3GPP TS 33.224 V11.1.0 (2013-03)

Technical Specification

3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Generic Authentication Architecture (GAA); Generic Bootstrapping Architecture (GBA) Push Layer (Release 11)

Keywords
UMTS, LTE, Security, Architecture

3GPP

Postal address

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

Internet
http://www.3gpp.org

Contents

Foreword

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

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

Version x.y.z

where:

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

Introduction

3GPP defined the Generic Authentication Architecture (GAA). The adoption of GAA by other standardization bodies showed that some services can not make the assumption that the User Equipment (UE) has always the possibility to connect to the Bootstrapping Server Function (BSF). This specification introduces a generic push layer that makes use of the GBA Push Function as specified in TS 33.223 [2].

1 Scope

The present document specifies a generic push layer that makes use of the GBA Push Function as specified in TS 33.223 [3]. The GPL specification includes a message format, cipher suites and processing model. GPL assumes that keys and other SA parameters have been preinstalled in the Push-NAF and UE in the form of a NAF SA. GPL is a protection protocol that can be applied in a unidirectional fashion.

The rationale for GPL is that having each application specify its own security mechanisms would for obvious reasons lead to duplication of work, specifications and implementations. Using a generic secure push layer avoids these problems. A generic secure push layer may also relieve the applications using the service of having to be aware of inner working of the security layer. As an analogy, TS 33.222 [4] can be mentioned, which provides a generic security layer for HTTP based applications.

2 References

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

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

[1] 3GPP TS 33.220: "Generic Authentication Architecture (GAA); Generic bootstrapping architecture".

[2] 3GPP TR 21.905: "Vocabulary for 3GPP Specifications".

[3] 3GPP TS 33.223: "Generic Authentication Architecture (GAA); Generic bootstrapping architecture: Push Function"

[4] 3GPP TS 33.222: "Generic Authentication Architecture (GAA); Access to network application functions using Hypertext; Transfer Protocol over Transport Layer Security (HTTPS)".

[5] FIPS PUB 180-2 (2002): "Secure Hash Standard".

[6] IETF RFC 2104 (1997): "HMAC: Keyed-Hashing for Message Authentication".

[7] ISO/IEC 10118-3:2004: "Information Technology – Security techniques – Hash-functions – Part 3: Dedicated hash-functions".

[8] NIST Special Publication 800-38 A (2001): "Recommendation for Block Cipher Modes of Operation - Methods and Techniques "

[9] FIPS PUB 197 (2001): "Advanced Encryption Standard"

[10] OMA-WAP-TS-WSP-V1_0-20020920-C: "Wireless Session Protocol 1.0"

[11] 3GPP TS 31.111: "Universal Subscriber Identity Module (USIM) Application Toolkit (USAT)"

[12] ETSI TS 102 600: "UICC-Terminal interface; Characteristics of the USB interface

[13] ETSI TS 102 483: "UICC-Terminal interface; Internet Protocol connectivity between UICC and terminal"

3 Definitions, symbols and abbreviations

3.1 Definitions

For the purposes of the present document, the terms and definitions given in TR 21.905 [2], TS 33.220 [1] and the following apply.

SN_h The highest sequence number received in a GPL message with validated MAC. SN_h is used for replay protection.
SN_s A counter used to generate sequence numbers for outgoing messages.

3.2 Abbreviations

For the purposes of the present document, the abbreviations given in TR 21.905 [2] 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 [2].

GBA Generic Bootstrapping Architecture
GPI GBA Push Information
GPL Generic Push Layer
GPL_ME GPL hosted in the ME
GPL_U GPL hosted in the UICC
HSP High Speed Protocol
NAF Network Application Function
KDF Key Derivation Function
MAC Message Authentication Code
SA Security Association
SAID Security Association Identifier
SN Sequence Number

4 GPL requirements

4.1 Session concept

It is reasonable to expect that there will exist Push-NAF based services that rely on some form of per device session concept, and which would benefit from pushing more than one message based on the same security association. An example could be a virus signature update server. It is possible that the virus signatures are delivered in multiple pushed messages (for size limitation reasons of the underlying push transport mechanism), and it would then be inefficient to establish a new security association for each message.

This requires that GPL provides replay protection in addition to integrity protection (and possibly confidentiality protection). Figure 4.1-1 depicts the usage scenario, where three push messages are delivered from the Push-NAF to the UE using a single security association. Note that steps 1 and 2 in Figure 4.1-1 are out of scope for this specification. One way to achieve steps 1 and 2 is to use TS 33.223 [3].

