3GPP TR 33.895 V0.7.0 (2013-10)

Technical Report

3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Study on Security aspects of integration of Single Sign-On (SSO) frameworks with 3GPP operator-controlled resources and mechanisms; (Release 12)

MCC selects keywords from stock list.

Keywords
<keyword[, keyword]>

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
Intpp.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

This clause is optional. If it exists, it is always the second unnumbered clause.

1 Scope

The present study investigates the security aspects of the service requirements specified by SA1 in TS 22.101 [11], on the integration of SSO frameworks with 3GPP networks for various operator authentication configurations (e.g. configurations using GBA or not using GBA).

In particular, this study evaluates existing interworking solutions between SSO frameworks and 3GPP authentication mechanisms against the SA1 service requirements. The study is not limited to evaluation of existing interworking solutions and new interworking solutions may be developed as appropriate.

The study covers the security requirements to enable the operator to become the preferred SSO Identity Provider by allowing the usage of credentials on the UE for SSO services, as well as ways for the 3GPP operator to leverage its trust framework and its reliable and robust secure credential handling infra-structure to provide SSO service based on operator-controlled credentials.

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 TR 22.895: "Study on Service aspects of integration of Single Sign-On (SSO) frameworks with 3GPP operator-controlled resources and mechanisms".

[3] 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)".

[4] 3GPP TR 33.924: "Identity management and 3GPP security interworking; Identity management and Generic Authentication Architecture (GAA) interworking".

[5] 3GPP TR 33.804: "Single Sign On Application Security for Common IMS – based on SIP Digest".

[6] 3GPP TS 33.220: "Generic Authentication Architecture (GAA); Generic Bootstrapping Architecture".

[7] 3GPP TS 24.109: "Bootstrapping interface (Ub) and network application function interface (Ua); Protocol details".

[8] 3GPP TS 29.109: "Generic Authentication Architecture (GAA); Zh and Zn Interfaces based on the Diameter protocol; Stage 3".

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

[10] 3GPP TS 33.222, “Access to network application functions using Hypertext Transfer Protocol over Transport Layer Security (HTTPS)“

[11] 3GPP TS 22.101, “Service aspects; Service principles”.

[12] 3GPP TR 33.905, "Recommendations for trusted open platforms".

[13] OpenID Foundation "OpenID Provider Authentication Policy Extension 1.0", http://openid.net/.

3 Definitions, symbols and abbreviations

Delete from the above heading those words which are not applicable.

Clause numbering depends on applicability and should be renumbered accordingly.

3.1 Definitions

For the purposes of the present document, the terms and definitions given in TR 21.905 [1], TS 22.101 [11] 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].

Definition format (Normal)

<defined term>: *<definition>.*

example: text used to clarify abstract rules by applying them literally.

3.2 Symbols

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

Symbol format (EW)

<symbol> <Explanation>

3.3 Abbreviations

For the purposes of the present document, the abbreviations given in TR 21.905 [1], TS 22.101 [11] 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].

Abbreviation format (EW)

<ACRONYM> <Explanation>

4 Relation of the present study to other related work in 3GPP

Other SSO related work in 3GPP

Completed SA1 work

- SSO requirements, TS 22.101 [11];
- Study on integration of SSO frameworks with 3GPP, TR 22.895[2].

Completed SA3 work

- Liberty - GBA interworking, TR 33.980 [3];

- OpenID – GBA interworking, TR 33.924 [4].
- SSO with SIP Digest, TR 33.804 [5].

What is the relation of this study to other work in 3GPP

This study should evaluate the completed and ongoing SA3 SSO work against the service requirements identified bySA1.

All input in this study should have a clear relation to the SA1 service requirements. This study should not duplicate functionality supporting SA1 service requirements, when such functionality can be offered by existing SSO mechanisms. In particular existing solutions in other SA3 specifications are evaluated and new ones should be proposed only if the existing solutions would not meet the SA1 service requirements.

5 Findings in SA1 study

Editor's Note: The purpose of this clause is to capture the findings of SA1 study, e.g. service requirements and the use cases of integration of different Identity and SSO frameworks for various operator authentication configurations. The findings of SA1 may also be evaluated from security perspective in the present clause. Copy-pasting of SA1 study content should be avoided.

6 Requirements identified in the present study

Editor's Note: The purpose of this clause is to identify possible security requirements in the present study, if any. The requirements may be general or specific to identified SSO frameworks as seen appropriate.

Requirement:

Rationale for the requirement:

7 Solutions for Liberty Alliance/SAML – 3GPP interworking

Editor's Note: The purpose of this clause is to describe existing (and possible new) solutions for interworking of Liberty Alliance/SAML and 3GPP authentication mechanisms and evaluate the solutions against the findings in SA1 study.

7.1 General

7.2 <Solution X>

7.2.1 Rationale for solution

Editor's Note: The purpose of this clause is to justify why the solution should be considered in the present study, i.e. if it is an existing solution, or in case of a new solution there needs to be justification why a new solution is proposed.

7.2.2 Solution description

7.2.3 Evaluation against findings in SA1 study

8 Solutions for OpenID – 3GPP interworking

Editor's Note: The purpose of this clause is to describe existing (and possible new) solutions for interworking of OpenID and 3GPP authentication mechanisms and evaluate the solutions against the findings in SA1 study.

8.1 General

8.2 GBA Lite

8.2.1 Rationale for solution

SSO has been identified as one of the most promising applications of GBA. Clearly, the value of this use-case for an external service provider depends on the number of supporting users. This number in turn depends on the availability of GBA-capable phones and the number of operators which have deployed the necessary GBA infrastructure

One way to overcome the initial threshold of supporting users is to simplify the deployment process. This is accomplished using an SSO specific implementation option of GBA called– GBA Lite. Later on, if an operator finds a need to support other applications as well, the SSO specific version can be extended to full GBA.

The solution presented here closely follows the GBA and OpenID interworking described in 3GPP TR 33.924 [4]. The difference is that the BSF and OP are co-located and hence the Zn interface is a matter of internal implementation. This results in a simpler implementation and deployment. All other nodes and interfaces remain unchanged.

The design goals for GBA Lite were the following:

- A simple migration path to use of full GBA
- The Client/UE and RP (Relying Party) follow TR 33.924 [4] without impact
- Aim for simplicity: keep only the core BSF functionality, remove the rest.

8.2.2 Solution description

8.2.2.1 Architecture

The architecture is identical to 3GPP TR 33.924 [4] Figure 4.3-1 except for the co-location of BSF and OP and the consequent internalization of the Zn interface.

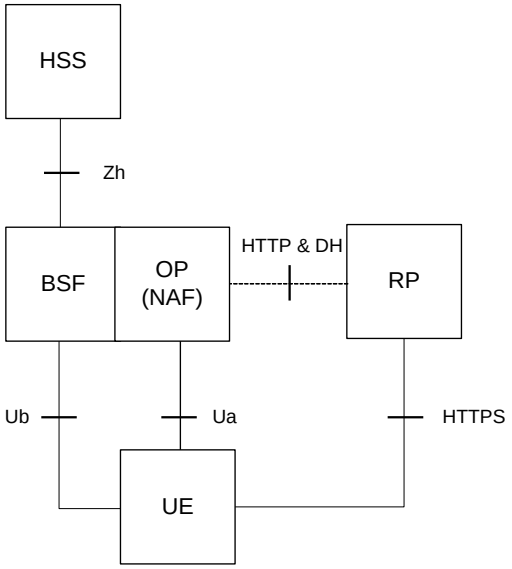

Figure 8.2.2.2-1 GBA Lite Network Architecture

8.2.2.2 BSF Implementation optimizations

No GUSS handling

In ordinary GBA the BSF has to support a wide range of applications with varying options and permissions. In GBA Lite, however, there is only one application: OpenID. This allows us to simplify both the handling of keys and of GBA user security settings (GUSS).

