**3GPP TSG-SA3 Meeting #100e *S3-202046***

**e-meeting, 17 - 28 August 2020**

|  |
| --- |
| *CR-Form-v12.0* |
| **CHANGE REQUEST** |
|  |
|  | **33.501** | **CR** | **0901** | **rev** | **1** | **Current version:** | **16.3.0** |  |
|  |
| *For* [***HE******LP***](http://www.3gpp.org/3G_Specs/CRs.htm#_blank)*on using this form: comprehensive instructions can be found at* [*http://www.3gpp.org/Change-Requests*](http://www.3gpp.org/Change-Requests)*.* |
|  |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ***Proposed change affects:*** | UICC apps |  | ME |  | Radio Access Network |  | Core Network | **x** |

|  |
| --- |
|  |
| ***Title:***  | N32 interface |
|  |  |
| ***Source to WG:*** | Nokia, Nokia Shanghai Bell |
| ***Source to TSG:*** | S3 |
|  |  |
| ***Work item code:*** | 5G\_eSBA |  | ***Date:*** | 7.8.2020 |
|  |  |  |  |  |
| ***Category:*** | **F** |  | ***Release:*** | Rel-16 |
|  | *Use one of the following categories:****F*** *(correction)****A*** *(mirror corresponding to a change in an earlier release)****B*** *(addition of feature),* ***C*** *(functional modification of feature)****D*** *(editorial modification)*Detailed explanations of the above categories canbe found in 3GPP [TR 21.900](http://www.3gpp.org/ftp/Specs/html-info/21900.htm). | *Use one of the following releases:Rel-8 (Release 8)Rel-9 (Release 9)Rel-10 (Release 10)Rel-11 (Release 11)Rel-12 (Release 12)**Rel-13 (Release 13)Rel-14 (Release 14)Rel-15 (Release 15)Rel-16 (Release 16)* |
|  |  |
| ***Reason for change:*** | Alignment with CT4 in line with LS S3-201464 on Misalignment on HTTP connections for N32-c and on N32-f contexts termination.Replacement of long-lived with short-lived connection is needed.In 13.2.1 Clarifications on the deployment of PRINS is needed, if no IPX is in between. CT4 is not ruling out to use PRINS also between SEPPs only as deployment, but SA3 should clarify that this is not preferrred, since TLS can do the job in this caseIn 13.2.2.1 clarification on N32-c is usage is missing  In 13.2.2.2 step 6, TLS session may be kept open, it should not be mandatory. Also the text suggests that N32-c and N32-f share the same HTTPS connection, therefore changed to say “TLS sessions” In 13.2.2.4.1 Type reveiving -> receiving and other small typos |
|  |  |
| ***Summary of change:*** | Clarification on deployment added by NoteClafication on N32-c for security mechanism negotiation and when PRINS is used |
|  |  |
| ***Consequences if not approved:*** | Unclear deployment scenarioN32-c usage unclear |
|  |  |
| ***Clauses affected:*** | 3.1, 13.2.1, 13.2.2.1, 13.2.2.2, 13.2.2.3, 13.2.2.4.1, 1.2.2.4.2, 13.2.4.4.1, 13.2.4.5.2, 13.2.4.7  |
|  |  |
|  | **Y** | **N** |  |  |
| ***Other specs*** |  | **x** |  Other core specifications  | TS/TR ... CR ...  |
| ***affected:*** |  | **x** |  Test specifications | TS/TR ... CR ...  |
| ***(show related CRs)*** |  | **x** |  O&M Specifications | TS/TR ... CR ...  |
|  |  |
| ***Other comments:*** |  |
|  |  |
| ***This CR's revision history:*** | update of S3-201796 |

\*\*\*\*\*\*\*\*\*\*\*\* START OF CHANGES

13.2 Application layer security on the N32 interface

13.2.1 General

The internetwork interconnect allows secure communication between service-consuming and a service-producing NFs in different PLMNs. Security is enabled by the Security Edge Protection Proxies (SEPPs) of both networks, henceforth called cSEPP and pSEPP respectively. The SEPPs enforce protection policies regarding application layer security thereby ensuring integrity and confidentiality protection for those elements to be protected.

For the usage of application layer security on the N32 interface with PRINS, it is assumed that there are interconnect providers between cSEPP and pSEPP. The interconnect provider, the cSEPP's operator has a business relationship with, is called cIPX, while the interconnect provider, the pSEPP's operator has a business relationship with, is called pIPX. There could be further interconnect providers in between cIPX and pIPX, but they are assumed to be transparent and simply forward the communication.

NOTE: Deployment scenarios, where PRINS can be applied to direct cSEPP-pSEPP without interconnect providers is possible, but not preferred since TLS is sufficient in this case.

The SEPPs use JSON Web Encryption (JWE, specified in RFC 7516 [59]) for protecting messages on the N32-f interface, and the IPX providers use JSON Web Signatures (JWS, specified in RFC 7515 [45]) for signing their modifications needed for their mediation services.