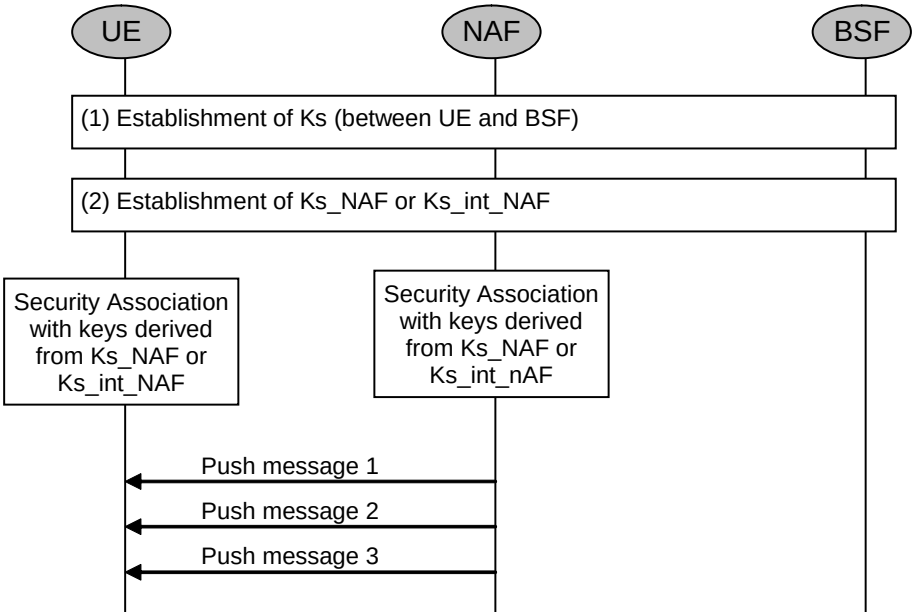

Figure 4.1-1: Example of a secure session

If GPL was to provide a complete session concept including reliability of delivered messages using timeouts/acknowledgments and re-transmissions, re-establishments of sessions, re-ordering of messages etc., GPL would be unnecessarily complex and the size of the GPL messages would be too large for many applications (e.g., when the underlying transport is SMS). Therefore GPL shall only provide sufficient session state to ensure that the security of multiple GPL messages is not compromised. GPL shall hence provide the security services confidentiality, integrity protection and replay protection for a GPL session.

If a more complex session concept is required by the application, where the session concept includes services other than security services, then, e.g., WSP [10] could be applied on top of GPL, but this is out of scope for this specification.

Even though it sometimes is sufficient with a secure downlink channel from the Push-NAF to the UE (for broadcast only UEs) a uplink channel may be present. An example of this is OMA's location based services, where a server requests location information from a terminal, which responds with its location information. This request/response exchange may be repeated every ten minutes. It is prudent to require that it shall be possible to secure also such an uplink channel. The security of the uplink channel can conveniently be based on the same NAF SA as the downlink channel.

To send a GPL message to the UICC, the Push-NAF selects a delivery channel that targets a UICC and that is supported by the GPL-capable ME (e.g. SMS class 2). The protocols that the GPL-capable ME shall support to receive the GPL_U messages depend on the type of interface between the ME and the UICC (ISO or HSP).

GBA Push as specified by TS 33.223 [3] is capable of establishing a Ks_int_NAF and a Ks_ext_NAF using one GPI. However, GPL is not designed to make full use of this. It is not possible to establish GPL SAs for both GPL_U and GPL_ME from a single GPI.

4.2 Requirements

The following requirements shall be posed for the generic secure push layer:

R1: It shall perform encapsulation of generic application layer messages from the Push-NAF to the UE.

R2: It shall allow sending multiple messages based on the same security association.

R3: Integrity protection and confidentiality protection shall be possible to provide for the messages. Integrity protection is mandatory to apply, while confidentiality protection is optional to apply.

R4: Detection of replayed messages within the same session shall be possible.

R5: If uplink messages are present in the application protocol, it shall be possible to apply the same level of protection to these messages, based on keys derived from the Ks_NAF or Ks_int_NAF.

R6: The Push-NAF shall select the target of the GPL message, UICC or ME, by choosing the type of delivery channel. To send a GPL message to the UICC, the Push-NAF shall select a delivery channel that targets a UICC and that is supported by the GPL-capable ME (e.g. SMS class 2).

R7: The protocols that the GPL-capable ME shall support to receive the GPL_U message depend on the type of interface between the ME and the UICC:

- In case of ISO interface between the ME and the UICC, the ME shall support "ENVELOPE SMS-PP data download" and "Bearer Independent Protocol in client mode" (class e in client mode) as specified in 3GPP TS 31.111 [11]
- In case of HSP interface between the ME and the UICC as specified in TS 102 600 [12], the ME shall support "ENVELOPE SMS-PP data download" over HSP and ETSI TS 102 483 [13]

NOTE: There is no need to specify new interface between the ME and the UICC.

To utilise GPL as described in this document the ME and/or UICC shall be equipped with a GPL protocol entity implementing the processing of the protocol. In addition, to be able to use GPL it is necessary to implement the GPI processing described in TS 33.223 [3].

The GPL protocol entity resides in the ME (called GPL_ME) or in the UICC (called GPL_U). When the GPL protocol entity to be used is in the ME, Ks_NAF shall be used as the shared master key between the ME and the Push-NAF. When the GPL protocol entity to be used resides in the UICC, Ks_int_NAF shall be used as the shared master key between the UICC and the Push-NAF.

The Push-NAF must have knowledge of ME's capabilities to support GPL_ME and/or the USIM/ISIMs capabilities to support GPL_U (depending on which GPL protocol entity is targeted). Otherwise the Push-NAF cannot know if it can send GPL messages to the UE or which type of GPL messages the UE understands. Therefore the GPL capabilities of the ME shall be indicated to the Push-NAF during GBA-Push UE registration procedure which is specified in Annex B in TS 33.223 [3]. The GPL_U capabilities of the USIM/ISIM shall be stored in the GUSS in the HSS.

5 GPL Processing

5.1 Processing model

In case of GPL_ME, the GPL protocol entity is conceptually located either between the transport mechanism (which could e.g. be SMS, IP, IP/UDP) and the application, or included in the application.

In case of GPL_U, the GPL_U protocol entity is located within the targeted USIM or ISIM.

When receiving a GPL protected message, the recipient transfers the message to the GPL protocol entity How the recipient knows that the message is a GPL message is up to each transport mechanism to define. It could be through, e.g., a special application ID that is tagged onto the message, in which case a GPL application ID needs to be defined.

In GPL_ME, the message is delivered to the transport mechanism again after GPL processing is complete. This time around the GPL application ID and GPL related data has been removed, and what remains is a regular application data message (which is routed to the intended application using the transport layers normal dispatching mechanism). The processing model for GPL_ME is depicted in Figure 5.1-1.

In case of GPL_U, the GPL protected message is delivered to the targeted USIM or ISIM that will process the GPL message. The application data remain in the USIM or ISIM for use by the application defined therin.

