ARES Ref. grow.ddg3.j.1(2018)1670062

Galileo Navigation Message Authentication Specification for Signal-In-Space Testing – v1.1

	Name	Date	Signature
Prepared by	I. Fernández, V. Rijmen, T. Ashur, J. Simón, C. Sarto, D. Calle, S. Cancela, P. Walker, G. Seco, D. Burkey, O. Pozzobon.		
Checked by			
Approved by			
Authorised by			

Contents

CHANGE RECORD

V1.0, 11/11/2016	First version.
V1.1, 18/06/2018	Update implementing following changes: - Modifications from v1.0 corrigendum - Typographic corrections, additional clarifications, and nomenclature unification. - Bit and byte ordering criteria (section 2.1) added to explain the field interpretation. - Chain Status field replaced by Chain and Public Key Status (CPKS) field section (3.1.3). - DSM-KROOT fields modified (section 3.3). Fields added: MAC Lookup Table (MACLT) and Rsvd. MACK Offset extended to 2 bits. KROOT DOW replaced by KROOT TOWH. DS and KROOT inverted. - MACK field nomenclature clarified (MACK section refers to the 480 bits per subframe, and MACK *block* refers to a block of MACs and one key. - KS values and MS values updated (sections 3.3.7 and 3.3.8). - Addition of MACLT and Rsvd (sections 3.3.9 and 3.3.10). - Addition of DSM-PKR and Merkle tree message and parameters (section 3.4), and cryptographic operations (sections 5.2 and 5.3). - Modification and clarification of GST_{SF} formula (section 5.4). - Addition of MAC0 and MACSEQ fields (sections 4, 5.5 and 5.6). - Modification of the PRN interpretation (table 19). - Update of the ADKD/IOD interpretation (tables 20-22), including addition of MAC for Galileo SAR RLM (ADKD=7). - Further explanation on DSM transmission and reception (sections 6.2 and 6.13) - Update of the renewal and revocation processes in accordance with the CPKS field. - MACK offsetting PRN allocation modified (section 6.9) - Annexes B (bitmask definitions and test vectors) and C (MAC Lookup Table) added.

1 INTRODUCTION AND PURPOSE

The Galileo programme will provide cryptographic data with the purpose of authenticating its Open Service navigation messages [1] [2]. This document provides the bit-level specification of the Galileo Open Service Navigation Message Authentication (OSNMA), to be used for Signal-In-Space (SIS) testing. It is organised as follows:

- Section 2 provides the overall OSNMA message structure, which is composed of two main sections: the Headers and Root Key section (HKROOT), and the MACs and Key section (MACK).
- Section 3 provides the HKROOT section structure and fields.
- Section 4 provides the MACK section structure and fields.
- Section 5 defines the cryptographic operations of OSNMA.
- Section 6 presents some general, non-exhaustive receiver guidelines for OSNMA implementation.
- Section 7 presents some parameters that need to be configurable for the SIS testing.

2 OSNMA MESSAGE STRUCTURE

The proposed scheme is based on the TESLA protocol [3], tailored for GNSS. The TESLA protocol uses message authentication codes generated with a key that is broadcast with some delay. This key is part of a pre-generated one-way chain whose root is public, known in advance by the user, and which is transmitted in reverse order with respect to its generation. The root key is authenticated by a digital signature (ECDSA) [4], and the digital signature public key can be renewed by a Merkle tree [5]. The protocol is fitted in the Galileo I/NAV navigation message transmitted in the E1-B Galileo Open Service signal. It is optimized for GNSS by using the same key chain from all satellites, and by authenticating data transmitted by other satellites from a given satellite, or *cross-authentication* [6]. It uses the E1-B I/NAV field highlighted in Figure 1 and called "OSNMA". This field is called "Reserved 1" in the OS SIS ICD I/NAV nominal pages [7], or EDBS (External Data Broadcast Service) field in other Galileo documentation. Figure 1 also presents the two sections that compose the OSNMA message: the HKROOT section (first 8 bits) and the MACK section (next 32 bits). The OSNMA field is transmitted 15 times every E1-B I/NAV 30-second sub-frame.

OSNMA is transmitted only from a subset of satellites, which currently is foreseen to be a maximum of 20, out of the total constellation. The remaining satellites will fill the OSNMA field with zeroes and their data will be *cross-authenticated* by the satellites transmitting OSNMA. OSNMA is not provided in I/NAV Dummy Messages or in I/NAV Alert Pages [7].