For illustration, consider the case where a service-consuming NF sends a message to a service-producing NF. If this communication is across PLMN operators over the N32 interface, as shown in Figure 13.2.1-1 below, the cSEPP receives the message and applies symmetric key based application layer protection, as defined in clause 13.2 of the present document. The resulting JWE object is forwarded to intermediaries. The pIPX and cIPX can offer services that require modifications of the messages transported over the interconnect (N32) interface. These modifications are appended to the message as digitally signed JWS objects which contain the desired changes. The pSEPP, which receives the message from pIPX, validates the JWE object, extracts the original message sent by the NF, validates the signature in the JWS object and applies patches corresponding to the modifications by intermediaries. The pSEPP then forwards the message to the destination NF.

The N32 interface, if used for PRINS, consists of:

- N32-c connection, for managing the N32 interface, and

- N32-f connection, for sending of JWE and JWS protected messages between the SEPPs.

.

****

**Figure 13.2.1-1: Overview of PRINS**

13.2.2 N32-c connection between SEPPs

13.2.2.1 General

Via N32-c the security mechanism as described in the procedure in clause 13.5 are negotiated.

When the negotiated security mechanism to use over N32is PRINS (see overview in clause 13.2.1), the SEPPs use the established TLS connection (henceforth referred to as N32-c connection) to negotiate the N32-f specific associated security configuration parameters required to enforce application layer security on HTTP messages exchanged between the SEPPs. A second N32-c connection is established by the receiving SEPP, if the receiving SEPP needs to send HTTP Requests.

The N32-c connection is further used for the following purposes when PRINS is used:

- Key agreement: The SEPPs independently export keying material associated with the first N32-c connection between them and use it as the pre-shared key for generating the shared session key required.

- Parameter exchange: The SEPPs exchange security related configuration parameters that they need to protect HTTP messages exchanged between the two Network Functions (NF) in their respective networks.

- Error handling: The receiving SEPP sends an error signalling message to the peer SEPP when it detects an error on the N32-f interface.

The following security related configuration parameters may be exchanged between the two SEPPs:

a. Modification policy. A modification policy, as specified in clause 13.2.3.4, indicates which IEs can be modified by an IPX provider of the sending SEPP.

b. Data-type encryption policy. A data-type encryption policy, as specified in 13.2.3.2, indicates which types of data will be encrypted by the sending SEPP.

c. Cipher suites for confidentiality and integrity protection, when application layer security is used to protect HTTP messages between them.

d. N32-f precontext identifier values.N32-f precontext identifier values, as specified in clause 13.2.2.4.1, are used by each SEPP to construct a common N32-f context ID that identifies the set of security related configuration parameters applicable to a protected message received from a SEPP in a different PLMN.

13.2.2.2 Procedure for key agreement and parameter exchange

1. The two SEPPs shall perform the following cipher suite negotiation to agree on a cipher suite to use for protecting NF service-related signalling over N32-f.

1a. The SEPP which initiated the first N32-c connection shall send a Security Parameter Exchange Request message to the responding SEPP including the initiating SEPP’s supported cipher suites. The cipher suites shall be ordered in initiating SEPP’s priority order. The SEPP shall provide a N32-f precontext ID for the responding SEPP. The precontext IDs are 32-bit random integers, represented as 0-left padded strings of hexadecimal digits.

1b. The responding SEPP shall compare the received cipher suites to its own supported cipher suites and shall select, based on its local policy, a cipher suite, which is supported by both initiating SEPP and responding SEPP.

1c. The responding SEPP shall send a Security Parameter Exchange Response message to the initiating SEPP including the selected cipher suite for protecting the NF service- related signalling over N32. The responding SEPP shall provide a N32-f precontext ID for the initiating SEPP.

1d. The SEPPs shall create the N32-f context ID as follows:

Initiating SEPP's N32-f precontext ID | responding SEPP's N32-f precontext ID

2. The two SEPPs may perform the following exchange of Data-type encryption policies and Modification policies. Both SEPPs shall store protection policies sent by the peer SEPP:

2a. The SEPP which initiated the first N32-c connection shall send a Security Parameter Exchange Request message to the responding SEPP including the initiating SEPP’s Data-type encryption policies, as described in clause 13.2.3.2, and Modification policies, as described in clause 13.2.3.4.

2b. The responding SEPP shall store the policies if sent by the initiating SEPP.

2c. The responding SEPP shall send a Security Parameter Negotiation Response message to the initiating SEPP with the responding SEPP’s suite of protection policies.

2d. The initiating SEPP shall store the protection policy information if sent by the responding SEPP.

3. The two SEPPs shall exchange IPX security information lists that contain information on IPX public keys or certificates that are needed to verify IPX modifications at the receiving SEPP.

4. The two SEPPs shall export keying material from the TLS session established between them using the TLS export function. For TLS 1.2, the exporter specified in RFC 5705 [61] shall be used. For TLS 1.3, the exporter described in section 7.5 of RFC 8446 [60] shall be used. The exported key shall be used as the master key to derive session keys and IVs for the N32-f context as specified in clause 13.2.4.4.1.