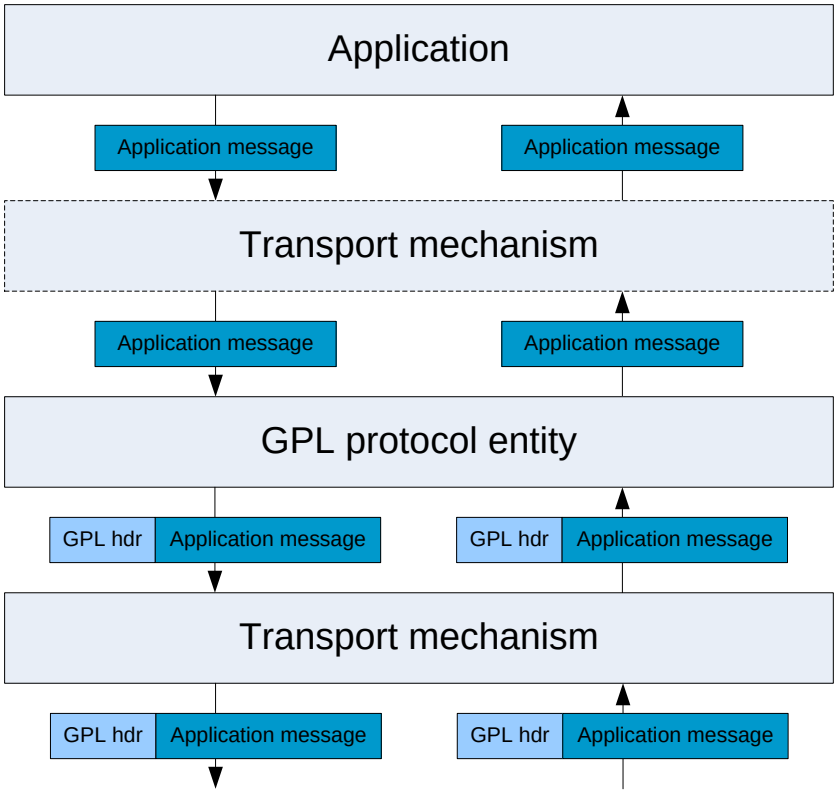

Figure 5.1-1: The processing model for GPL_ME agnostic applications. Inbound processing to the right and outbound processing to the left.

The case when the GPL_ME protocol entity is logically located between the transport mechanism and the application is illustrated in Figure 5.1-1. In this case the application does not need to be aware of GPL, and any legacy application can use GPL without modifications. The other case is that the application is aware of GPL. The reason for this may be that the application needs to inform the GPL protocol entity about which security association to use (i.e., the application calls the GPL protocol entity directly, and the GPL protocol entity may either pass the GPL encapsulated message to the transport mechanism, or return it to the application).

5.2 Session start

A GPL session is considered started in a GPL protocol entity when the corresponding GPL Security Association (GPL-SA) is initialised, see clause 5.9 For the Push-NAF, this means that the session shall be considered initiated as soon as it has received the GPI from the BSF and configured the NAF SA (see [3]) and corresponding GPL-SA. For the UE, the session shall be considered started when it has received the GPI and configured the GPL-SA.

In addition to the GPI, the GPL protocol entity needs to get GPL policy information for the session, e.g., which encryption and integrity algorithms to use etc. The policy information may be decided by the application itself or by some other management entity.

The Push-NAF shall choose the policy to use for the downlink messages and it shall be included in the GPL message. The Push-NAF shall choose a cipher suite for downlink GPL messages, and the UE shall (in case an uplink is present) choose the cipher suite for the uplink. It is recommended that the UE chooses the same cipher suite for the uplink as the Push-NAF chose for the downlink.

5.3 Session termination

Sessions are not explicitly terminated, i.e., there is no specific GPL message for closing a session. The NAF-key life time is kept by each peer and when that time is reached, the GPL-session is over and the NAF SA, corresponding downlink GPL-SA and corresponding uplink GPL-SA (if it exists) shall be deleted.

5.4 GPL security association

A GPL Security Association (GPL-SA) is the data required for processing of either inbound or outbound GPL messages. That is, in case there is a bi-directional communication link, each peer shall keep two GPL-SAs, one for the inbound traffic, and one for the outbound traffic.

A GPL-SA shall be derived from a NAF SA with which it shares life time and ID (see TS 33.223 [3] for further information on NAF SAs). A GPL-SA shall be deleted if the corresponding NAF SA is deleted and vice versa.

A GPL-SA shall be associated with an identifier, the SAID, to allow indexing of GPL-SA's within a Push-NAF and a UE respectively. An uplink/downlink GPL-SA can always be identified via the SAID, NAF_ID of the Push-NAF and the direction.

The GPL-SA shall contain at least the following items:

SAID: An identity which helps identify GPL-SAs within a Push-NAF or a UE. The uplink/downlink SAID is identical to the corresponding uplink/downlink NAF SA identity.

Master key: A 256-bit key used as master key for the key derivations of integrity and encryption keys. The master key is derived from the corresponding NAF-key.

SN_h: The highest sequence number received in a GPL message with validated MAC. SN_h is used for replay protection. This state-variable is only used in an inbound GPL-SA.

SN_s: A counter used to generate sequence numbers for outgoing messages. The counter shall be increased after each message is output. This state-variable is only used in an outbound GPL-SA.

Cipher suite: The cipher suite used for protection of messages. A cipher suite shall consist of one integrity protection algorithm, one encryption algorithm, and one key derivation algorithm.

GPL-SA life time: This is the expiry time of the GPL-SA. The GPL-SA life time and its format shall be the same as the life time of the corresponding NAF-key.

5.5 Combined delivery

It is possible to send the GPI message either within the GPL message or separately. When the GPI message is sent within the GPL message this is called combined delivery.

NOTE 1: GBA-Push TS 33.223 [3] allows that GPI messages are retransmitted several times including cases when it is sent every time a payload is pushed to the UE. To handle retransmissions efficiently TS 33.223 [3] defines a mechanism how the UE is able to only invoke the USIM/ISIM after checking that the GPI does not correspond to an already existing NAF SA.