Key handling can be simplified since we only need to deal with OpenID specific keys. For example, the NAF identifier used in the key derivation can be static instead of dynamically determined at the run of the Zn protocol.

The information contained in the GUSS (key lifetime, UICC type, MSISDN etc) can either be statically encoded (key lifetime) or stored as part of the OpenID user account (UICC type, MSISDN). Typically, the OP will maintain a user account for each of its users where the OpenID identifier, attributes, and settings are stored.

The Zh interface should be utilized with minimal effort i.e. no support of GBA User Security Settings (GUSS) is required.

Zn implementation options

Since the Zn interface is internal the vendor or operator is free to choose whatever modifications and optimizations it sees fit. For example, the BSF can be made stateless if the bootstrapping information (B-TID, keys, etc) is pushed over Zn and stored in the OP database. Another option is to use a common database backend and replace Zn with two database calls. Of course, one could also choose not to make any changes and implement the standard Zn interface. The latter approach makes it easier to migrate to full GBA in the future.

8.2.2.3 Message Flow

The following message flow is identical to the Direct Interworking Scenario in TS 33.924 [4] except for the B-TID lookup (step 8 below) and a slightly different wording.

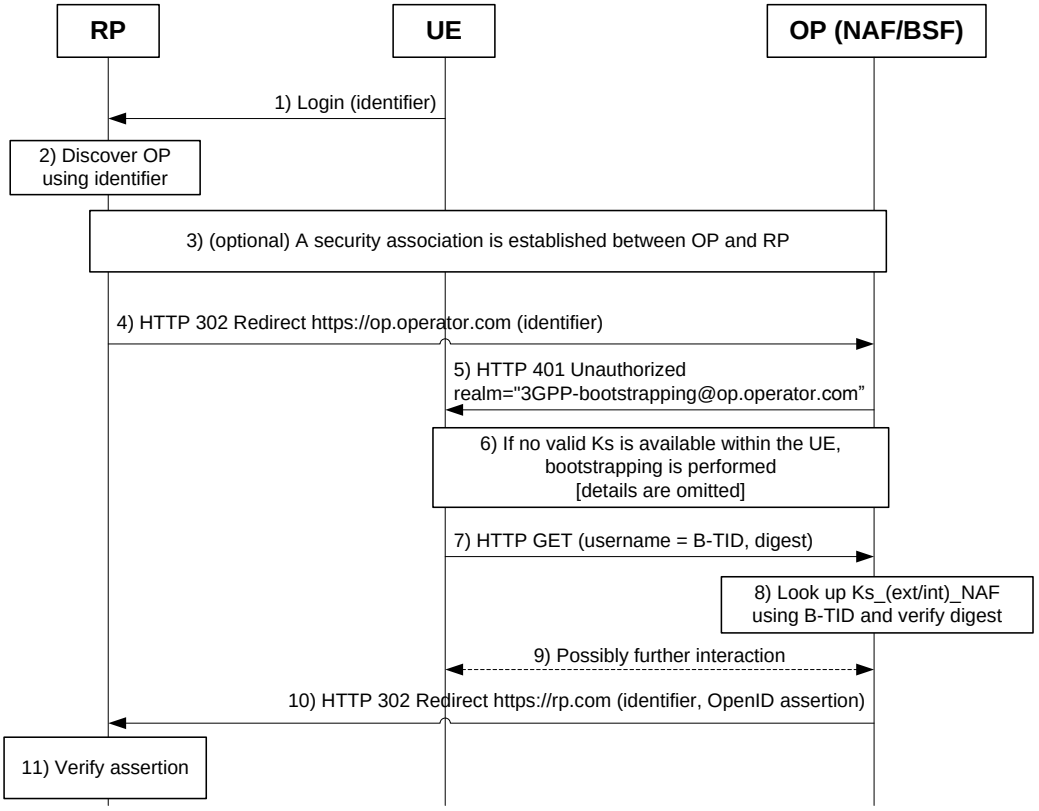

Figure 8.2.2.3-1 Interworking message flow for GBA / OpenID

1. The user initiates authentication by presenting a User-Supplied Identifier to the Relying Party via their User-Agent

2. After normalizing the User-Supplied Identifier, the Relying Party performs discovery on it and establishes the OP Endpoint URL that the end user uses for authentication.

3. (optional) The Relying Party and the OP establish an association – a shared secret established using Diffie-Hellman Key Exchange. The OP uses an association to sign subsequent messages and the Relying Party to verify those messages; this removes the need for subsequent direct requests to verify the signature after each authentication request/response.

4. The Relying Party redirects the end user's User-Agent to the OP with an OpenID Authentication request (Requesting Authentication).

5. The OP (NAF) initiates the UE authentication and responds with a HTTPS response code 401 "Unauthorized", which contains a WWW Authenticate header carrying a challenge requesting the UE to use Digest Authentication with GBA as specified in TS 33.222 [10] with server side certificates.

6. If no valid Ks is available, then the UE bootstraps with the BSF as described in TS 33.220, which results in the possession of the UE of a valid Ks. From this the UE can derive the application specific (OpenID specific) Ks_(ext/int)_NAF key(s).

7. The UE generates a HTTP GET request to the NAF. The HTTP request carries an authorization header containing the B-TID received from the BSF and a response digest.

8. Using the B-TID the NAF retrieves the shared application specific NAF key and validates the response digest.

   Note: Since BSF–OP/NAF interface is internal, several implementation options are possible. E.g. the standard Zn interface could be implemented.

9. Possibly further interaction where e.g. the user is made aware that he is logging in to RP with OpenID.

10. The OP redirects the end user's User-Agent back to the Relying Party with either an assertion that authentication is approved or a message that authentication failed.

11. The Relying Party validates the assertion received from the by using either the shared key established during the association or by sending a direct request to the OP. If the validation is successful, then the user is logged in to the service of the RP.

8.2.3 Evaluation against findings in SA1 study

The collocated GBA architecture shows an easy entry solution for an operator that has not yet deployed GBA, but would like to have an extensible system.

8.3 Third Party IdP binding for two-factor authentication

8.3.1 Rationale for solution

Enterprises and "Over-The-Top" application services providers (OTT) need a means of asserting users' identities for their subsequent authorization. Current use of user ID/password credentials is considered as inadequate security for value added applications such as mobile payments and access to enterprise applications.

The most widespread two-factor authentication is based on the user's ID/password as a first authentication factor (for user's presence authentication) as well as a hardware-based token as a second authentication factor (confirming a user's possession of a physical entity such as a token or device on which such token functionality resides).

When a smartphone containing UICC mutually authenticates with its MNO, reuse of the user's UICC as a second authentication factor allows MNOs to become ID Providers (IDP) and inherently provide more security than the sole use of user ID/password credentials. Existing 3GPP SSO solutions do not provide a means to confirm the presence of a registered user of a data application, nor do they provide a means for binding (e.g. cryptographically) the results of two discrete authentication mechanisms.