5. The responding SEPP in the first N32-c connection shall now setup a second N32-c connection by establishing a mutually authenticated TLS connection with the peer SEPP.

6. The two SEPPs start exchanging NF to NF service-related signalling over N32-f and may keep the TLS sessions open for:

- any further N32-c communication that may occur over time while application layer security is applied to N32-f, or

- any further N32-c and N32-f communication, if TLS is used to protect N32-f.

13.2.2.3 Procedure for error detection and handling in SEPP

Errors can occur on an active N32-c connection or on one or more N32-f connections between two SEPPs.

When an error is detected, the SEPP shall map the error to an appropriate cause code. The SEPP shall create a signalling message to inform the peer SEPP, with cause code as one of its parameters.

The SEPP shall use the N32-c connection it initiated to send the signalling message to the peer SEPP.

If the error occurred in the processing of the one or more N32-f message(s), the SEPP shall include the corresponding message ID (s), obtained from the metadata section of the N32-f message, as a parameter in the signalling message. This allows the peer SEPP to identify the source message(s) (HTTP Request or Response) on which the other SEPP found the error.

NOTE: Local action taken by either SEPP is out of 3GPP scope.

13.2.2.4 N32-f Context

13.2.2.4.0 N32-f parts

The N32-f context consists of the following main parts as illustrated in Figure 13.2.2.4.0-1:

1. N32-f context ID

2. N32-f peer information

3. N32-f security context

4. N32-f context information

****

**Figure 13.2.2.4.0-1: N32-f context overview**

13.2.2.4.1 N32-f context ID

The N32-f context ID is used to refer to an N32-f context. The SEPPs shall create the N32-f context ID during the N32-c negotiation and use it over N32-f to inform the receiving peer which security context to use for decryption of a received message.

The SEPPs shall create the N32-f context ID by combining the two N32-f precontext IDs, obtained during the N32-c negotiation. To avoid collision of the N32-f context ID value, the SEPPs shall select the N32-f precontext ID as a random value during the exchange over N32-c.

During transfer of application data over N32-f, the SEPP shall include the N32-f context ID in a separate IE in the metadata part of the JSON structure, see clause 13.2.4.2. The receiving SEPP shall use this information to apply the correct key and parameters during decryption and validation.

13.2.2.4.2 N32-f peer information

The N32-f connection between SEPPs is bidirectional and consists of the two SEPP endpoints and possibly up to two IPX providers. The SEPPs are identified by the PLMN ID and additionally a SEPP ID to distinguish between several SEPPs in the same PLMN. The remote SEPP address is necessary for routing the messages to the correct destination.

The N32-f peer information shall consist of the following parameters:

- Remote PLMN ID;

- Remote SEPP ID;

- Remote SEPP address.

13.2.2.4.3 N32-f security context

The N32-c initial handshake described in clause 13.2.2.2 establishes session keys, IVs and negotiated cipher suites. Counters are used for replay protection. Modification policies are identified by modification policy IDs, to be able to verify received messages that have undergone IPX modifications.

The N32 security context shall consist of the following parameters:

- Session keys

- Negotiated cipher suites

- Data type encryption policy IDs

- Modification policy IDs (if IPXs are used)

- Counters

- IVs

- List of security information of the IPX providers connected to the SEPPs (IPX security information list)

- IPX provider identifier

- List of raw public keys or certificates for that IPX

13.2.2.4.4 N32-f context information

The N32 context information shall consist of the following parameters:

- Validity.

- Usage (PRINS).

13.2.3 Protection policies for N32 application layer solution

13.2.3.1 Overview of protection policies

The protection policy suite is comprised of a data-type encryption policy and a modification policy. Together, these policies determine which part of a certain message shall be confidentiality protected and which part of a certain message shall be modifiable by IPX providers. The SEPP shall apply the protection policies for application layer protection of messages on the N32-f interface.

There are two types of protection policies, namely:

- Data-type encryption policy: specifies which data types need to be confidentiality protected;

- Modification policy: specifies which IEs are modifiable by intermediaries.

In addition, there is a mapping between the data-types in the data-type encryption policy and the IEs in NF API descriptions which is given in a NF-API data-type placement mapping.

13.2.3.2 Data-type encryption policy

The SEPP shall contain an operator-controlled protection policy that specifies which types of data shall be encrypted. The data-types defined are the following:

- Data of the type 'SUPI';

- Data of the type 'authentication vector';

- Data of the type 'location data';

- Data of the type 'cryptographic material';

- Data of the type 'authorization token'.

The policy shall be specific per roaming partner. The policy shall contain a policy identifier and a release number referring to the release it is applicable for.

The data-type encryption policies in the two partner SEPPs shall be equal to enforce a consistent ciphering of IEs on N32-f.

13.2.3.3 NF API data-type placement mapping

Each NF API data-type placement mapping shall contain the following:

- Which IEs contain data of the type 'SUPI' or type 'NAI'.

