking among different IdM solutions and federations. The major ones are described in the following sections:

- Integration of the PKI-based authentication with IMS (section 6.2.6)

- Integration of the PKI-based authentication and the SAML assertion mechanisms (section 6.2.7)
- Integration of *OpenID*-based authentication with AKA (section 6.2.8)
- Generic Bootstrapping Architecture (section 6.2.9)
- Correlation and Binding (section 6.3)
- Federated Identity Functions (section 6.7)

6.7.2 Discovery of IdSPs in Federated Environment

The SAML specification [ITU-T X.1141] in section 11.4.3 defines *identity provider discovery profile*, which enables a service provider to discover the identity providers of a user. The profile is specified in support of the SAML *Web Browser SSO profile* (defined in section 11.4.1 of [ITU-T X.1141]).

OpenID [b-OpenID v.2] specifies a discovery mechanism that enables a relying party to locate a user's IdSP based on the user-supplied *OpenID* identifier.

6.8 Identity Information Access Control

[ITU-T Y.2721] requires that identity information only be accessible to authorized entities subject to applicable regulation and policy. This section describes mechanisms that can be used to verify authorization privileges.

6.8.1 SAML-based mechanism for attribute sharing

SAML assertions containing the attribute statements can be used as a mechanism for privilege management. The mechanism described in section 6.2.1 can be used for distributing of the SAML tokens.

6.8.2 X.509-based Privilege Management Infrastructure

The attribute certificate framework defined in [ITU-T X.509] can be used as a mechanism for a privilege management infrastructure.

6.9 Single Sign-on

Single sign-on (SSO) is a network capability that enables a user to log in once and obtain access to the multiple application services of a network without being repeatedly requested to provide her or his authentication credentials for each individual application service. This capability significantly improves user experience by enabling a user to receive various services without having to maintain multiple authentication credentials (e.g., username/password pairs). Because the SSO allows a user to have one set of the authentication credentials for accessing multiple application services, it makes it easier for the service providers to enforce more strict rules for establishing the credentials. This helps to improve the network security.

On the other hand, if the user credentials are compromised the impact on the SSO-enabled networks could be greater than on the systems that do not support SSO. To that end, it is essential for the SSO to employ secure mechanisms. This section provides an overview of several mechanisms that can be used for supporting SSO.

6.9.1 GBA-based mechanism

Clause 6.2.9 describes the use of the GBA for authentication of a user to any Network Application Function (NAF). Thus, the GBA effectively provides a single mechanism for signing a user to all GBA-enabled NAFs on a network. Indeed, if a user has been signed-on to a NAF, the BSF and UE have already authenticated each other and established a shared key (Ks). Then the procedure of signing on the user to a next NAF will consist of the steps 1, 3, 4, 5, and 6 (step 2 is skipped), which are described in clause 6.2.9. Again, the procedure results in a secret (Ks_NAF) being shared between the UE and a new NAF. This shared secret can be used for authentication between the UE and NAF.

The GBA-based SSO is recommended for the use in the environments where GBA has been deployed.

6.9.2 SAML-based mechanism

The mechanisms of the SAML-based Single Sign-on (SSO) are specified in section *11.4 SSO Profiles of SAML* of [ITU-T X.1141]. The section defines a set of the SAML profiles supporting SSO, which includes also a *Single logout profile* (section 11.4.4). The profile specifies a procedure that allows a user to logout from all application to which she or he had signed using SSO. SAML v.2 presumes trust relationships between an IdSP and RPs established in advance. It also supports pseudonym identifiers available between an IdSP and an RP. It is suitable for applications where contractual agreement such as SLA, or high value information and transactions are involved.

6.9.3 *OpenID*-based mechanism

OpenID Authentication 2.0 supports Single Sign On capability to allow an end user to access more than one Relying Parties once the end user is successfully authenticated. It does not require trust relationship between IdSP and RP. Since it only supports URL/URI based format to identify users, DNS is necessary for its use. Thus, it is suitable for web services applications where relatively lower value information and transactions are involved.