5.6 Message format

5.6.1 Data unit transfer format

A GPL message is laid out as shown in Figure 5.6.1-1. The GPL message encapsulates and protects an application message in the GPL payload.

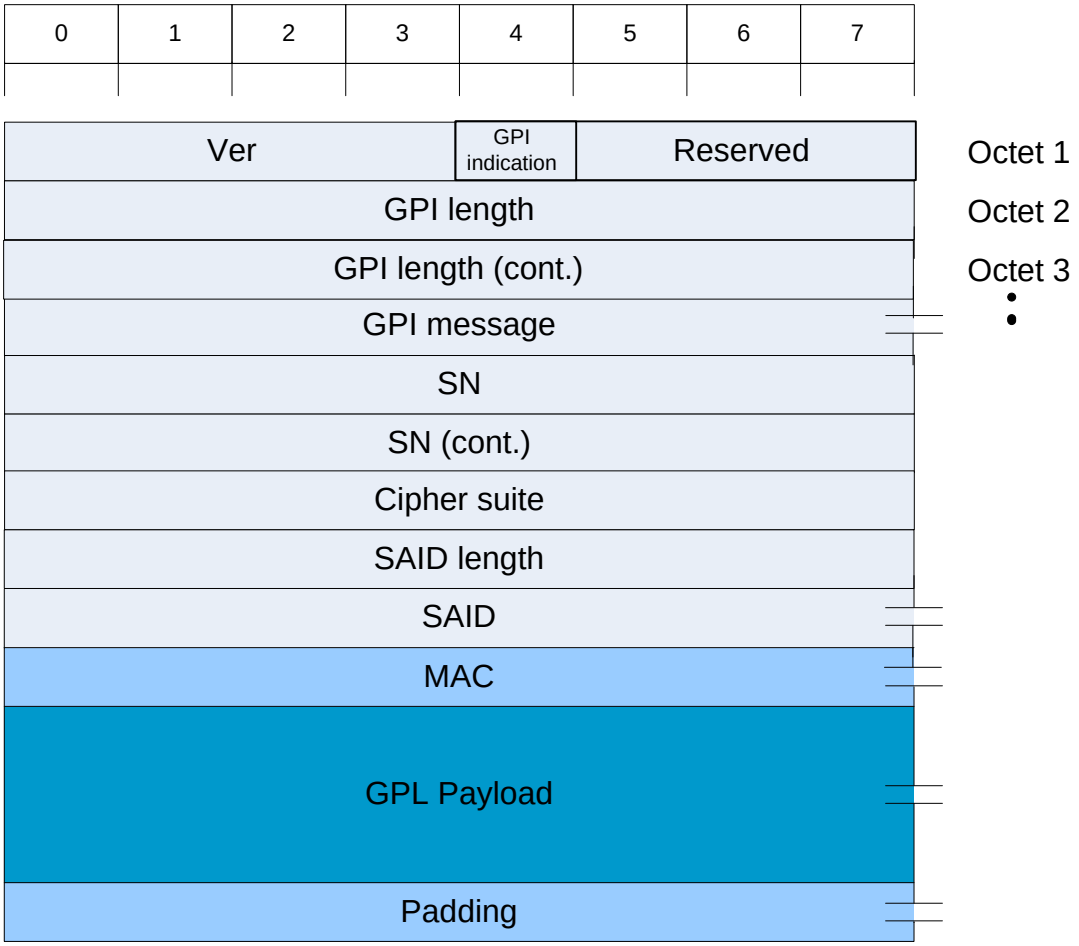

Figure 5.6.1-1: GPL message format

Each field is encoded in network byte order (i.e., big endian) with the most significant bit being bit number zero. The fields of the message are the following.

Ver (4 bits): The version of the GPL protocol encoded as an integer. The version of any message conforming to this specification shall use the value 1, i.e., the first nibble of the message shall be 0x1.

GPI Indication (1 bit): Indicates if combined delivery is used, i.e. if a GPI message is present in the GPL message or not. When GPI Indication equals 0, fields for GPI length and GPI message are not present. When GPI Indication equals 1, fields for GPI length and GPI message are present.

Reserved (3 bits): These bits are reserved for future versions of this specification. Implementations conforming to this specification shall set these bits to zero before transmitting a message, and the receiver of the message shall ignore these bits.

GPI length (16 bits): The length of the GPI message in number of octets. This field is present only when GPI Indication is set to 1.

GPI message (variable length): The GPI message. This field is present only when GPI Indication is set to 1.

SN (16 bits): The sequence number used for synchronizing the encryption and providing replay-protection.

Cipher suite (8 bits): The cipher suite used for protection of the message. The cipher suite consists of one integrity protection algorithm, one encryption algorithm, and one key derivation algorithm.

SAID length (8 bits): The length of the SAID in number of octets.

SAID (variable length): The identity of the GPL security association used for protection of the message.

MAC (variable length): The message authentication code providing integrity protection of the GPL message. The length of this field is determined by the output size of the integrity protection algorithm used, but shall be a multiple of 8 bits.

GPL Payload (variable length): The actual application message that is protected. The length of the message shall be a multiple of 8 bits, and must be padded by the application unless this condition is met. Any such padding is up to the application and is out of scope for this specification. This field is encrypted.

Padding (variable length): Padding as required by the encryption transform. Exactly how the padding is generated, verified and removed is defined by each encryption transform. In case the encryption transform does not require padding, this field is not present. This field is encrypted.

The fields Ver, GPI length, GPI message, Cipher suite, SAID length and SAID are fixed during the GPL-SA life time and shall not change between messages. The fields GPI length and GPI message may however be present in some messages and absent in some messages depending on the GPI Indication field.

5.7 Inbound processing

Before processing of any inbound GPL message, the GPL protocol entity initiates the GPL-SA as described in clause 5.9. In case of combined delivery, this step shall be performed after GPI message processing (step 2) below.