- Which IEs contain data of the type 'authentication vector’.

- Which IEs contain data of the type 'location data'.

- Which IEs contain data of the type 'cryptographic material'.

- Which IEs contain data of the type 'authorization token'.

The location of the IEs refers to the location of the IEs after the SEPP has rewritten the message for transmission over N32-f.

An NF API data-type placement mapping shall furthermore contain data that identifies the NF API, namely

- The name of the NF;

- The API version;

- An identifier for the NF API data-type placement mapping;

- The NF’s 3GPP Release version.

NOTE: Larger networks can contain multiple NFs with the same API, e.g. three AMFs. The NF API policy applies to all NFs with the same API.

The NF API data-type placement mapping shall reside in the SEPP.

13.2.3.4 Modification policy

The SEPP shall contain an operator-controlled policy that specifies which IEs can be modified by the IPX provider directly related to this particular SEPP. These IEs refer to the IEs after the sending SEPP has rewritten the message.

Each PLMN-operator shall agree the modification policy with the IPX provider it has a business relationship with prior to establishment of an N32 connection. Each modification policy applies to one individual relation between PLMN-operator and IPX provider. To cover the whole N32 connection, both involved roaming partners shall exchange their modification policies. Both complementary modification policies shall comprise the overall modification policy for this specific N32 connection.

NOTE 1: In order to validate modifications for messages received on the N32-f interface, the operator’s roaming partners will have to know the overall modification policy.

NOTE 2: Modification includes removal and addition of new IE. IEs therefore may not be present in the rewritten message.

The IEs that the IPX is allowed to modify shall be specified in a list giving an enumeration of JSON paths within the JSON object created by the SEPP. Wildcards may be used to specify paths.

This policy shall be specific per roaming partner and per IPX provider that is used for the specific roaming partner.

The modification policy shall reside in the SEPP.

For each roaming parter, the SEPP shall be able to store a policy for sending in addition to one for receiving.

The following basic validation rules shall always be applied irrespective of the policy exchanged between two roaming partners:

- IEs requiring encryption shall not be inserted at a different location in the JSON object.

13.2.3.5 Provisioning of the policies in the SEPP

The SEPP shall contain an interface that the operator can use to manually configure the protection policies in the SEPP.

The SEPP shall be able to store and process the following policies for outgoing messages:

- A generic data-type encryption policy;

- Roaming partner specific data-type encryption policies that will take precedence over a generic data-type encryption policy if present;

- NF API data-type placement mappings;

- Multiple modification policies, to handle modifications that are specific per IPX provider and modification policies that are specific per IPX provider and roaming partner.

The SEPP shall also be able to store and process the following policies for incoming messages during the initial connection establishment via N32-c:

- Roaming partner specific data-type encryption policies;

- Roaming partner specific modification policies that specify which fields can be modified by which of its IPX providers.

13.2.3.6 Precedence of policies in the SEPP

This clause specifies the order of precedence of data-type encryption policies and modification policies available in a SEPP.

In increasing order of precedence, the following policies apply for a message to be sent on N32:

1. The set of default rules specified in the present specification:

- For the data-type encryption policy, the rules on data-types that are mandatory to be encrypted according to clause 5.9.3.3.

- For the modification policy, the basic validation rules defined in clause 13.2.3.4.

2. Manually configured policies:

 - For the data-type encryption policy: rules according to clause 13.2.3.2, on a per roaming partner basis.

 - For the modification policy: rules according to clause 13.2.3.4, per roaming partner and per IPX provider that is used for the specific roaming partner.

NOTE 1: It is assumed that operators agree both data-type encryption and modification policy in advance, for example as part of their bilateral roaming agreement. The protection policies exchanged via N32-c during the initial connection establishment only serve the purpose of detecting possible misconfigurations.

NOTE 2: It is assumed that the default rules and manually configured policies do not overlap or contradict each other. The manually configured policies are used to extend the protection by the default rules in the present document and are applied on top of them.

When a SEPP receives a data-type encryption or modification policy on N32-c as specified in clause 13.2.2.2, it shall compare it to the one that has been manually configured for this specific roaming partner and IPX provider. If a mismatch occurs for one of the two policies, the SEPP shall perform one of the following actions, according to operator policy:

- Send the error message <TBD> to the peer SEPP.

Editor's Note: The error message type needs to be specified by CT4.

- Create a local warning.

13.2.4 N32-f connection between SEPPs

13.2.4.1 General

The SEPP receives HTTP/2 request/response messages from the Network Function. It shall perform the following actions on these messages before they are sent on the N32-f interface to the SEPP in the other PLMN:

a) It parses the incoming message and, if present, rewrites the telescopic FQDN of the receiving NF to obtain the original FQDN as described in clause 13.1.

b) It reformats the message to produce the input to JSON Web Encryption (JWE) [59] as described in clause 13.2.4.3.

c) It applies JWE to the input created in b) to protect the reformatted message as described in clause 13.2.4.4.

d) It encapsulates the resulting JWE object into a HTTP/2 message (as the body of the message) and sends the HTTP/2 message to the SEPP in the other PLMN over the N32-f interface.