6.10 Single Sign-off

The *SAML Single Logout Protocol* [ITU-T X.1141], clause 8.2.7 enables End User to sign-off from multiple participating sessions near-simultaneously. The participating session are those that have been established through the IdSP (i.e., the IdSP has asserted the user identity for the sessions between a user and the applications). The IdSP keeps track of all authenticated sessions with various relying parties that a user has established through the IdSP. This includes invalidation of the authentication credentials (e.g., cookies, assertions) for the concluded sessions. The protocol can be used in the following cases:

1. The user signs off from one of the sessions and indicates that she or he wishes to logout of all sessions that have been initiated by the IdSP
2. The user indicates directly to the IdSP that she or he wishes to logout of all sessions
3. The IdSP logs out a user without her or his request (e.g., due to a timeout)

The protocol defines the participating entities, their behavior, the message flow and the format of the exchanged messages. The following clauses describe the use of the protocol for the cases listed above.

6.10.1 The user signs off from one of the sessions and indicates that she or he wishes to logout of all sessions that have been initiated by IdSP

Figure 11 illustrates the basic steps of the message flow, which are described below.

6.10.1.1 Entities involved in the procedure and the information flow

The involved entities are as follows:

- End-User Function.
- Application Server 1 (AS1) — an entity providing a service. It plays a role of a Relying Party. It acts as a SAML requestor and responder as defined in [ITU-T X.1141].
- Application Server 2 (AS2) — an entity providing a service. It plays a role of a Relying Party. It acts as a SAML requestor and responder as defined in [ITU-T X.1141].
- A-2: Application gateway functional entity (APL-GW-FE), which serves as an IdSP and acts as a SAML requestor and responder as defined in [ITU-T X.1141].