When a GPL message arrives at the receiver's GPL protocol entity, the following processing steps shall be taken:

1. Verify that the Ver field is equal to 1. If this is not the case the GPL message shall be discarded and the processing shall stop.

2. In case of GPI indication does not indicate combined delivery goto step 4, Else process the GPI message as follows:

   In case of GPL message sent to the ME, the GPL_ME processes the GPI message as defined in TS 33.223 [3].

   In case of GPL message sent to the UICC, the GPL_U protocol entity checks if the GPI corresponds to an already existing NAF SA. If not, the GPL_U protocol entity derives Ks_ext/int_NAF from GPI information as described in TS 33.223 [3], creates a NAF SA according to TS 33.223 [3] and stores the NAF SA associated to Ks_int_NAF. The key Ks_ext_NAF key is not sent outside the UICC.

NOTE: GBA-Push TS 33.223 [3] allows that a GPI message is retransmitted several times including cases when it is sent every time a payload is pushed to the UE. To handle retransmissions efficiently TS 33.223 [3] defines a mechanism how the UE is able to only invoke the USIM/ISIM after checking that the GPI does not correspond to an already existing NAF SA.

3. Retrieve the GPL-SA which corresponds to the SAID in the GPL header. If no GPL-SA matching the SAID is found, the message shall be discarded and the processing shall stop.

4. If the Cipher suite variable in the GPL-SA has not been set, verify that the cipher suite indicated in the GPL message is supported and set the Cipher suite variable equal to the Cipher suite field in the GPL message. If the variable has been set, verify that the variable and field are equal. If the cipher suite is not supported or the field does not equal the variable, then the GPL message shall be discarded and the processing shall stop.

5. Verify that the sequence number carried in the SN field has not yet been received. One way of accomplishing this is to verify that the sequence number in the SN field is larger than the currently highest received sequence number SN_h. If this is not the case, the message shall be discarded and the processing shall stop. When SN_h is equal to 0xffff, all messages with the given SAID shall be discarded and the processing shall stop. It is not mandatory to implement this particular replay mechanism (which is not robust against message reordering), but the receiver's GPL protocol entity shall verify that the sequence number in the SN field has not been received before in a valid message.

6. Compute the MAC using the integrity algorithm indicated by the cipher suite. The MAC is computed over the entire GPL-message, and during the computation, the MAC field shall be treated as containing all zeros. After the MAC is computed, it shall be compared to the MAC carried in the MAC field. If the two MACs differ, the message shall be discarded and the processing shall stop.

7. Update the replay protection state. In case the mechanism described in step 5 is used, the state-variable SN_h is set equal to the SN field.

8. Decrypt the application message using the decryption algorithm defined by the GPL-SA. The decryption transform is applied to GPL payload and padding fields in the GPL message.

9. In case of GPL_ME, return the application message of the GPL message (i.e., what remains after removing the GPL header and possible padding field) to the transport mechanism the message was received from.

   In case of GPL_U, the payload of the GPL message remains within the application.

If the processing is stopped by the GPL protocol entity before the full processing is complete an error indication may be returned from the GPL protocol entity.

5.8 Outbound processing

Before processing of any outbound GPL message, the GPL protocol entity initiates the GPL-SA as described in clause 5.9. When an application message arrives at the sender's GPL protocol entity, the following processing steps shall be taken:

1. If the state-variable SN_s is equal to 0xffff, the processing shall stop and an error indication shall be returned from the GPL protocol entity.

2. In case of combined delivery, include GPI in the GPI message field and set the GPI Indication and GPI Length fields accordingly.

3. Retrieve the GPL-SA which corresponds to the SAID as indicated by the caller of the GPL protocol entity. If no GPL-SA is found, the processing shall stop. In case of combined delivery, the GPL-SA needs to be the one derived from the NAF SA of the corresponding GPI.

4. Set the Ver field to 1. Set the Cipher suite, SAID and SAID length fields according to the GPL-SA. Set the SN field to the value of the state-variable SN_s.

5. Encrypt the application message using the encryption algorithm defined by the GPL-SA. If needed by the encryption algorithm, the message shall be padded before encryption.

6. Set the MAC field of the GPL header to zero and compute a MAC over the entire GPL message using the integrity protection algorithm defined by the GPL-SA. Copy the resulting MAC to the MAC field of the GPL header.

7. Increase the state-variable SN_s by one.

8. Deliver the GPL protected message to the next step in the processing chain.

If the processing is stopped by the GPL protocol entity before the full processing is complete an error indication may be returned from the GPL protocol entity.

5.9 Initialization of GPL-SA

5.9.1 General

The GPL-SA is initiated from a corresponding NAF SA. A NAF SA (used for GPL) shall be associated with one downlink GPL-SA and may be associated with an uplink GPL-SA. It is only allowed to initialize one GPL-SA per direction from a NAF SA. The reason for this is that the same key otherwise could be reused with the same sequence number. The NAF SA is defined in TS 33.223 [3]. The fields of the GPL-SA copied from the NAF SA keep their definitions.

5.9.2 Initialization of downlink GPL-SA from a NAF SA

The Push-NAF shall initialize the downlink GPL-SA from the corresponding NAF SA before sending the first GPL message to the UE. The Push-NAF shall:

- Set the GPL-SA SAID equal to the NAF SA's DL_SA_Id value, which is defined in section 5.2.2 in TS 33.223[3].

- Set the GPL-SA master key equal to Ks_NAF or Ks_int_NAF according to the NAF SA.
- Set the GPL-SA SN_s equal to 1.
- Set the life-time of the GPL-SA equal to the life-time of the NAF SA.
- Set the cipher suite and key indication ID according to the application's policy.

The UE shall initialize the downlink GPL-SA from the corresponding NAF SA when the NAF SA has been established (e.g., after processing a GPI). The UE shall:

- Set the GPL-SA SA ID equal to the NAF SA's DL_SA_Id value, which is defined in section 5.2.2 in TS 33.223[3].
- Set the GPL-SA master key equal to Ks_NAF or Ks_int_NAF according to the NAF SA.
- Set the GPL-SA SN_h equal to 0.
- Set the life-time of the GPL-SA equal to the life-time of the NAF SA.