The message may be routed via the cIPX and pIPX nodes. These IPX nodes may modify messages as follows:

a) The IPX node recovers the cleartext part of the HTTP message from the JWE object, modifies it according to the modification policy, and calculates an "operations" JSON Patch object. It then creates a temporary JSON object with the "operators" JSON Patch object and some other parameters for replay protection etc. as described in clause 13.2.4.5.1.

b) The IPX node uses the temporary JSON object as input into JSON Web Signature (JWS) [45] to create a JWS object, as described in clause 13.2.4.5.2.

c) The IPX node appends the JWS object to the received message and sends it to the next hop.

The JWS objects generated by the two IPX providers form an auditable chain of modifications that to the receiving SEPP shall apply to the parsed message after verifying that the patches conform to the modification policy.

Encryption of IEs shall take place end to end between cSEPP and pSEPP.

A SEPP shall not include IEs in the clear that are encrypted elsewhere in the JSON object.

A SEPP shall verify that an intermediate IPX has not moved or copied an encrypted IE to a location that would be reflected from the producer NF in an IE without encryption.

13.2.4.2 Overall Message payload structure for message reformatting at SEPP

The SEPP reformats an HTTP message received from an internal Network Function into two temporary JSON objects that will be intput to JWE:

a. The **dataToIntegrityProtect**, containing information that is only integrity protected. It consists of the following:

- clearTextEncapsulationMessage: contains the complete original HTTP message, excluding attribute values which require encryption and, including the pseudo-header fields, HTTP headers and HTTP message body.

- metadata: contains SEPP generated information i.e. authorizedIPX ID, N32-f message ID and N32-f context ID.

b. The **dataToIntegrityProtectAndCipher**: contains attribute values of the original message that require both encryption and integrity protection.

****

**Figure 13.2.4.2-1 Example of JSON representation of a reformatted HTTP message**

13.2.4.3 Message reformatting in sending SEPP

13.2.4.3.1 dataToIntegrityProtect

13.2.4.3.1.1 clearTextEncapsulatedMessage

The clearTextEncapsulatedMessage is a JSON object that contains the non-encrypted portion of the original message.Specifically, it consists of the following objects:

1.a) Pseudo\_Headers – the JSON object that includes all the Pseudo Headers in the message.

- For HTTP Request messages, the object contains one entry for each of the ":method", ":path", ":scheme" and ":authority" pseudo headers. If the ":path" pseudoheader contains multiple parts separated by a slash (/) or includes a query parameter (following a "?"), an array is used to represent :path, with one element per part of the path (i.e. per "directory").

NOTE: This enables encryption of individual elements of the path (e.g. if SUPI is passed).

- For HTTP Response messages, the object contains the ":status" pseudo header.

1.b) HTTP\_Headers – the JSON object that includes all the Headers in the message.

All the headers of the request are put into a JSON array called HTTP\_Headers.Each entry contains a header name and value, where the value part can be an encoded index to the dataToIntegrityProtectAndCipher block, if the header value is encrypted.

1.c) Payload – the JSON object that includes the content of the payload of the HTTP message.

Each attribute or IE in the payload shall form a single entry in the Payload JSON object. If there is any attribute value that requires encryption, the value shall be replaced by null. The SEPP shall calculate a JSON patch document, **dataToIntegrityProtectAndCipher** (described in clause 13.2.4.3.2), that replaces any nulls with the required values. This is an array of objects as per RFC 6902 [64].

13.2.4.3.1.2 metadata

The JSON object containing information added by the sending SEPP. It shall contain:

a) N32-f **message ID**: Unique identifier (64-bit integer) representing a HTTP Request/Response transaction between two SEPPs. The N32-f message ID is generated by the sending SEPP and included in the HTTP Request sent over the N32 interface. The receiving SEPP uses the same N32-f message ID when it responds back with a HTTP Response. The N32-f message ID is included in the metadata portion of the JSON structure.

b) **authorizedIPX** **ID**: String identifying the first hop IPX (cIPX or pIPX) that is authorized to update the message. This field shall always be present. When there is no IPX that is authorized to update, the value of this field is set to null. The sending SEPP selects one of the IPX providers from the list exchanged with the other SEPP during parameter exchange over N32-c and includes its identifier value in this field.

c) **N32-f context ID**: Unique identifier representing the N32-f context information used for protecting the message. This is obtained from N32-f precontext IDs exchanged during parameter exchange over N32-c (clause 13.2.2.4.1).

13.2.4.3.2 dataToIntegrityProtectAndCipher

The dataToIntegrityProtectAndCipher is a JSON patch document as per RFC 6902 [64] that contains all the attribute values that require both encryption and integrity protection. Attribute values may come from any part of the original HTTP message – Pseudo\_Headers, HTTP\_Headers and Payload. The dataToIntegrityProtectAndCipher is an array of JSON objects. If no IEs require protection, then this shall be an array with no elements. The sending SEPP shall construct the JSON patch document so that the receiving SEPP can insert the attribute values into the clearTextEncapsulatedMessage by applying the JSON patch document.