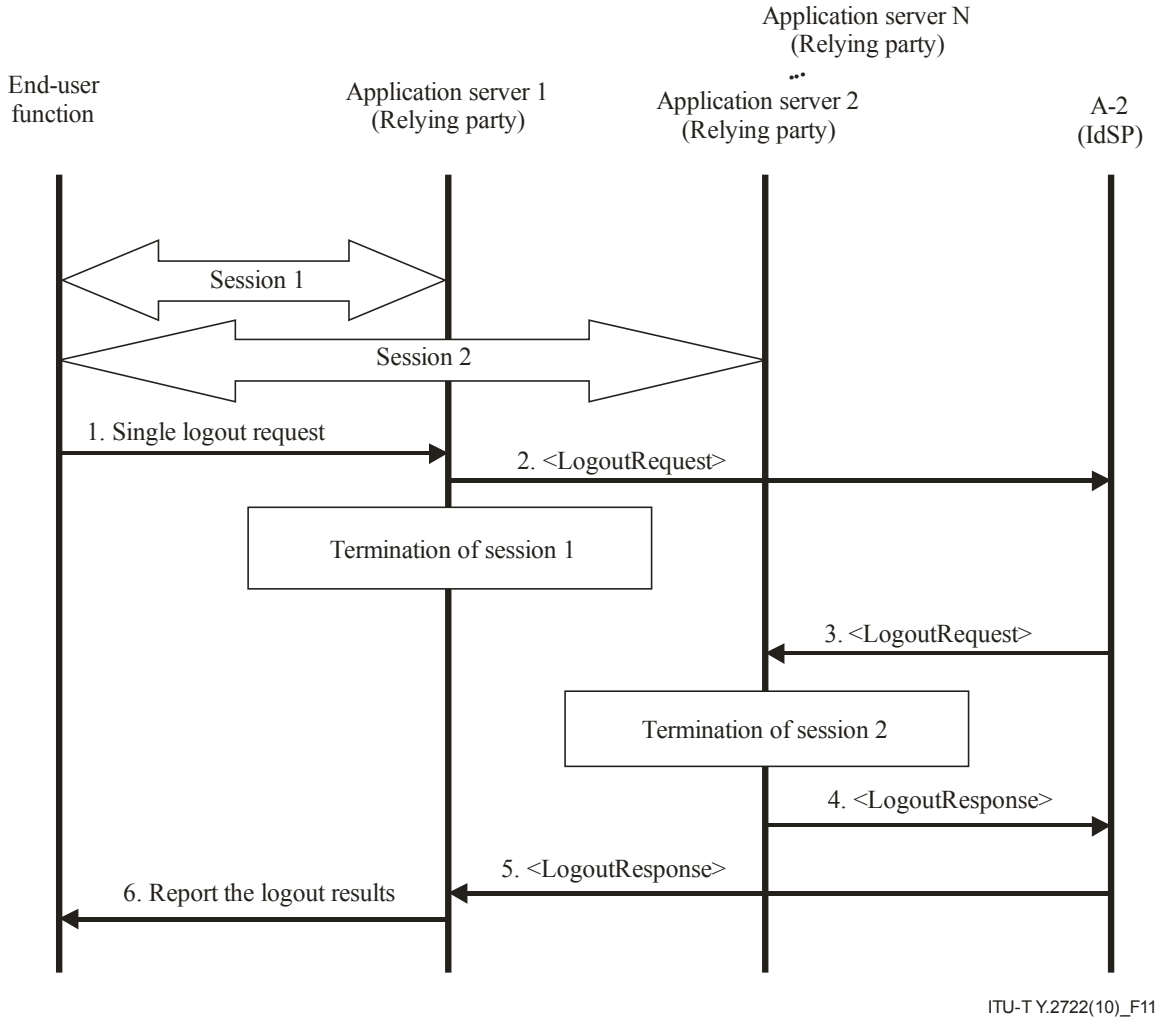


Figure 11– SAML-based Single Sign-off requested by a user at a participating session

The basic steps of the Single Sign-off (also called single logout) procedure are as follows:

1. The End User Function invokes a logout request at the Application Server 1 (AS1) indicating that it wishes to logout from all participating sessions.

2. AS1 requests a logout from all participating sessions by sending <LogoutRequest> to A-2. The request should be signed for authentication and integrity protection as specified in clause 8.2.7 of [ITU-T X.1141].

   After this step AS1 attempts to terminate session 1. To that end AS1 invalidates the session's authentication credentials (e.g., assertions, cookies), which will force the End User Function to go through authentication procedure if it issues another request to AS1.

3. After validating the request from AS1, A-2 sends to all relying parties (only AS2 is shown in Figure 1) the <LogoutRequest> messages. The requests should be signed as specified in clause 8.2.7 of [ITU-T X.1141].

   After validating the logout request AS2 attempts to terminate session 2.

4. AS2 reports to the sender of the logout request (A-2) on the result of the logout attempt by sending <LogoutResponse>, which should be signed.

5. A-2 sends <LogoutResponse> to the initial sender of the logout request (AS1) reporting the results of the single logout (e.g., success, partial logout). The response should be signed.

   After this step A-2 updates its list of the active sessions and invalidates the authentication credentials (e.g., cookies, assertions) for the sessions that had to be terminated.

6. AS1 responds with the logout result to the request of End User Function in step 1.

6.10.2 The user indicates directly to IdSP that she or he wishes to logout of all sessions

Figure 12 illustrates the basic steps of the message flow, which are described below.

6.10.2.1 Entities involved in the procedure and the information flow

The entities involved in the logout procedure are the same as those described in section 6.10.1.1. In this use case, End User Function has a separate session (Session 3) with A-2 (IdSP). It uses the session to send a single logout request in Step 1. Other steps of the procedure are similar to the steps described in clause 6.10.1.1.