Traditionally, 3GPP was focusing on the developing the means to authenticate subscriptions, rather than subscribers (i.e., presence of registered users). Existing SSO solutions do not provide adequate mechanisms to confirm presence of a registered user, since it is the subscription credentials (vs. User credentials) that are being authenticated by existing SSO solutions.

Some of the existing solutions may be deemed capable of providing means for two-factor authentication. Their analysis is presented below.

GBA – Liberty interworking via using GBATwoFactor authentication as described in TS 29.109

TR 33.980 [3] describes 3GPP framework for GBA-Liberty Alliance interworking while not having specific provisions for multi-factor authentication. TS 29.109 [8] in its informative Annex E defines the following information elements and with Associated 3GPP URIs and Class schemas for invoking two-factor authentication using interworking with Liberty Alliance:

GBATwoFactorUnregistered

GBATwoFactorContract

It is, however, unclear how such authentication proceeds, what entity is the Master IDP, and how the binding of authentication factors is being achieved. It is presumed that such binding is possible to accomplish.

GBA – OpenID interworking via using PAPE extensions

PAPE (Provider Authentication Policy Extension) [13] defines a mechanism which allows an OpenID Relying Party to achieve the following:

- request identity providers to use specific authentication policies when authenticating a user.
- require an identity provider to inform the relying party of the authentication policies used during authentication.

- require an identity provider to communicate the levels of authentication used as defined in sets of requested custom assurance levels.

It is possible to use PAPE for the GBA service to request, and to successfully perform GBA authentication. It seems reasonable to have both factors authenticated either in sequence or concurrently. However, PAPE does not seem to provide a mechanism to bind authentication processes for different factors. While PAPE is defined outside of 3GPP, such binding mechanism arguably needs to be defined in 3GPP to be successfully used for multi-factor authentication by 3GPP operators.

SA1 Service Requirements to be taken into SA3 consideration

As part of the technical specification work for Rel-12, 3GPP SA1 defined requirements (see TS 22.101 section 26.1) on providing Single Sign-On service for the UE and the SSO Provider. One of the requirements states that the UE and the SSO Service Provider have mechanisms in place in order to confirm the presence of a registered user of a data application.

In addition, the 3GPP SSO Service is required to support flexibility regarding user configuration of third party SSO identities in the process of gaining access to a service using 3GPP SSO Service. It is required to interwork with such SSO technologies as OpenID (see TS 22.101 section 26.1).

MNO Benefits

Customer records are the biggest MNO asset, together with the MNO's ability to authenticate subscriptions based on AKA credentials residing in the MNO network and UICC. When presence of the user's UICC in the smartphone is verified to serve as a second authentication factor, the MNO becomes an IDP. MNO-provisioned IDP services, anchored on the trust in the MNOs, can be revenue-producing and more importantly, allow MNOs to leverage their ability to provide value-adding authentication services to either over-the-top application services or to enterprises.

Application Services/Enterprise Benefits

Over-the-top application services and enterprises need a secure way of authenticating their users. Two-factor authentication, with user ID/password as the first factor and possession of a token as the second factor, is considered to be a strong form of user authentication.

8.3.2 Solution 1 description

8.3.2.1 General

Example high-level Flow: OTT as a master IDP and MNO as authenticator for factor 2. Figure 8.3.2-1

A User attempts to login to an application service (or to an enterprise network) requiring two-factor authentication.

Upon verification of the first authentication factor by an over-the-top (OTT) application service, the OTT initiates a second factor authentication (token-based) with the user's MNO.

When the second factor authentication is completed, the results of the two authentications (from the OTT based on the first factor and from the user's MNO based on the second factor) are bound together by the OTT. Such authentication binding may be achieved either cryptographically or on the protocol level.

1) User Authentication: OTT performs first factor authentication (e.g. using UID/Password) and decides, based on policy, whether to proceed with a second authentication factor;

2) Second Factor Authentication: OTT forwards a request to the Browser Agent for second factor authentication;

3) UE Authentication Request: Browser Agent forwards authentication request to the UE;

4) UICC based Authentication: GBA based authentication occurs based on AKA credentials;

5) Send Result to OTT: Upon successful completion of Step 4, OP/NAF (MNO) asserts UE Identity to the OTT. The functional interface between OP/NAF/MNO and RP/OTT can be realized via OpenID indirect requests using HTTP re-direct;

6) Conclude Second Factor Authentication: OTT receives confirmation of second authentication factor and binds the two authentication factors.

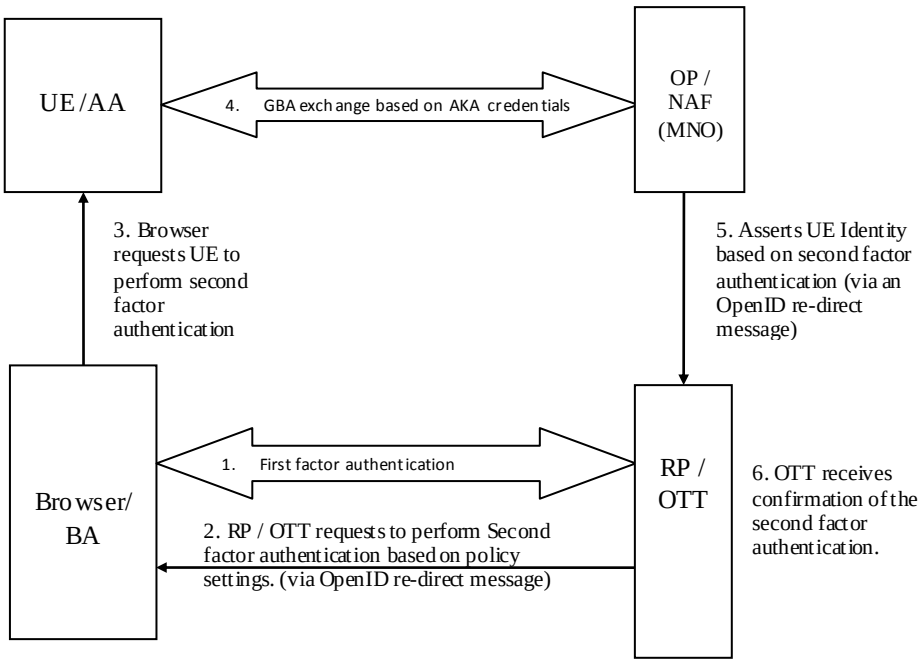

Figure 8.3.2.1-1

Steps 1 through 6 create a "proof of possession", thus providing two-factor authentication for the OTT.

Caching/storing user identity credentials (e.g., user identity and password) in the browser has to be avoided since such caching can potentially interfere with confirming the presence of the "registered user of the data application" and effective user authentication. Preventative measures against storage/caching of user identity credentials may include the use of a freshness indication (e.g. when the password was supplied by the user) in the authentication protocol by utilizing appropriate policies. Defining such preventative measures is outside of the scope of this TR.

Example high-level flow: MNO as Master IdP (authenticator for factor 1 & 2). Figure 8.3.2-2.

A User attempts to login (using MNO credentials) to an over-the-top application service (or to an enterprise network).

The OTT, based on policies, determines that two-factor authentication is required and requests the User to perform two-factor authentication with the MNO that works as the master IdP.