13.2.4.4 Protection using JSON Web Encryption (JWE)

13.2.4.4.0 General

The SEPP shall use JSON Web Encryption (JWE) as specified in RFC 7516 [59] for the protection of reformatted HTTP messages between the SEPPs. All encryption methods supported by JWE are AEAD methods, i.e. methods that encrypt and integrity protect in one single operation and can additionally integrity protect additional data.

The dataToIntegrityProtectAndCipher and dataToIntegrityProtect blocks shall be input to JWE as plaintext and JWE Additional Authenticated Data (AAD) respectively. The JWE AEAD algorithm generates JWE encrypted text (ciphertext) and a JWE Authentication Tag (Message Authentication Code). The ciphertext is the output from symmetrically encrypting the plaintext, while the authentication tag is a value that verifies the integrity of both the generated ciphertext and the Additional Authenticated Data.

The Flattened JWE JSON Serialization syntax shall be used to represent JWE as a JSON object.

The session key shared between the two SEPPs, as specified in clause 13.2.4.4.1, shall be used as the Content Encryption Key (CEK) value to the algorithm indicated in the Encryption algorithm ("enc") parameter in the JOSE header. The algorithm ("alg") parameter in the JOSE header denoting the key exchange method shall be set to "dir", i.e. "Direct use of a shared symmetric key as the CEK".

The 3GPP profile for supported cipher suites in the "enc" parameter is described in clause 13.2.4.9.

The generated JWE object shall be transmitted on the N32-f interface in the payload body of a SEPP to SEPP HTTP/2 message.

13.2.4.4.1 N32-f key hierarchy

The N32-f key hierarchy is based on the N32-f master key generated during the N32-c initial handshake by TLS key export. The N32-f key hierarchy consists of two pairs of session keys and two pairs of IV salts, which are used in two different HTTP/2 sessions. In one Session the N32-c initiator acts as the HTTP client and in the second the N32-c responder acts as the client.

If the exported master secret is reused to set up multiple HTTP sessions or to set up new HTTP sessions on stream ID exhaustion, a new, unique, N32-f Context ID shall be generated to avoid key and IV re-use.

The master key shall be obtained from the TLS exporter. The export function takes 3 arguments: Label, Context, Length (in octets) of the desired output. For the N32 Master key derivation, the label shall be "EXPORTER\_3GPP\_N32\_MASTER", the Context shall be "" (the empty string) and the Length shall be 64.

Editor’s Note: The exporter label for this usage should be registered with IANA

The N32 key derivation function N32-KDF shall be based on HKDF [62] and shall use only the HKDF-Expand function as the initial key material has been generated securely:

 N32-KDF (label, L) = HKDF-Expand (N32-f master key, "N32" || N32-Context-ID || label, L),

where

 - label is a string used for key separation,

 - L is the length of output keying material in octets.

Each run of N32-KDF (label, L) produces either one session key or one IV salt.

There are two pairs of session keys and IV salts to be derived.

NOTE: In AES-GCM re-use of one IV may reveal the integrity key (Joux’s Forbidden attack). The binding of session keys and IV salts to N32-f context IDs and labels is essential to protect against inadvertent use of the same key with a repeated IV.

The labels for the JWE keys are:

- "parallel\_request\_key"

- "parallel\_response\_key"

- "reverse\_request\_key", and

- "reverse\_response\_key".

The keys derived with labels starting parallel shall be used for request/responses in an HTTP session with the N32-c initiating SEPP acting as the client (i.e. in parallel to the N32-c connection). The keys derived with the labels starting reverse shall be used for an HTTP session with the N32-c responding SEPP acting as the client.

To generate the IV salts, the length is 8 and the labels are:

- "parallel\_request\_iv\_salt",

- "parallel\_response\_iv\_salt",

- "reverse\_request\_iv\_salt", and

- "reverse\_response\_iv\_salt".

The 96-bit nonce for AES\_GCM shall be constructed as the concatenation of the IV salt (8 octets, 64-bits) and the sequence counter, SEQ, following section 8.2.1 of NIST Special Publication 800-38D [63] :

 Nonce = IV salt || SEQ.

The sequence counter shall be a 32-bit unsigned integer that starts at zero and is incremented for each invocation of the encryption. A different sequence counter shall be maintained for each IV salt.

13.2.4.5 Message modifications in IPX

13.2.4.5.1 modifiedDataToIntegrityProtect

****

**Figure 13.2.4.5.1-1 Example of JSON representation of IPX provider modifications**

This is a temporary JSON object generated by an IPX provider as it modifies the original message. It shall contain the following:

a) **Operations** – This is a JSON patch document that captures IPX modifications based on RFC 6902 [64]. If no patch is required, the operations element shall be set to null.

b) **Identity** – This is the identity of the IPX performing the modification.

c) **Tag** – A JSON string element to capture the “tag” value (JWE Authentication tag) in the JWE object generated by the sending SEPP. This is required for replay protection.