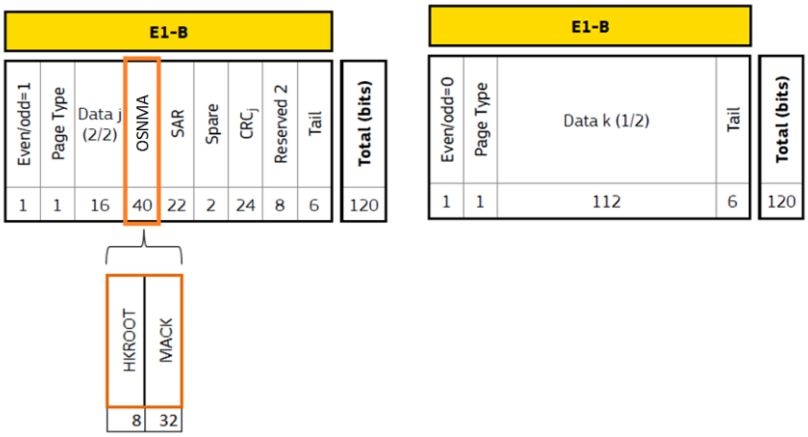

Figure 1 – OSNMA field in each I/NAV Word

The HKROOT section includes the global headers and the Digital Signature Message (DSM), usually signing a root key (KROOT). The MACK section contains the MACs and associated keys. They are described in the following sections.

2.1 Bit and Byte Ordering Criteria

All data values are encoded using the following bit and byte ordering criteria:

- For numbering, the most significant bit/byte is numbered as bit/byte 0.
- For bit/byte ordering, the most significant bit/byte is transmitted first.
- Except when noted, all fields are represented as unsigned integer as per the Galileo OS SIS ICD [7].

3 HKROOT SECTION

The HKROOT section occupies 120 bits per I/NAV sub-frame and contains three fields: the NMA Header (8 bits), the DSM Header (8 bits) and a DSM block (104 bits). The message signed in the DSM is usually a TESLA root key, or KROOT, but it can also be a new OSNMA public key. Figure 2 shows the distribution of the HKROOT fields in a nominal I/NAV sub-frame including 15 OSNMA fields. A DSM is composed of multiple blocks, and therefore is transmitted through various sub-frames. Its total length is variable, depending on the message and the cryptographic parameters used. Satellites can transmit different blocks of the same DSM at a given sub-frame.

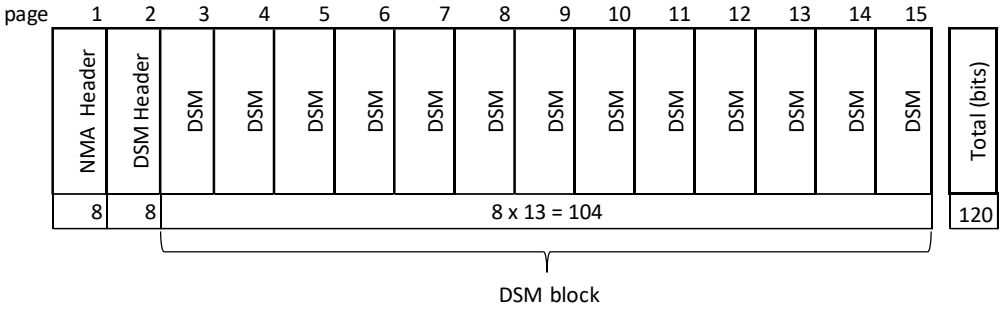

Figure 2- NMA Header and DSM fields as transmitted in the HKROOT in 15 odd page parts of one I/NAV 30-second sub-frame

3.1 NMA Header

The NMA Header defines the status of the NMA service. Its fields are described below.

NMA Status	Chain ID	Chain & PK status	Reserved	Total (bits)
2	2	3	1	8

Table 1 – NMA Header

3.1.1 NMA Status

This field presents the overall status of the NMA service, according to the values in the following table.