5.9.3 Initialization of uplink GPL-SA from a NAF SA

If the application requires an uplink GPL-SA, the Push-NAF shall initialize the uplink GPL-SA from the corresponding NAF SA before processing the first uplink GPL message from the UE. The Push-NAF shall:

- Set the GPL-SA SA ID equal to the NAF SA's UL_SA_Id value, which is defined in section 5.2.2 in TS 33.223[3].
- Set the GPL-SA master key equal to Ks_NAF or Ks_int_NAF according to the NAF SA.
- Set the GPL-SA SN_h equal to 0.
- Set the life-time of the GPL-SA equal to the life-time of the NAF SA.

The UE shall initialize the uplink GPL-SA from the corresponding NAF SA after the NAF SA has been established (e.g., after processing a GPI) and before sending the first uplink GPL message to the Push-NAF. The UE shall:

- Set the GPL-SA SA ID equal to the NAF SA's UL_SA_Id value, which is defined in section 5.2.2 in TS 33.223[3].
- Set the GPL-SA master key equal to Ks_NAF or Ks_int_NAF according to the NAF SA.
- Set the GPL-SA SN_s equal to 1.
- Set the life-time of the GPL-SA equal to the life-time of the NAF SA.
- Set the cipher suite and key indication ID according to the application's policy.

5.10 Cipher suites

The following cipher suites are defined for use with GPL:

Cipher suite 1:

ID: 0x01

Encryption algorithm: CTR-AES128 as specified in NIST Special Publication 800-38 A (2001) [8] and FIPS PUB 197 (2001) [9]. The standard incrementing function is used with m=16, according to appendix B in NIST Special Publication 800-38 A (2001) [8], i.e. the 16 least significant bits in T behave like a counter while the 112 most significant bits are static. The 128-bit initial counter block T_1 shall be the 96 least significant bits of the RAND (from NAF SA) concatenated with the 16-bit sequence number SN and right padded with 16 zeroe bits:

$$T_1 = \text{RAND} \,||\, \text{SN} \,||\, 0\text{x}0000$$

Integrity protection algorithm: HMAC-SHA256-32 as specified in FIPS PUB 180-2 (2002) [5], IETF RFC 2104 (1997) [6] and ISO/IEC 10118-3:2004 [7]. The MAC field is hence 32 bits long.

Key derivation function: As specified in TS 33.220 [1] Annex B. The input to the KDF is the 256-bit master key and the string S defined by:

- FC = 0x40
- P0 = key-purpose string
- L0 = length of the key-purpose string as a 16-bit integer.
- P1 = direction indicator
- L1 = length of direction indicator (i.e., 0x00 0x01)
- P2 = cipher suite ID
- L2 = length of cipher suite ID (i.e., 0x00 0x01)

The FC number space is used controlled by TS 33.220 [1], FC values allocated for this specification are in range of 0x40 – 0x48.

The key-purpose string shall be "gba-push-enc" for encryption keys and "gba-push-int" for integrity keys. The 128 least significant bits of the KDF output are used as key bits.

In case of bi-directional usage of GPL there is a need for two pairs of GPL-SAs, one for each direction. The direction indicator shall be set accordingly for the pair. The direction indicator shall be 0x00 for the GPL-SA protecting messages from the Push-NAF to the UE and 0x01 for messages sent from the UE to the Push-NAF (if such an SA is required by the application).

Cipher suite 2:

ID: 0x02

Cipher suite 2 shall be exactly as cipher suite 1 with the only difference that the integrity protection algorithm shall be defined as:

Integrity protection algorithm: HMAC-SHA256-64 as specified in FIPS PUB 180-2 (2002) [5], IETF RFC 2104 (1997) [6] and ISO/IEC 10118-3:2004 [7]. The MAC field is hence 64 bits long.

Cipher suite 3:

ID: 0x03

Cipher suite 3 shall be exactly as cipher suite 1 with the only difference that the integrity protection algorithm and the encryption algorithm shall be defined as:

Encryption algorithm: NULL encryption. This means that no encryption is applied. Using this ciphersuite provides no confidentiality protection.

Integrity protection algorithm: HMAC-SHA256-32 as specified in FIPS PUB 180-2 (2002) [5], IETF RFC 2104 (1997) [6] and ISO/IEC 10118-3:2004 [7]. The MAC field is hence 32 bits long.

Cipher suite 4:

ID: 0x04

Cipher suite 4 shall be exactly as cipher suite 1 with the only difference that the integrity protection algorithm and the encryption algorithm shall be defined as:

Encryption algorithm: NULL encryption This means that no encryption is applied. Using this ciphersuite provides no confidentiality protection

Integrity protection algorithm: HMAC-SHA256-128 as specified in FIPS PUB 180-2 (2002) [5], IETF RFC 2104 (1997) [6] and ISO/IEC 10118-3:2004 [7]. The MAC field is hence 128 bits long.

Annex A (informative): Use cases

A.0 General

This annex describes use cases for the combination of GBA Push as specified by TS 33.223 [3] and GPL as specified by this document. Some of the information is hence not applicable to this specification, but rather describes features of GBA Push as it is used to establish security associations for GPL.

A.1 Generic Push Layer - use case for terminals without a return channel

This clause describes a use case, how an application could make use of GBA Push and GPL.

The goal of the application is to be able to securely push a message from an application server (implemented in a Push-NAF) to a UE. For example, the Push-NAF pushes a message to a UE, including the latest virus signatures. This is a case which can be of interest for terminals without a return channel, e.g., pure broadcast terminals.