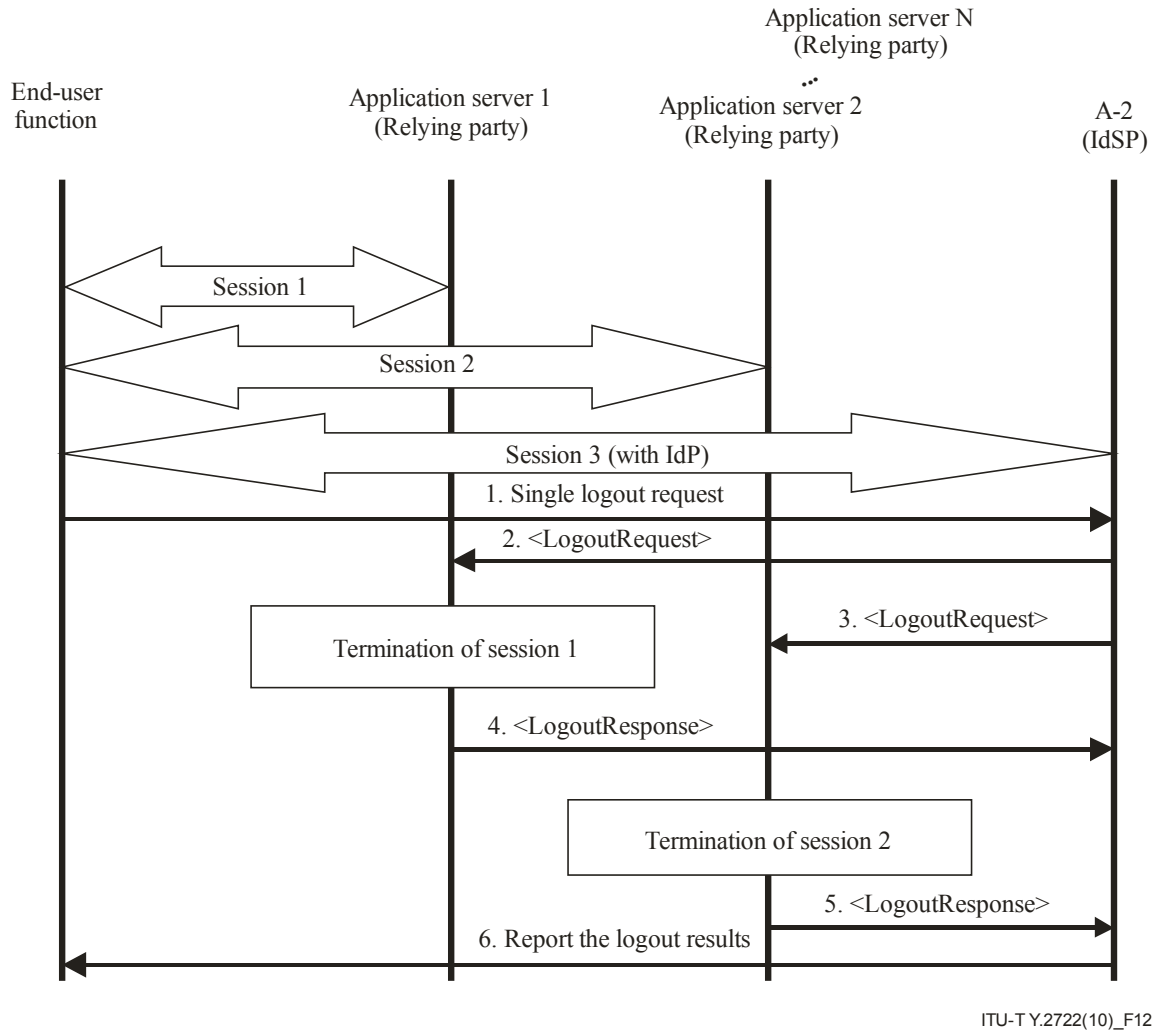


Figure 12– SAML-based Single Sign-off requested by a user at IdSP

The basic steps of the Single Sign-off procedure are as follows:

1. The End User Function invokes a single logout directly at A-2.
2. A-2 sends <LogoutRequest> to AS1. The request should be signed for authentication and integrity protection as specified in clause 8.2.7 of [ITU-T X.1141].

   After validating the request AS1 attempts to terminate session 1. To that end AS1 invalidates the session's authentication credentials (e.g., assertions, cookies), which will force the End User Function to go through authentication procedure if it issues another request to AS1.
3. A-2 sends <LogoutRequest> to AS2 (It also sends the request to all other servers of the participating sessions). This step is similar to step 2.

   After validating the logout request AS2 attempts to terminate session 2.
4. AS1 reports to the sender of the logout request (A-2) on the result of the logout attempt by sending <LogoutResponse>, which should be signed.
5. Similarly to the action in the previous step, AS2 reports to the sender of the logout request (A-2) on the result of the logout attempt by a signed <LogoutResponse>.

   After this step A-2 updates its list of the active sessions and invalidates the authentication credentials (e.g., cookies, assertions) for the sessions that had to be terminated.

After validating all logout responses A-2 reports to End User Function on the single logout result. This is a response to the request of End-User Function in step 1.

7 Security

The mechanisms covered in this Recommendation together with the mechanisms specified in [ITU-T Y.2704] address the IdM security requirements of [ITU-T Y.2721].