Upon verification of the first authentication factor by the OP / NAF, the MNO initiates a second factor authentication (token-based).

When the second factor authentication is completed, the results of the two authentications (first factor based on the User authentication and second factor based on the user's UICC-) are bound together. Such authentication binding may be achieved either cryptographically or on the protocol level.

1) OTT/RP decides, based on its policy, to request the User to perform two-factor authentication;

2) User Authentication: MNO/OP/NAF performs first factor authentication e.g. using UID/Password;

3) UE Authentication Request: Browser forwards authentication request to the UE;

4) UICC based Authentication: GBA based authentication occurs based on AKA credentials;

5) Bind UE/AA and Browser/BA: Upon successful completion of Step 4 and step 2, OP/NAF (MNO) asserts User and UE Identity based on the success of two-factor authentication. The functional interface between OP/NAF/MNO and RP/OTT can be realized via OpenID indirect request using HTTP re-direct;

6) Conclude Second Factor Authentication: OTT receives confirmation of second authentication factor.

This high-level message flow example is amplified in Section 8.3.2.2.

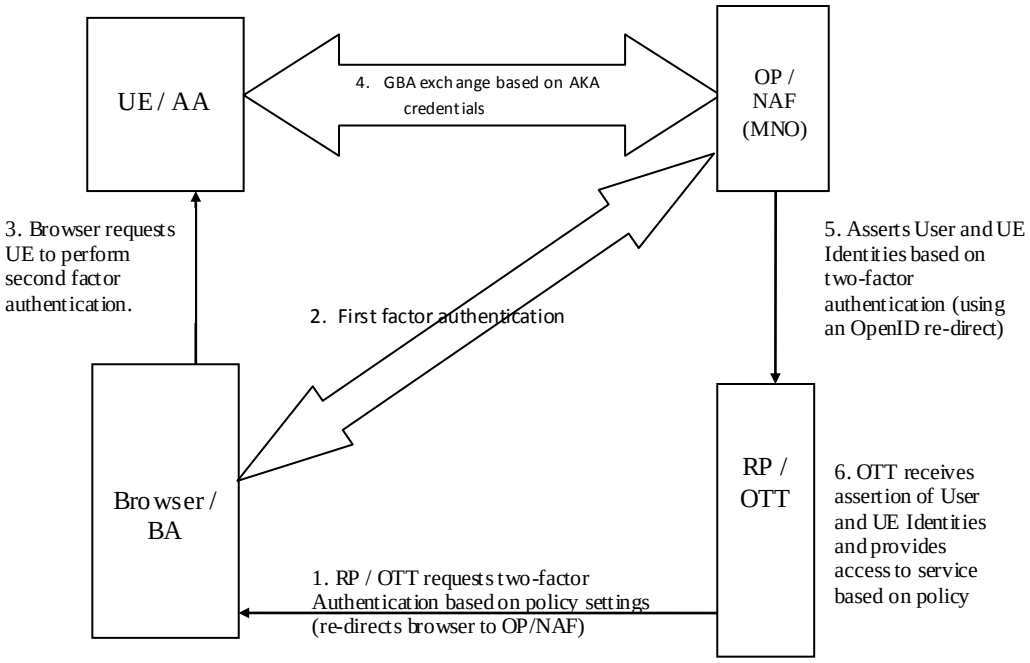

Figure 8.3.2.1-2

8.3.2.2 Example solutions for two factor authentication

Variant 1,

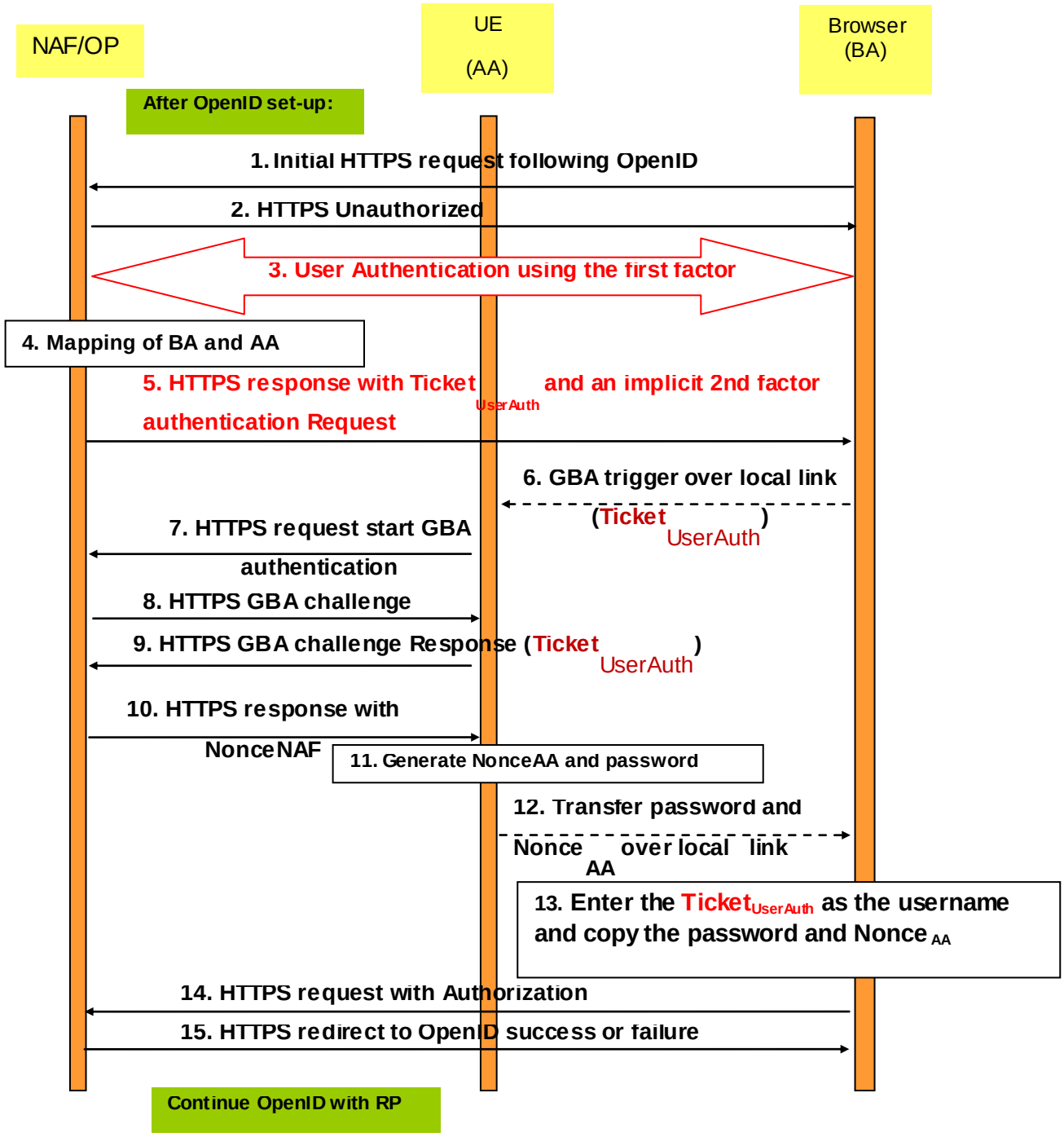

Figure 8.3.2.2 -1

Detailed call flow description

For better understanding of the higher-level diagram 8.3.2-1 and the detailed call flow presented here, note that the RP or Service Provider may be an OTT and OP/NAF may be a MNO.