This functionality can be separated into two distinct phases: establishment of the security association between the Push-NAF and the UE, and the protection of the message. The security association contains keys derived from the Ks(_ext/int)_NAF. The two phases are depicted in Figure A-1, where the first phase includes steps (1) and (2), and the second phase includes step (3). The security associations are established between the Push-NAF and the UE as a result of phase 1 using GBA Push as specified in TS 33.223[3]. The second phase can use GPL as specified in this document.

The establishment of the security association boils down to establishment of the Ks, followed by establishment of the Ks(_ext/int)_NAF.

When it comes to the protection of the actual message that is to be pushed there are two options, either the push is a one-time occurrence, or the concept of a session can be introduced. A session would here mean that a secure one-way communication channel is established between the Push-NAF and the UE. The first phase is out of scope for this specification, which only deals with the second phase.

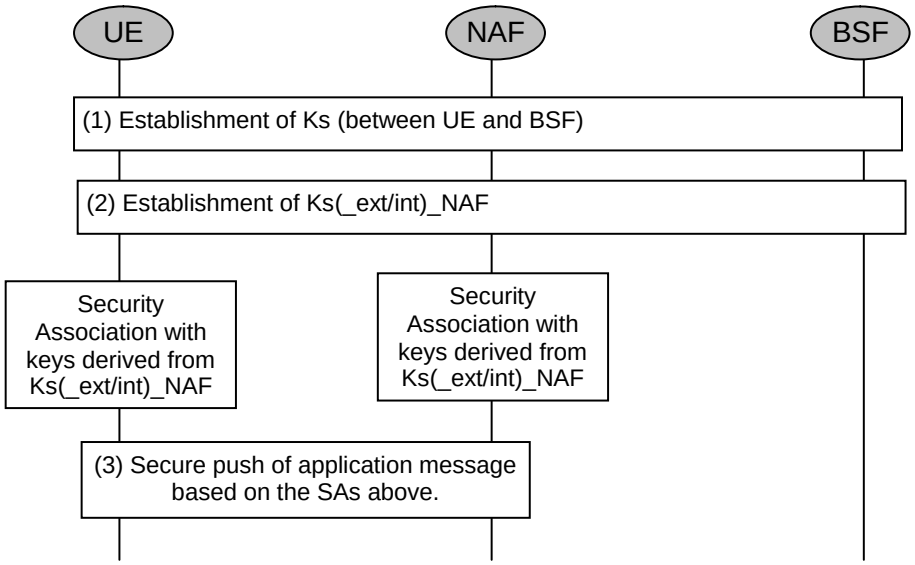

Figure A.1-1: The two phases involved in securely pushing a message to a UE from an application server

A.2 Specific use cases

A.2.1 Network initiated NAF-key refesh and distribution of keys

This use case is related to key distribution to terminals that receive broadcasted content and need a key be able to render the content and to key refresh for an existing security association. Keys could be pushed to the terminal at regular intervals to avoid having long-lived keys in the system. As the push message could be sent out over the broadcast system a back channel cannot be assumed to be available in all cases.

Services that have not their own inbuilt key refresh mechanism (e.g. like MBMS) may want to push a new security association to a user. The reasons for that might be continuous user service experience (e.g. keys expire during some critical time period), to spread load evenly on the network side or there is some maintenance in the network at the time of expiry (e.g. on the Push-NAF to BSF connection which causes reduced bandwidth). The maximum Ks lifetime is known by the BSF and the Push-NAF, if the Push-NAF desires a fresh key (because it expires or for Push-NAF policy reasons) it can proactively push a new security association to the UE.

Characteristics:

- Some UE's require the security association to be established at a predefined time before the old key expires
- There exists already a Ua security association, for the case where the keys are refreshed.
- The service consumption may occur much later then the security provisioning.
- The old NAF-keys might be NAF-keys generated in a GBA run according to TS 33.220 or in a GBA Push run.
- GBA_ME must be supported and GBA_U support might be needed, but not for OMA BCAST and MBMS, since those have their own inbuild refresh solution. Nevertheless, GBA Push might be useful for MBMS.

A.2.2 Distribution of tokens

This use case is very similar to the key distribution case except for one important difference and that is that the token should be presented to the service provider over an IP connection. Thus, it can be assumed that a back-channel exists

and that this back channel could be used to report the success of a GBA Push. On the other hand, the usefulness of the solution would be diminished if delivery of the tokens only was allowed to take place to terminals that are on-line, which would be required to get a timely success report and exclude use of deferred delivery channels.

A.2.3 MBMS GBA_U use case

GBA Push could be useful for MBMS.

GBA was created to secure MBMS and to allow UICC-based solution by means of GBA_U. In case of use of GBA Push for MBMS, it should be possible to send GBA Push messages protected by means of keys stored on the UICC, i.e. Ks_int_NAF-related keys. GBA Push solution shall support GBA_U.

A.2.4 OMA related use cases

OMA BCAST provided use-cases for GBA Push. One of this use-case is OMA BCAST smart card profile, which requires GBA Push solution with endpoint in the UICC. Consequently, GBA Push solution shall support GBA_U. In particular, OMA provided the following information:

"The smart card based service protection profile uses MBMS GBA mechanisms for registration and long term key delivery. It would be advantageous to also for this profile have network initiated registration and delivery of long term keys. A secure GBA Push mechanism would enable such a solution. "

Others OMA use-cases could rely on GBA_U-based GBA Push solution. E.g. OMA SEC group identified GBA Push-based key management as a good enhancement for future device management and client provisioning releases. This GBA Push solution could rely on Ks_ext_NAF/Ks_int_NAF keys to reinforce the security. But, for those use-cases, there is no OMA requirement mandating that GBA Push solution shall address the case of endpoint in the UICC.

A.2.5 Network initiated services

There are quite a number of services that are initiated from the network, which require that the terminal connects to a server in the network. Examples of OMA defined enablers having this modus operandi are e.g. Device Management (DM), Download DRM (DLDRM), DRM, and Secure User Plane Location (SUPL). In all these cases it is assumed that the triggering push message can be sent over SMS.