Appendix I: WSS X.509 v3 Message Authentication

The following example demonstrates a way to process messages with WSS X.509 token, as described in the section 6.5.1.2 of the main body of this Recommendation.

```
<?xml version="1.0" encoding="UTF-8"?>
<s:Envelope xmlns:s="http://schemas.xmlsoap.org /soap/envelope/"
        xmlns:sb="urn:liberty:sb:2006-08"
        xmlns:pp="urn:liberty:id-sis-pp:2003-08"
        xmlns:sec="urn:liberty:security:20 06-08"
        xmlns:wsse="http://docs.oasis-open.org/wss/2004 /01/oasis-200401-wss-wssecurit y-secext-1.0.xsd"
        xmlns:wsu="http://docs.oasis-open.or g/wss/2004/01/oasis-200401-wss -wssecurity-utility-1.0.xsd"
        xmlns:wsa="http://www.w3.org/2005/08/addressing">

<s:Header>
<!-- see Liberty SOAP Binding Specification for which headers are required and optional -->

<wsa:MessageID wsu:Id="mid">...</wsa:MessageID>

<wsa:To wsu:Id="to">...</wsa:To>

<wsa:Action wsu:Id="action">...</wsa:Action>

<wsse:Security mustUnderstand="1">

        <wsu:Timestamp wsu:Id="ts">
                <wsu:Created>2005-06-17T04:49:17Z</ wsu:Created >
        </wsu:Timestamp>

        <wsse:BinarySecurityToken
                ValueType="http://docs.oasis-open.or g/wss/2004/01/oasis-200401-wss-x509-token-profile-
1.0#X509v3 "
                wsu:Id="X509Token"
                EncodingType="http://docs.oasis-open.org/wss/20 04/01/oasis-200401-wss-soap-message-
security-1.0#Base64Binar y">
                MIIB9zCCAWSgAwIBAgIQ...
        </wsse:BinarySecurityToken>

        <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
                <ds:SignedInfo>

                        <!-- in general include a ds:Reference for each wsa: header added according to SOAP
binding -->

                        <!-- include the MessageID in the signature -->
                        <ds:Reference URI="#mid">...</ds:Reference>

                        <!-- include the To in the signature -->
                        <ds:Reference URI="#to">...</ds:Reference>

                        <!-- include the Action in the signature -->
                        <ds:Reference URI="#action">...</ds:Reference>

                        <!-- include the Timestamp in the signature -->
```

```
                <ds:Reference URI="#ts">...</ds:Reference>

                <!-- bind the security token (thwart cert substitution attacks) -->
                <ds:Reference URI="#X509Token">
                        <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/x mldsig#sha1"/>
                        <ds:DigestValue>Ru4cAfeBABE...</ ds:DigestValue>
                </ds:Reference>

                <!-- bind the body of the message -->
                <ds:Reference URI="#MsgBody">
                        <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig# sha1"/>
                        <ds:DigestValue>YgGfS0pi56pu...</ds:Di gestValue>
                </ds:Reference>
        </ds:SignedInfo>
        <ds:KeyInfo>
                <wsse:SecurityTokenReference>
                        <wsse:Reference URI="#X509Token" />
                </wsse:SecurityTokenReference>
        </ds:KeyInfo>
        <ds:SignatureValue>
                HJJWbvqW9E84vJVQkjjLLA6nNvBX7mY00TZhwBdFNDElgscS XZ5Ekw==
        </ds:SignatureValue>
    </ds:Signature>
</wsse:Security>
</s:Header>

<s:Body wsu:Id="MsgBody">
    <pp:Modify>
        <!-- this is an ID-SIS-PP Modify message -->
    </pp:Modify>
</s:Body>


</s:Envelope>
```

Appendix II: "OpenID+OAuth"-based mechanism for access control

Clause 6.2.7 in the main body of this Recommendation describes the use of the OpenID for authentication of a user to any Network Application Function (NAF). Thus, it proposes to introduce OAuth based on OpenID for protecting the PII and access controlling.

II.1 OAuth [b-IETF RFC 5849]

OAuth is an open protocol enabling an application to access end user information from a Web service when the application is authorized by the end user. The end user's information is securely transferred without revealing the identity of the user.

The goal of OAuth is to acquire an access token from Web Server, which can then be used to exchange user-specific data with a Web service (such as calendar information or an address book). The regular OAuth process is a four-step sequence:

(1) ask for a "request" token

(2) ask for the token to be authorized, which triggers user approval

(3) exchange the authorized request token for an "access" token

(4) use the access token to interact with the user's Web service data.

For more information about the OAuth, refer to [b-IETF RFC 5849].