NOTE: Since there is no central registry that can ensure unique IPX Identities, it is expected that an IPX will include its Fully Quantified Domain Name (FQDN) in the JSON modification object.

13.2.4.5.2 Modifications by IPX

NOTE 1: It is assumed that operators act as a certification authority for IPX providers they have a direct business relationship with. In order to authorize N32-f message modifications, operators sign a digital certificate for each of these IPX providers and provide it to both the IPX provider itself as well as their roaming partners to enable them to validate any modifications by this IPX provider.

Only cIPX and pIPX shall be able to modify messages between cSEPP and pSEPP. In cases of messages from cSEPP to pSEPP, the cIPX is the first intermediary, while the pIPX is the second intermediary. In cases of messages from pSEPP to cSEPP the pIPX is the first intermediary, while the cIPX is the second intermediary.

The first intermediary shall parse the encapsulated request (i.e. the clearTextEncapsulationMsg in the dataToIntegrityProtect block) and determine which changes are required. The first intermediary creates an Operations JSON patch document to describe the differences between received and desired message, using the syntax and semantic from RFC 6902 [64], such that, when applying the JSON patch to the encapsulated request the result will be the desired request. If no patch is required, the operations element is null.

NOTE 2: It is necessary to create a JWS object even if no patch is required to prevent deletion of modifications.

The first intermediary shall create a modifiedDataToIntegrityProtect JSON object as described in clause 13.2.4.5.1. The JSON object shall include the intermediary’s identity and the JWE authentication tag, which associates this update by the intermediary with the JWE object created by the sending SEPP.

The first intermediary shall use the modifiedDataToIntegrityProtect JSON object as input to JWS to create a JWS object. The first intermediary shall append the generated JWS object to the payload in the HTTP message and then send the messageto the next hop.

The second intermediary shall parse the encapsulated request, apply the modifications described in the JSON patch appended by the first intermediary and determine further modifications required for obtaining the desired request. The second intermediary shall record these modifications in an additional JSON patch against the JSON object resulting from application of the first intermediary's JSON patch. If no patch is required, the operations element for the second JSON patch is null.

The second intermediary shall create a modifiedDataToIntegrityProtect JSON object as described in clause 13.2.4.5.1. It shall include its identity and the JWE authentication tag, which associates this update by the second intermediary with the JWE object created by the sending SEPP.

The second intermediary shall use the modifiedDataToIntegrityProtect JSON object as input to JWS to create a JWS object. The second intermediary shall append the generated JWS object to the payload in the HTTP message and then send the message to the receiving SEPP.

13.2.4.6 Protecting IPX modifications using JSON Web Signature (JWS)

The IPX providers shall use JSON Web Signature (JWS) as specified in RFC 7515 [45] for the protection of IPX provider modified attributes. The mechanism described in this clause uses signatures, i.e. asymmetric methods, with private/public key pairs.

More specifically, when an IPX node modifies one or more attributes of the original HTTP message and creates a modifiedDataToIntegrityProtect object to record its modifications, it shall use JWS to integrity protect the modifiedDataToIntegrityProtect object.

The IPX provider shall use its private key as input to JWS for generating the signature representing the contents of the modifiedDataToIntegrityProtect object.

The "alg" parameter in the JOSE header indicates the chosen signature algorithm. The 3GPP profile for supported algorithms is described in clause 13.2.4.9.

The Flattened JWS JSON Serialization syntax shall be used to represent JWS as a JSON object.

13.2.4.7 Message verification by the receiving SEPP

The receiving SEPP shall decrypt the JWE ciphertext using the shared session key and the following parameters obtained from the JWE object – Initialization Vector, Additional Authenticated Data value (clearTextEncapsulatedMessage in "aad") and JWE Authentication Tag ( "tag").

The receiving SEPP shall check the integrity and authenticity of the clearTextEncapsulatedMessage and the encrypted text by verifying the JWE Authentication Tag in the JWE object with the JWE AAD algorithm. The algorithm returns the decrypted plaintext (dataToIntegrityProtectAndCipher) only if the JWE Authentication Tag is correct.

The receiving SEPP shall apply the decrypted JSON patch in the dataToIntProtectAndCipher to the clearTextEncapsulatedMessage. The receiving SEPP shall use the NF API data type placement mapping and the encryption policy to verify that the correct information elements have been encrypted.

The receiving SEPP shall next verify IPX provider updates, if included, by verifying the JWS signatures added by the intermediaries. The SEPP shall verify the JWS signature, using the corresponding raw public key or certificate that is contained in the IPX provider’s security information list obtained during parameter exchange in the related N32-c connection setup or, alternatively, has been configured for the particular peer SEPP. It shall then check that the raw public key or certificate of the JWS signature IPX's Identity in the modifiedDataToIntegrity block matches to the IPX provider referred to in the "authorizedIPX ID" field added by the sending SEPP, based on the information given in the IPX provider security information list.