After the OpenID setup as per specification:

1. Initial HTTPS request following OpenID redirect (same as in TR 33.924, Section 4.4.2. , Fig. 4.4.2.4-3)
2. HTTP Unauthorized Response (same as in TR 33.924, Section 4.4.2. , Fig. 4.4.2.4-3)

3. Message 3 is an aggregate of more than one actual message. It is shown as a single message for simplicity with the intent of being agnostic to any particular authentication mechanism or protocol. User proceeds with the First Factor authentication to OP (e.g., user ID and password). Note that the first factor freshness, e.g. password being cached in the Browser, etc. has to be addressed by the OP policy. To enforce such policies, a Trusted Execution Environment (TEE), a TPM or a similar trusted entity such as a UICC may be needed for policy control (e.g., Policy Enforcement Point and Policy Decision Point.) The way OP addresses enforcement of policies is outside of this Technical Report's scope. Upon successful first factor authentication, a HTTP request is sent by the BA to the OP/NAF requesting a Ticket. This HTTP request is an implicit request within Message 3.
4. Mapping of BA and AA is performed at the NAF / OP.
5. The OP generates a $Ticket_{UserAuth}$ (e.g. a nonce) and sends it within the HTTPS response message, which is in response to the HTTPS request that was sent by the BA as part of the Message 3 exchange. Sending of the $Ticket_{UserAuth}$, has to be interpreted as an implicit request for second factor authentication. Response to this request message is Message 12.
6. GBA is triggered by Message 6, carrying $Ticket_{UserAuth}$ from the Browser (BA) to the UE (AA). This message is corresponding to the (analogous to message in TR 33.924, Section 4.4.2., Fig. 4.4.2.4-3)
7. HTTPS request start GBA authentication (same as in TR 33.924, Section 4.4.2., Fig. 4.4.2.4-3)
8. HTTPS GBA challenge (same as in TR 33.924, Section 4.4.2., Fig. 4.4.2.4-3)
9. HTTPS GBA challenge Response carrying $Ticket_{UserAuth}$ with B-TID from the UE (AA) to the NAF/OP. This message is corresponding to the (analogous to message in TR 33.924, Section 4.4.2. , Fig. 4.4.2.4-3). At this time NAF/OP received $Ticket_{UserAuth}$ and is able to verify that the second factor authentication (UICC-based) is bound to the first factor in Step 3.
10. NAF/OP responds with a NonceNAF
11. The AA generates $Nonce_{AA}$ and uses the NonceNAF and $Nonce_{AA}$ in order to generate a password.
12. The password and $Nonce_{AA}$ is copied over a local link to the BA.
13. Copy NonceAA as Username and password received over the local link
14. Steps 14-15 are reproduced here only for referential integrity with the Solution 3 from TR 33.924. They are not germane for the purpose of this Section.

Variant 2.

For better understanding of the higher-level diagram 8.3.2-1 and the detailed call flow presented here, note that the RP or Service Provider may be an OTT and OP/NAF may be a MNO.

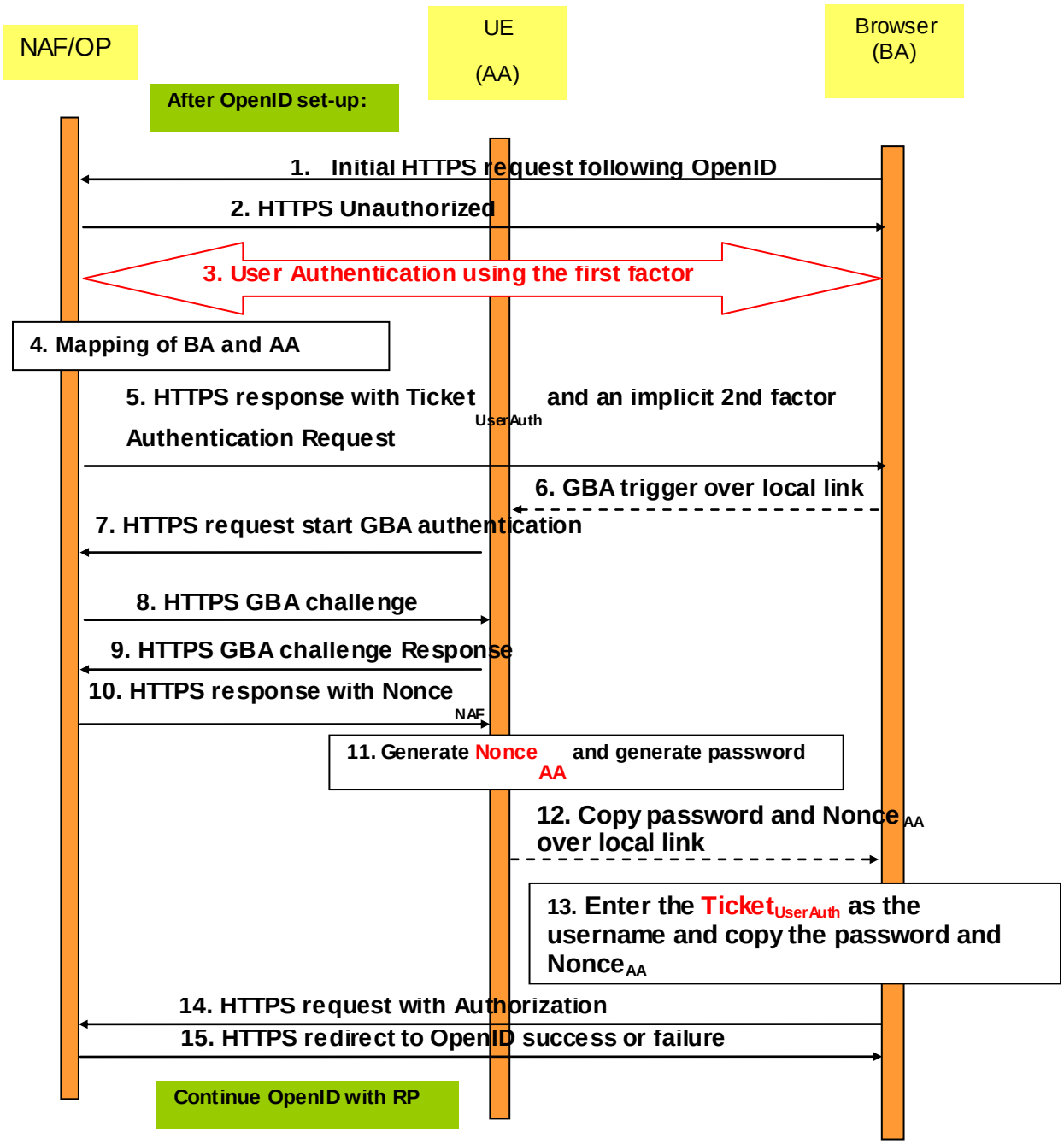

Figure 8.3.2.1-2

After the OpenID setup as per specification:

