**3GPP TSG-SA3 Meeting #102-e** S3-210382-r1

**e-meeting, 18 – 29 January 2021, Online**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *CR-Form-v12.0* | | | | | | | | |
| **CHANGE REQUEST** | | | | | | | | |
|  | | | | | | | | |
|  | **33.501** | **CR** | 1046 | **rev** | **1** | **Current version:** | **15.11.0** |  |
|  | | | | | | | | |
| *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:*** | Align the JSON format on encryption IE with CT4 in Rel15 | | | | | | | | | |
|  |  | | | | | | | | | |
| ***Source to WG:*** | Huawei, Hisilicon, Nokia, Nokia Shanghai Bell | | | | | | | | | |
| ***Source to TSG:*** | S3 | | | | | | | | | |
|  |  | | | | | | | | | |
| ***Work item code:*** | 5GS\_Ph1-SEC | | | | |  | ***Date:*** | | | 2020-12-25 |
|  |  | | | |  | |  | | |  |
| ***Category:*** | **F** |  | | | | | ***Release:*** | | | Rel-15 |
|  | *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 can be 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-15 (Release 15) Rel-16 (Release 16) Rel-17 (Release 17) Rel-18 (Release 18)* | |
|  |  | | | | | | | | | |
| ***Reason for change:*** | | In SA3 #101e meeting, it was agreed to align the JSON format on encryption IE with CT4. Hence, the figure 13.2.4.2-1 Example of JSON representation of a reformatted HTTP message and the related procedures should be updated according to the CT4 specification. | | | | | | | | |
|  | |  | | | | | | | | |
| ***Summary of change:*** | | Figure and the related procedures update to align with CT4. | | | | | | | | |
|  | |  | | | | | | | | |
| ***Consequences if not approved:*** | | Figure and the related procedures is not align with CT4 specification. | | | | | | | | |
|  | |  | | | | | | | | |
| ***Clauses affected:*** | | 13.2.4.2, 13.2.4.3.1.1, 13.2.4.3.2, 13.2.4.7, 13.2.4.8 | | | | | | | | |
|  | |  | | | | | | | | |
|  | | **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:*** | |  | | | | | | | | |

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

#### 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**

\*\*\* NEXT CHANGE \*\*\*

###### 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, it shall be moved into the **dataToIntegrityProtectAndCipher** JSON object (clause 13.2.4.2), and the original value in this element shall be replaced by the index in the form {"encBlockIdx": <num>} where "num" is the index of the corresponding entry in the **dataToIntegrityProtectAndCipher** array.

\*\*\* NEXT CHANGE \*\*\*

##### 13.2.4.3.2 dataToIntegrityProtectAndCipher

The dataToIntegrityProtectAndCipher is a JSON array 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 JSON array shall contain one array entry per attribute value that needs encryption. Each array entry represents the value of the attribute to be protected, and the index in the array is used to reference the protected value within the dataToIntegrityProtect block. This associates each attribute in the dataToIntegrityProtectAndCipher block with the original attribute in the dataToIntegrityProtect block. This is needed to reassemble the original message at the receiving SEPP.

\*\*\* NEXT CHANGE \*\*\*

#### 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 refers to the NF API data-type placement mapping table to re-construct the original reformatted message by updating corresponding entries in clearTextEncapsulatedMessage with values in the dataToIntegrityProtectAndCipher array.

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.

\*\*\* NEXT CHANGE \*\*\*

#### 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 each attribute that require end-to-end encryption between the two SEPPs, the attribute value is copied into a dataToIntegrityProtectAndCipher JSON object and the attribute's value in the clearTextEncapsulatedMessage is replaced by the index of attribute value in the dataToIntegrityProtectAndCipher block.

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 replacing the references to the dataToIntegrityProtectAndCipher block with the referenced decrypted values from the dataToIntegrityProtectAndCipher block.

- 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.

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