Having an efficient secure push system would allow greater flexibility as trusted parameters and keys could be sent in the trigger and provisioned parameters like server and sender white-lists could be avoided. Furthermore, a secure push would also allow protection against replay and DoS attacks.

As the services require that the terminal connects to the network there is a back-channel in these use cases. When the terminal connects to the server, the initiating Push-NAF will implicitly receive a confirmation that a GBA Push has succeeded. The success could then be reported by the server to the BSF via the Push-NAF.

A special case of a network initiated service is the case, where the operator may want to update securely information on the terminal e.g. device management or client provisioning. The UE has not contacted the operator before and hence can not be securely triggered to bootstrap (i.e. usage of SMS or WAP Push). The device management information should be pushed in a secure manner to the UE and the pushing may occur at a fixed point of time or during a pre-defined period. It must be ensured that the originator of the management message is authorized and that only the correct terminal can utilize the data.

Characteristics:

- The usage of the transferred information (called management message) to the UE may occur directly after the provisioning.
- Source of the management message must be identifiable i.e. the Push-NAF
- Only authorized recipients of the management message should be able to utilize it.
- Security association establishment (Upa) and delivery of the management message (Ua) might not occur together. Operator may want to send several messages secured by the same SA.

- Management information is targeted for the terminal, hence Ks_NAF support is sufficient.

A.2.6 BSF and HSS load balancing for broadcast

GBA is used for MBMS and OMA BCAST. The main advantage of broadcast is that many devices can be served with content at the same time. Typical broadcast scenarios include soccer games, olympics, eurovision contest and other events of general interest. If a majority of the UEs make the GBA bootstrapping run just before the event starts, then the BSF server and the HSS have to deal with a large load. Therefore it seems desirable, that the network can trigger the registration and delivery of the long term keys and many UEs can be provisioned with GBA credentials (NAF-keys) at times, where the BSF is having a low load e.g. the night before the event. The receiving terminal also supports GBA and in case, that it receives a broadcast message, for which no security association is available, then it start a GBA run according to TS 33.220.

Characteristics:

- Many UE's that require a security association at a specific time for service consumption.
- If GBA Push and GBA sessions according to TS 33.220 are managed together, then BSF can re-use Ks that was created for GBA Push for the GBA run according to TS 33.220. This also may work the other way around, that the Ks that has been established according to TS 33.220 could be used for GBA Push. If there is no uplink channel then this re-usage can not take place.
- It may be possible that there in no uplink channel available or it is undesired (due to network load) that many UE's make an uplink access at almost the same time (in the same area).
- The service consumption may occur much later then the security provisioning.
- Terminal should also support GBA according to TS 33.220 to allow service usage for the case that the SA via GBA Push did not arrive.
- GBA according to TS 33.220 is used (if supported) only for the case that no SA was received.
- The service that needs load balancing may require GBA (TS 33.220) or GBA_U support from the UE.

A.2.7 Download of vouchers / tickets

One use that has been discussed is to distribute vouchers/tickets for different types of public events or collection of physical goods by pushing a corresponding vouchers/ticket to the customers' mobile phone. The distribution could be scheduled to take place close in time to when the event starts to minimize the risk that the voucher/ticket is erased, lost in some other way or duplicated or it could be distributed well in advance to distribute the load on the network.

Vouchers/tickets need to be securely delivered to the legitimate receiver as they usually are not personalized.

If the distribution mechanism is reliable there is no need to report the reception of the voucher back to the issuer. Even with unreliable delivery channels reporting back is probably not needed as the user would note that the ticket has not been delivered and the user could then contact the issuer and ask for a retransmission.

The consumption of the voucher/ticket can not be used as a reliable means for reporting the success of the download. There are two reasons for this. The first one is that the vouchers may be consumed via off-line devices. The second reason is that the consumption may take place a long time after delivery.

Note also that this type of use case should work with deferred delivery of the message containing the voucher/ticket. If the receiver has turned off his phone, he should receive the message a soon as he turns the phone on again.

A.2.8 Distribution of news / information / commands

Distribution of news/information/commands, like stock prices or work orders, to employees need protection, especially integrity protection and source authentication, but in many cases also confidentiality protection. Such information should preferably be distributed over a system supporting deferred delivery to relieve the service provider of having to keep track of the user status and to make the information available as soon as the user switches on his terminal.

A.2.9 Set-top box use-case

It also exists the use-case of set-top box equipped with UICC reader and without return channel to the network. This use case implies that the GBA Push messages would be protected by means of the UICC.

A.2.10 Summary

The use cases and their characteristics above give rise to the following requirements:

(1) There are use cases, where the Ua protocol may terminate in the UICC and use cases, where it terminates in ME.

(2) GBA Push should support both GBA_ME and GBA_U.

(3) Separate delivery of Ua and Upa messages should be supported.

(4) The Upa bootstrapping should be able to support unidirectional message delivery (e.g. Upa over broadcast).

Annex B (informative): Change history

Change history							
Date	**TSG #**	**TSG Doc.**	**CR**	**Rev**	**Subject/Comment**	**Old**	**New**
Mar 2009	SA-43	SP-090135	--	--	Presentation to SA for information	---	1.0.0
Sep 2009	SA-45	SP-090528	--	--	Presentation to SA for approval	2.0.0	9.0.0
Dec 2009	SA-46	SP-090820	001	2	Description of GPL_U solution	9.0.0	9.1.0
Mar 2010	SA-47	SP-100098	002	1	General corrections	9.1.0	9.2.0
2011-03	-	-	-	-	Update to Rel-10 version (MCC)	9.2.0	10.0.0
2012-09	SA-57	SP-120605	003	1	Addition of missing cipher suites with NULL encryption for GPL	10.0.0	11.0.0
2013-03	SA-59	SP-130039	005	1	Update S3-130020 corrections to TS 33 224	11.0.0	11.1.0