1. Initial HTTPS request following OpenID redirect (same as in TR 33.924, Section 4.4.2. , Fig. 4.4.2.4-3)
2. HTTP Unauthorized Response (same as in TR 33.924, Section 4.4.2. , Fig. 4.4.2.4-3)
3. Message 3 is an aggregate of more than one actual message. It is shown as a single message for simplicity with the intent of being agnostic to any particular authentication mechanism or protocol. User proceeds with the First Factor authentication to OP (e.g., user ID and password). Note that the first factor freshness, e.g. password being cached in the Browser, etc. has to be addressed by the OP policy. To enforce such policies, Trusted Execution Environment, similar to the UICC may be needed for execution of policy

control and enforcement (e.g., Policy Enforcement Point and Policy Decision Point.) The way OP addresses the first factor freshness e.g. password being cached in the Browser, etc. is outside of this Technical Report's scope. Upon successful first factor authentication, a HTTP request is sent by the BA to the OP/NAF requesting a Ticket. This HTTP request is an implicit request within Message 3

4. Mapping of BA and AA is performed at the NAF / OP
5. The OP generates a TicketUserAuth (e.g. a nonce) and sends it within the HTTPS response message, which is in response to the HTTPS request that was sent by the BA as part of the Message 3 exchange. Sending of the $Ticket_{UserAuth}$, has to be interpreted as an implicit request for second factor authentication. While Message 12 is the response to this request.
6. GBA is triggered by Message 6. This message is corresponding to the (analogous to message in TR 33.924, Section 4.4.2. , Fig. 4.4.2.4-3)
7. HTTPS request start GBA authentication (same as in TR 33.924, Section 4.4.2. , Fig. 4.4.2.4-3)
8. HTTPS GBA challenge (same as in TR 33.924, Section 4.4.2. , Fig. 4.4.2.4-3)
9. HTTPS GBA challenge Response with B-TID from the UE (AA) to the NAF/OP. This message is corresponding to the (analogous to message in TR 33.924, Section 4.4.2.4, Scenario 3).
10. NAF/OP responds with a NonceNAF
11. The AA uses the NonceNAF and $Nonce_{AA}$ in order to generate a password.
12. The password and $Nonce_{AA}$ is copied over a local link to the BA.
13. The $Ticket_{UserAuth}$ is copied into the Username field while the password and $Nonce_{AA}$ received over local link is copied into the Password field. The functionality of $Nonce_{AA}$ and NonceNAF are dedicated to binding AA with BA, and preserved in this example for conformance with the solution described in TR 33.924. The functionality of TicketUserAuth is devoted to binding authentications procedure/for the 1st Factor with the authentication procedure for the 2nd factor.

14. Steps 14-15 are reproduced here only for referential integrity with the Solution 3 from TR 33.924. They are not germane for the purpose of this Section.

8.3.3 Solution 2 description

8.3.3.1 Solution based on OpenID-GBA interworking where OTT performs username/password authentication

The solution presented here is based on OpenID – GBA interworking. Two factor authentication is achieved by the additional step in the beginning where the RP authenticates the user using username/password. Provided the first factor authentication is successful, the RP will redirect the user to the IdP for the second factor GBA based authentication. Once the authentication is done the IdP sends an OpenID token back to the RP via the user, asserting the user's identity.

Since the RP receives the username/password and OpenID token in the same TLS tunnel/HTTP session it is assured that the they were both provided by the same entity, In other words the "binding" between the first and second factor of authentication is accomplished by the TLS tunnel/HTTP session.

A benefit of this solution is that it requires no additional standardization. This is because the first factor of authentication and the binding is handled by the RP on its own, and the RP is not a 3GPP entity.

A high-level call flow is presented below. Note that the order in which the authentications are performed does not matter, An alternative flow would be to perform the username/password authentication after the OpenID authentication.

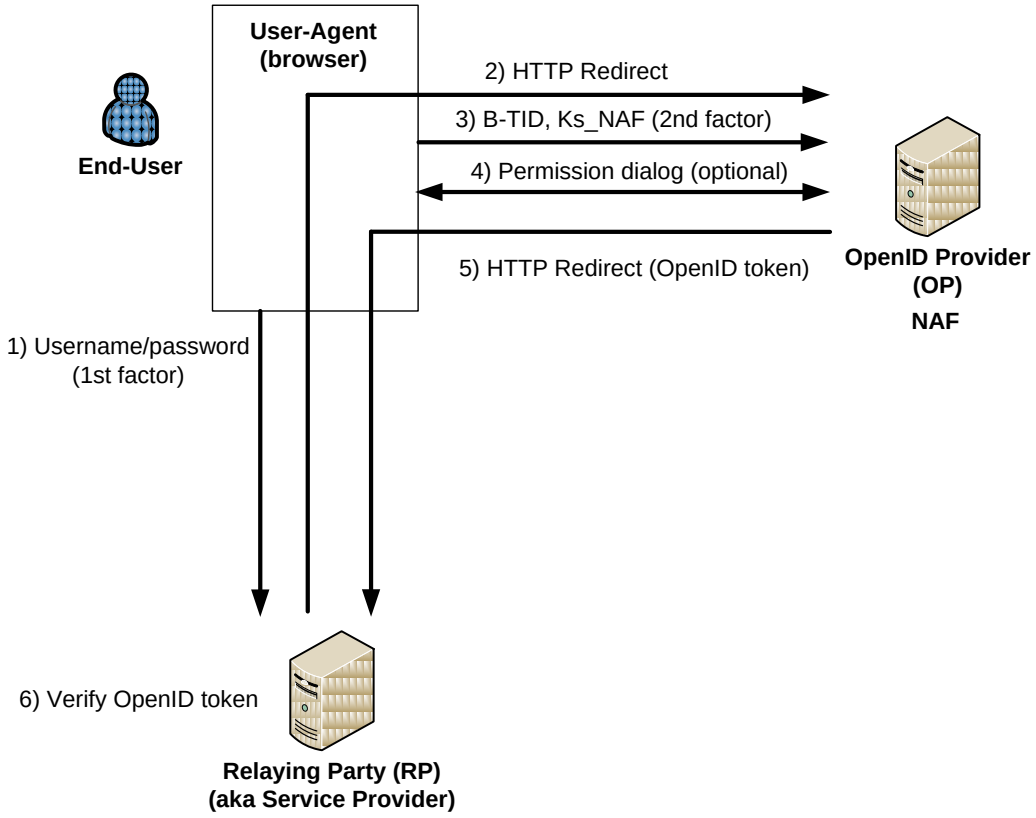

Figure 8.3.3.1-1: Two factor authentication based on OpenID – GBA interworking where OTT performs username/password authentication

1. The user initiates the login process by sending his username/password to the Relying Party via the User-Agent
2. The Relying Party verifies the username/password, and if successful, redirects the end user's User-Agent to the OP and thereby requests OpenID authentication
3. The OP initiates GBA authentication which triggers the User-Agent to start authentication using its GBA credentials with the OP
4. The OP verifies the GBA credentials and, optionally, presents a permission dialog asking the user whether OpenID data should be sent to the OP
5. If the user gives his approval in step 4, the OpenID token is sent to the RP via the User-Agent
6. The Relying Party verifies the OpenID token and if the verification is successful the user is considered logged in.

8.3.3.2 Solution based on OpenID-GBA interworking where MNO performs both GBA and username/password authentication

The solution presented here is based on OpenID – GBA interworking. Two factor authentication is achieved by adding an additional step before the GBA authentication where the OP requests username/password from the user.