The receiving SEPP shall check whether the modifications performed by the intermediaries were permitted by the respective modification policies. If this is the case, the receiving SEPP shall apply the patches in the Operations field in order, perform plausibility checks, and create a new HTTP request according to the "patched" clearTextEncapsulatedMessage.

The receiving SEPP shall verify that the PLMN-ID contained in the incoming N32-f message matches the PLMN-ID in the related N32-f context.

13.2.4.8 Procedure

The following clause illustrates the message flow between the two SEPPs with modifications from cIPX and pIPX.

****

**Figure 13.2.4.8-1 Message flow between two SEPPs**

1. The cSEPP receives an HTTP request message from a network function. If the message contains a telescopic FQDN, the cSEPP removes its domain name from this FQDN to obtain the original FQDN as described in clause 13.1.

2. The cSEPP shall reformate the HTTP Request message as follows:

a. The cSEPP shall generate blocks (JSON objects) for integrity protected data and encrypted data, and protecting them:

The cSEPP shall encapsulate the HTTP request into a clearTextEncapsulatedMessage block containing the following child JSON objects:

- Pseudo\_Headers

- HTTP\_Headers with one element per header of the original request.

- Payload that contains the message body of the original request.

For the attributes that require end-to-end encryption between the two SEPPs, the cSEPP shall create a JSON patch document, whose application inserts the values to the clearTextEncapsulatedMessage. The SEPP shall replace the attribute's value in the clearTextEncapsulatedMessage by null.

The cSEPP shall create a metadata block that contains the N32-f context ID, message ID generated by the cSEPP for this request/response transaction and next hop identity.

The cSEPP shall protect the dataToIntegrityProtect block and the dataToIntegrityProtectAndCipher block as per clause 13.2.4.4. This results in a single JWE object representing the protected HTTP Request message.

b. The cSEPP shall generate payload for the SEPP to SEPP HTTP message:

 The JWE object becomes the payload of the new HTTP message generated by cSEPP.

3. The cSEPP shall use HTTP POST to send the HTTP message to the first intermediary.

4. The first intermediary (e.g. visited network's IPX provider) shall create a new modifiedDataToIntegrityProtect JSON object with three elements:

a. The Operations JSON patch document contains modifications performed by the first intermediary as per RFC 6902 [64].

b. The first intermediary shall include its own identity in the Identity field of the modifiedDataToIntegrityProtect.

c. The first intermediary shall copy the "tag" element, present in the JWE object generated by the cSEPP, into the modifiedDataToIntegrityProtect object. This acts as a replay protection for updates made by the first intermediary.

The intermediary shall execute JWS on the modifiedDataToIntegrityProtect JSON object and append the resulting JWS object to the message.

5. The first intermediary shall send the modified HTTP message request to the second intermediary (e.g. home network's IPX) as in step 3.

6. The second intermediary shall perform further modifications as in step 4 if required. The second intermediary shall further execute JWS on the modifiedDataToIntegrityProtect JSON object and shall append the resulting JWS object to the message.

7. The second intermediary shall send the modified HTTP message to the pSEPP as in step 3.

 NOTE 1: The behaviour of the intermediaries is not normative, but the pSEPP assumes that behaviour for processing the resulting request.

8. The pSEPP receives the message and shall perform the following actions:

- The pSEPP extracts the serialized values from the components of the JWE object.

- The pSEPP invokes the JWE AEAD algorithm to check the integrity of the message and decrypt the dataToIntegrityProtectAndCipher block. This results in entries in the encrypted block becoming visible in cleartext.

- The pSEPP updates the clearTextEncapsulationMessage block in the message by applying the JSON patch document in the dataToIntegrityProtectAndCipher block. The receiving SEPP shall use the NF API data type placement mapping and the encryption policy to verify that the correct information elements have been encrypted, as described in clause 13.2.4.7.

- The pSEPP then verifies IPX provider updates of the attributes in the modificationsArray. It checks whether the modifications performed by the intermediaries were permitted by policy.

 The pSEPP further verifies that the PLMN-ID contained in the message is equal to the "Remote PLMN-ID" in the related N32-f context.

- The pSEPP updates the modified values of the attributes in the clearTextEncapsulationMessage in order.

The pSEPP shall re-assemble the full HTTP Request from the contents of the clearTextEncapsulationMessage.

9. The pSEPP shall send the HTTP request resulting from step 8 to the home network's NF.

10.-18. These steps are analogous to steps 1.-9.

13.2.4.9 JOSE profile

SEPPs shall follow the JWE profile defined in TS 33.210 [3] with the restriction that it shall only use AES GCM with a 128-bit or 256-bit key. The security considerations for the use of AES GCM in section 8.4 of RFC 7516 [59] shall be taken into account. In particular, the same key shall not be used more than 232 times and an IV value shall not be used more than once with the same key.

SEPPs and IPXs shall follow the JWS profile as defined in TS 33.210 [3] with the restriction that they shall only use ES256 algorithm.

\*\*\*\*\*\*\*\*\*\*\*\* END OF CHANGES