II.2 Using OpenID in conjunction with OAuth

While OpenID can be used as an IdM mechanism for authenticating users, OAuth could also be used for authorizing permission to sensitive user data. In such scenario, the IdSP provides combined functions and serves both as an OpenID Identity Provider (OP) and an OAuth Service Provider.

II.3 OpenID + OAuth Authorization Flow

With OpenID+OAuth, this sequence remains essentially the same. The difference is that getting an authorized OAuth request token (steps 1 and 2) is wrapped up in the OpenID authentication request. In this way, the user can approve login and service access at the same time.

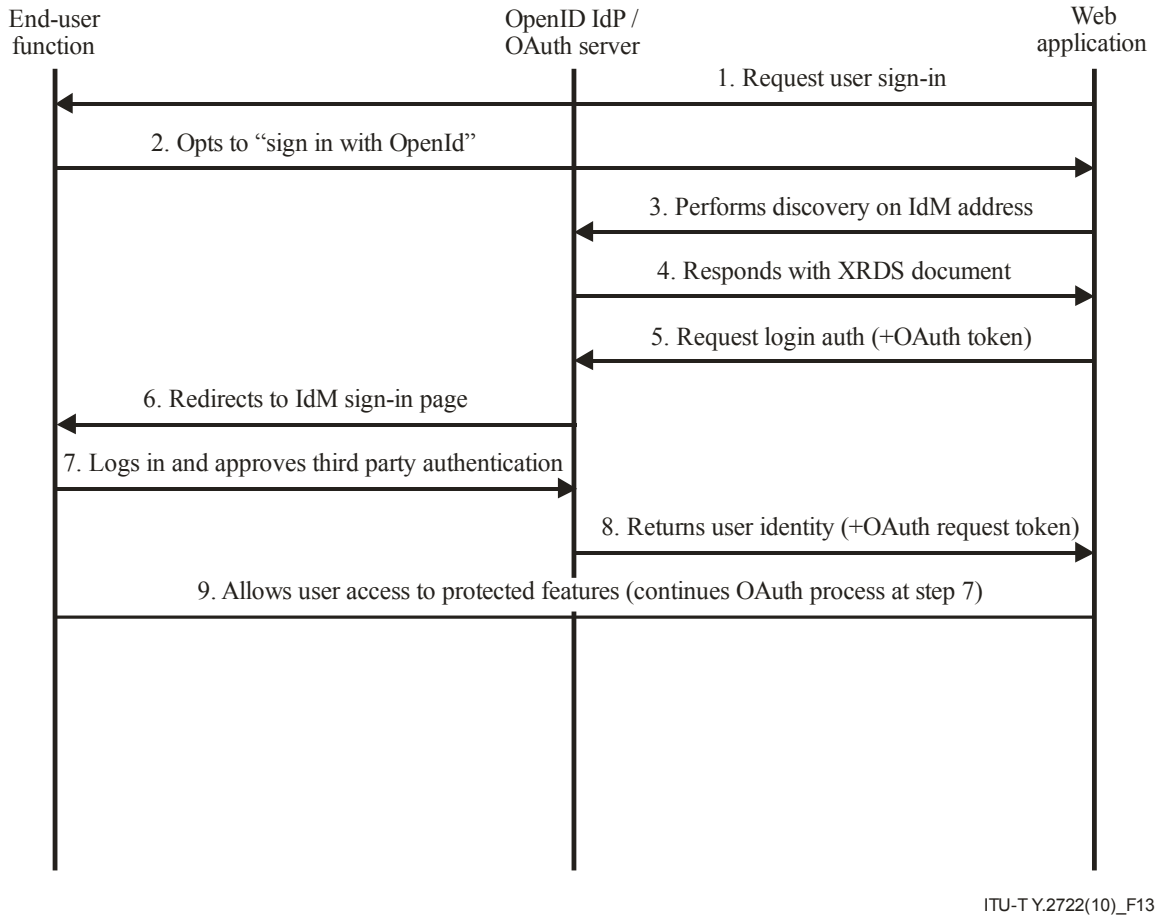


Figure 13 – *OpenID+OAuth*-based authentication

The basic steps are as follows:

1. The web application asks the end user to log in by offering a set of log-in options, including using their OpenID account.
2. The user selects the "Sign in with OpenID" option.
3. The web application sends a "discovery" request to IdSP to get information on the IdSP login authentication endpoint.
4. IdSP returns an XRDS document, which contains the endpoint address.
5. The web application sends a login authentication request to the IdSP endpoint address.
6. This action redirects the user to an IdSP Federated Login page, either in the same browser window or in a popup window, and the user is asked to sign in.
7. Once logged in, IdSP displays a confirmation page and notifies the user that a third-party application is requesting authentication. The page asks the user to confirm or reject linking their IdSP account login with the web application login. The user is then asked to approve access to a specified set of IdSP services. Both the login and user information sharing must be approved by the user for authentication to continue.
8. If the user approves the authentication, IdSP returns the user to the URL specified in the *openid.return_to* parameter of the original request. A IdSP-supplied identifier, which has no relationship to the user's actual IdM account name or password, is appended as the query parameter *openid.claimed_id*. If the request also included attribute exchange, additional user information may be appended. For OpenID+OAuth, an authorized OAuth request token is also returned.

9. The web application uses the IdSP-supplied identifier to recognize the user and allow access to application features and data. For OpenID+OAuth, the web application uses the request token to continue the OAuth sequence and gain access to the user's IdSP services.

Appendix III: Bibliography

[b-ETSI TS 133 220] ETSI TS 133 220 v6.3.0 (2004-12), Universal Mobile Telecommunications System (UMTS) ; Generic Authentication Architecture (GAA) ; Generic bootstrapping architecture.

[b-OpenID v.2] *OpenID Authentication 2.0* < http://openid.net/specs/openid-authentication-2_0.html>

[b-IETF RFC 2616] IETF RFC 2616 (1999), Hypertext Transfer Protocol -- HTTP/1.1 <http://datatracker.ietf.org/doc/rfc2616/>

[b-IETF RFC 2617] IETF RFC 2617 (1999), HTTP Authentication: Basic and Digest Access Authentication <http://datatracker.ietf.org/doc/rfc2617/>

[b-IETF RFC 3310] IETF RFC 3310 (2002), Hypertext Transfer Protocol (HTTP) Digest Authentication Using Authentication and Key Agreement (AKA) <http://www.rfc-editor.org/rfc/rfc3310.txt>

[b-IETF RFC 4169] IETF RFC 4169 (2005), Hypertext Transfer Protocol (HTTP) Digest Authentication Using Authentication and Key Agreement (AKA) Version-2 http://www.ietf.org/rfc/rfc4169.txt?number=4169

[b-IETF RFC 4279] IETF RFC 4279 (2005), Pre-Shared Key Ciphersuites for Transport Layer Security (TLS) http://datatracker.ietf.org/doc/rfc4279/

[b-IETF RFC 5849] IETF RFC 5849 (2010), The OAuth 1.0 Protocol http://tools.ietf.org/html/rfc5849

[b-OASIS WSS X.509 profile] OASIS (2006), Web Services Security X.509 Certificate Token Profile 1.1

[b-OASIS SAML token] OASIS (2006), SAML Token Profile 1.1, Approved Erratta 1

[b-OASIS WSS SOAP] OASIS Web Services Security (WSS) SOAP Message Security 1.1 (2004)

[b-LA WSF] Liberty Alliance (2008), Identity Web Services Framework: A Technical Overview, http://www.projectliberty.org/liberty/content/download/4120/27687/file/idwsf-intro-v1.0.pdf

[b-LA ID-WSF security] Liberty ID-WSF Security Mechanisms Core version 2.0-errata version 1.0 (2007), Liberty Alliance Project

[b-LA SOAP binding] Liberty SOAP Binding Version 2.0,2006, Liberty Alliance Project Web Services Security (WSS)

[b-W3C XML signature] World Wide Web Consortium (W3C) (2008), XML Signature Syntax and Processing (Second Edition)

[b-3GPP TR 33.924] 3GPP TR 33.924 Release 9 (2009), 3rd Generation Partnership Project; Identity management and 3GPP security interworking; Identity management and Generic Authentication Architecture (GAA) interworking (Release 9)

[b-NIST-SP 800-122] NIST Special Publication SP 800-122 (2010), Guide to Protecting the Confidentiality of Personally Identifiable Information (PII), http://csrc.nist.gov/publications/nistpubs/800-122/sp800-122.pdf

---