Since OP receives the username/password and GBA credentials in the same TLS tunnel or HTTP session it is assured that they were both provided by the same entity, In other words the "binding" between the first and second factor of authentication is accomplished by the TLS tunnel or HTTP session.

Note that the method for distributing username/password pairs to end-users is considered out-of-scope.

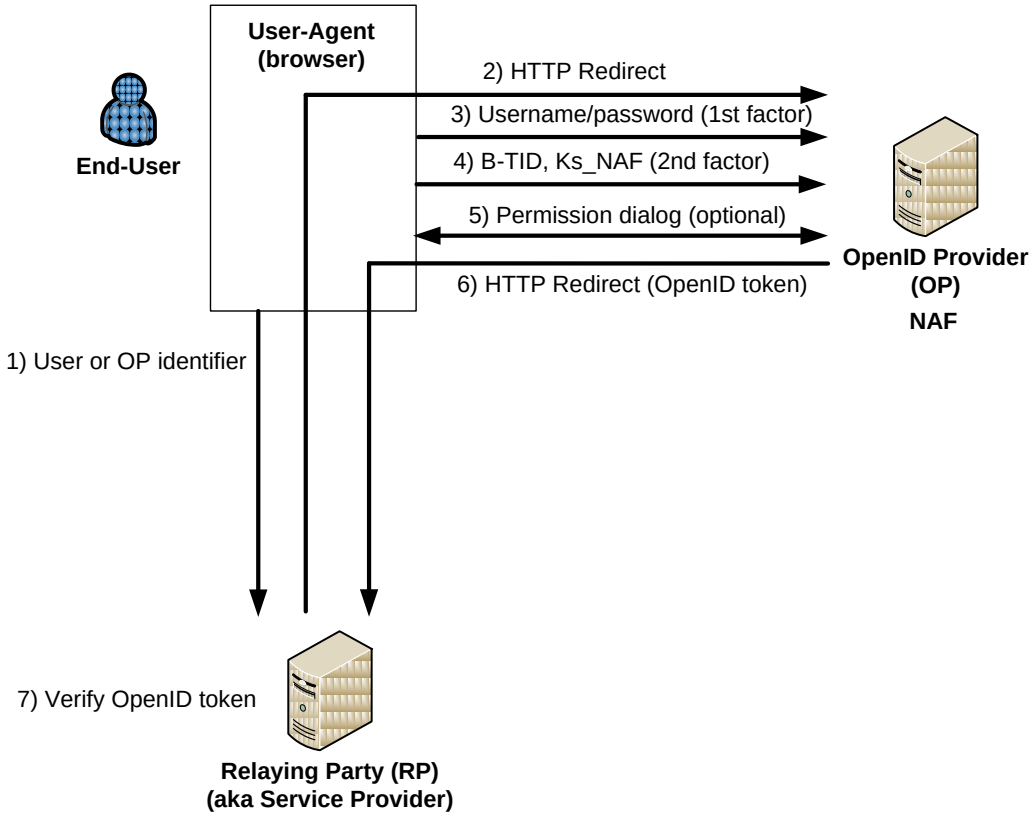

Figure 8.3.3.2-1: Two factor authentication based on OpenID – GBA interworking where OP also performs username/password authentication

1. The user initiates the login process by presenting an identifier of himself or the OP to the Relying Party via the User-Agent
2. The Relying Party redirects the end user's User-Agent to the OP and thereby requests OpenID authentication
3. The OP requests username/password which the end user supplies via the User-Agent
4. Provided the username/password pair is valid, the OP initiates GBA authentication which triggers the User-Agent to start authentication using its GBA credentials with the OP
5. The OP verifies the GBA credentials and, optionally, presents a permission dialog asking the user whether OpenID data should be sent to the RP
6. If the user gives his approval in step 4, the OpenID token is sent to the RP via the User-Agent. Optionally, the OP can indicate to the Relaying Party that two-factor authentication was used via the OpenID PAPE extension [13].
7. The Relying Party verifies the OpenID token and if the verification is successful the user is considered logged in.

8.3.4 Evaluation against findings in SA1 study

8.4 Using user consent for GBA and SSO

8.4.1 Rationale for solution

This solution is based on user giving her consent, or authorization, for the GAA server in terminal to derive NAF keys for a specific GAA client. The consent is achieved by a local user authentication (e.g. a PIN) between the user and the User Equipment. The intention of the local user authentication is to confirm the presence of the authorized user according to SA1 requirements in TS 22. 101 [11] and thereby avoid that GBA-based authentication would be used to access services in the background without the user noticing it, and ensure that only authorized persons are able to use GBA-based authentication.

The solution enables confirming that the authorized user is present and gives consent for using GBA keys for an application. Using a nonce approach ensures that the NAF keys are always fresh and not cached in the GAA client.

8.4.2 Solution description

8.4.2.1 General

The solution uses the concepts defined in TR 33.905 [12] "Recommendations for trusted open platforms", where the realization of GBA functionality in a trusted open terminal platform is divided into so called GAA server and GAA client. The GAA server in the terminal is the counterpart of the BSF, and the GAA client in the terminal is the counterpart of the NAF. This is assumed to be a typical division in a terminal implementing GBA. Typically the terminal internal interfaces or APIs are not standardized, and it is not the intention here either. The internals of a terminal are shown in order to explain the solution.

The flow is very similar to the regular GBA flow where the GAA client in the terminal contacts the NAF in order to access a service. The NAF then indicates to the GAA client to use GBA-based keys to secure the Ua application protocol, but in addition the NAF also requires that the presence of the authorized user needs to be confirmed (by sending $Nonce_{UI}$). "UI" stands for "User Involvement". When the GAA client requests NAF keys from the GAA server, the GAA client also consequently requests local user authentication.

The exact mechanism for local user authentication does not need to be specified. It can be for example a PIN code which the user has defined for the GAA server. It should be noted that it is not the same as the PIN to activate the USIM application.

8.4.2.2 GBA_ME-based solution

By local user authentication, the GAA server can locally confirm that the authorized user is present. For instance, the GAA server may present a dialog box to the user asking to authorize that application "Bank.com" can use GBA authentication.

If and only if the GAA server has locally authenticated the user, the GAA server derives new type of NAF keys which are bound to the ongoing transaction by taking the $Nonce_{UI}$ in the NAF key derivation. It should be noted that the result of the local user authentication (e.g. a PIN) is *not* taken into the NAF key derivation. Instead, the GAA server is a trusted element in the terminal which, in addition to performing bootstrapping and deriving NAF keys for applications, is trusted to perform local user authentication when the GAA client indicates that local user authentication is needed. If the GAA client does not indicate that local user authentication is needed, the GAA server derives the regular NAF keys. This approach avoids the burden and complexity of syncing the user authentication credentials, e.g. a PIN, with the network.

The GAA client uses the received NAF keys for authentication in the Ua application protocol. The NAF requests the NAF keys from the BSF and includes the $Nonce_{UI}$ in the Zn request and gets the same NAF keys as the GAA client did.

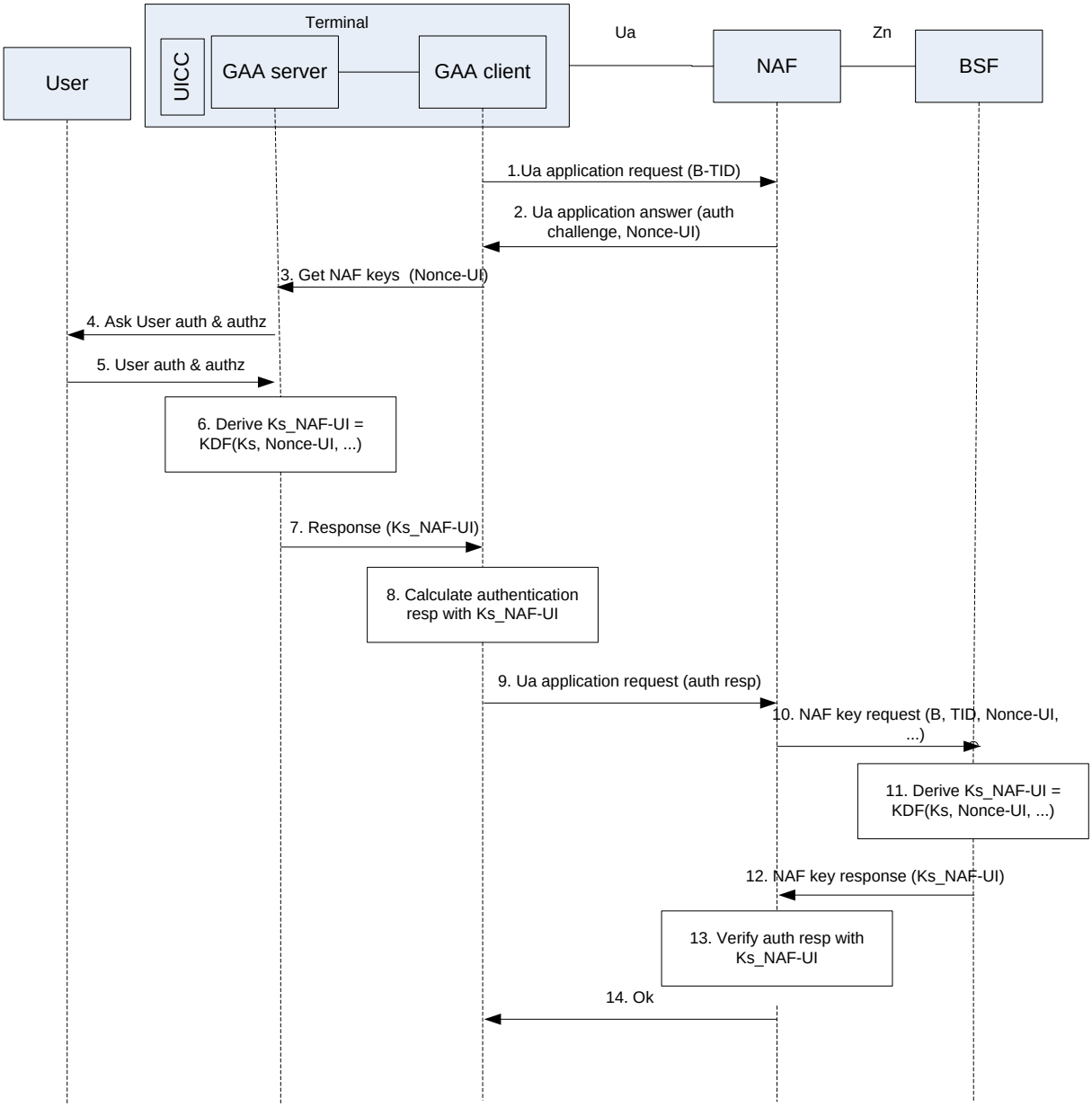

Figure 8.4.2.2.-1: Using User consent for GBA_ME

1. The GAA client in the terminal sends an Ua application request to the application server (i.e. NAF). The request includes the B-TID. In case of GBA – Open ID interworking the UE has been redirected by the RP to contact OP/NAF.

2. The NAF sends back an Ua application answer with an authentication challenge and $Nonce_{UI.}$ The $Nonce_{UI}$ could be sent for exmple in HTTP product token.

3. When the GAA client requests NAF keys from the GAA server in the terminal it includes the $Nonce_{UI}$ in the request.

4. When the GAA server in the terminal receives a requst for NAF keys with $Nonce_{UI,}$ the GAA server requests for user's authorization (e.g. a PIN) to derive the NAF keys for this GAA client.

5. The user provides authorization (e.g. PIN).

6. If the user authorization was given, e.g. the provided PIN is correct, the GAA server in the terminal derives NAF keys using $Nonce_{UI}$ as an input in the following way Ks_NAF-UI = KDF(Ks, $Nonce_{UI}$, …), where Ks_NAF-UI derivation takes the same input as Ks_NAF derivation, but added with the NonceUI (and with a different FC value). If needed, the GAA server runs bootstrapping before step 6.

7. The GAA server provides Ks_NAF-UI to the GAA client.
8. The GAA client uses the Ks_NAF-UI as the key to calculate the authentication response for the Ua application request.
9. The GAA client sends the Ua application request to the NAF.
10. The NAF requests NAF keys, and optionally USS, from the BSF over Zn. $Nonce_{UI}$ is included in the request.
11. When the BSF receives the Zn request with $Nonce_{UI}$, the BSF calculates the Ks_NAF-UI using $Nonce_{UI}$ as an input in the NAF key derivation similarly as in step 6.
12. The BSF sends Zn response with Ks_NAF-UI to the NAF.
13. The NAF uses the received Ks_NAF-UI to verify authentication response received from the GAA client in step 9.
14. The NAF sends an Ua response to the GAA as a result of a successful authentication. In case of GBA – Open ID interworking the UE is re-directed back to the the RP.

The flow shows a generic authentication handshake between the GAA client and the NAF over Ua relying on GBA_ME to illustrate how the mechanism works, and it should be noted that the derived NAF keys could be used to protect in principle any Ua application protocol.

8.4.2.3 GBA_U-based solution

Editor's Note: GBA_U-based solution is FFS.

8.Y <Solution Y>

8.Y.1 Rationale for solution

Editor's Note: The purpose of this clause is to justify why the solution should be considered in the present study, e.g. if it is an existing solution, or in case of a new solution there needs to be justification why a new solution is proposed.

8.Y.2 Solution description

8.Y.3 Evaluation against findings in SA1 study

9 Conclusions

Editor's Note: The purpose of this clause is to draw conclusions from the study, including possible recommendations for the way forward.

Annex <A>:
<Annex title>

Annexes are labelled A, B, C, etc. and are "informative"(3G TRs are informative documents by nature).

A.1 Heading levels in an annex

Annex <X>: Change history

Change history							
Date	TSG #	TSG Doc.	CR	Rev	Subject/Comment	Old	New
2011-01					Skeleton and scope agreed in SA3 #62		0.1.0
2011-05					S3-110507 agreed in SA3 #63.	0.1.0	0.2.0
2011-12					Added S3-111056 agreed in email discussion after SA3 #65.	0.2.0	0.3.0
2012-12					Updated with modified version of S3-121158 after email approval after SA3 #69.	0.3.0.	0.4.0
2013-01					Updated with modified version of S3-130229 at SA3 #70.	0.4.0	0.5.0
2013-05					Updated after SA3 #71 due to email approval of S3-130570	0.5.0	0.6.0
2013-10					Updated after SA3 #72 due to email approval of S3-130703, S3-130704, S3-130705, S3-130706, S3-130707, S3-130723, and S3-130891.	0.6.0	0.7.0