0	1	2	3
N/A	Test	Operational	Don't Use

Table 2 –NMA Status values

The modes are described as follows:

- "N/A": this value is not applicable.
- "Test": The NMA service is provided without any operational guarantees. The receiver may use authenticated navigation data at its own risk.
- "Operational": The NMA service is provided according to the specifications.
- "Don't Use": The receiver must not navigate using OSNMA data.

3.1.2 Chain ID (CID)

This 2-bit field represents the ID of the key chain in force. Supported values are 0 to 3, after which the CID rolls over.

3.1.3 Chain and Public Key Status (CPKS)

This field presents the status of the chain, according to the values in the following table. For further details, see section 6.4 for renewal and revocation of chains and section 6.5 for renewal and revocation of public keys.

0	1	2	3	4	5	6	7
Reserved	Nominal	EOC	CREV	NPK	PKREV	Reserved	Reserved

Table 3 –Chain and Public Key Status values

- "Reserved": This value is reserved for future use.
- "Nominal": The status of the chain in force (identified by CID) and the public key in force (identified by PKID in DSM-KROOT) is nominal.
- "EOC" (End Of Chain): The current chain is coming to an end. A DSM-KROOT with the root key of the next chain is regularly transmitted.
- "CREV" (Chain Revoked): A chain is or has been revoked.
  - If NMA Status = "Don't Use", the *current* chain, associated to the transmitted CID, is revoked.
  - If NMA Status = "Operational", a *previous* chain, associated to a previous CID, has been revoked.
- "NPK" (New Public Key): The public key in force is being renewed. A DSM-PKR with a new public key is transmitted.
- "PKREV" (Public Key Revoked): A public key is or has been revoked.
  - If NMA Status = "Don't Use", the *current* public key, identified by PKID in DSM-KROOT, is revoked.
  - If NMA Status = "Operational", a *past* public key, not anymore in force, has been revoked.

3.1.4 Reserved

This bit is reserved for future use.

3.2 DSM Header

This header informs about the DSM and block being transmitted. Its fields are described below.

DSM ID	DSM Block ID	Total (bits)
4	4	8

Table 4 – DSM Header

3.2.1 DSM ID

DSM ID is a 4-bit identifier of the DSM. As a DSM is transmitted in several blocks (1 block per I/NAV sub-frame), DSM ID identifies the DSM associated to the current block. There are two types of DSM:

- DSM-KROOT: A DSM that provides a digitally signed KROOT from a TESLA chain. This is the most frequently transmitted DSM type.
- DSM-PKR (Public Key Renewal): A DSM for public key renewal. This type of DSM will be transmitted when the public key is renewed or revoked.

DSM ID values 0 to 11 are allocated to DSM-KROOT, while DSM ID values 12 to 15 are allocated to DSM-PKR, as shown in the below table.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
DSM-KROOT IDs												DSM-PKR IDs			

Table 5 – DSM ID values

3.2.2 DSM Block ID (BID)

This field represents the ID of the DSM block sent by the transmitting satellite in the current sub-frame. It identifies the position of the block in the sequence: the first block is identified by BID = 0, the second block by BID = 1, etc., up to a maximum of 16 blocks, where the last block is BID = 15. In order to decode a full DSM, the receiver must be able to receive all the blocks and sort them in sequential order. The total number of blocks of a DSM is defined within Block 0 of the DSM, as described later, both for DSM-KROOT and DSM-PKR.

0	1	…	15
1st Block	2nd Block	…	16th Block

Table 6 – Block ID values

3.3 DSM-KROOT

A DSM-KROOT authenticates a root key of the chain in force, or that of the next chain, with the public key in force. It also defines and authenticates the chain cryptographic functions, the key and MAC sizes, and other parameters which are fixed for each given chain. It is the DSM message type transmitted more often. The DSM-KROOT structure for a digitally signed KROOT is shown in Table 7 and described below.

Nb. of Blocks	Public Key ID	Chain ID KROOT	Nb. of MACK blocks	Hash Function	MAC Function	Key Size	MAC Size	MAC Lookup Table	Rsvd	MACK Offset	KROOT WN	KROOT TOWH	α	KROOT	Digital Signature	Padding (P1)	Total (bits)
4	4	2	2	2	2	4	4	8	2	2	12	8	48	v	v	v	v

Table 7 –DSM-KROOT fields. 'v' stands for variable bit size.

3.3.1 Nb. of Blocks (NB)

This field identifies the number of blocks of the DSM. A DSM block corresponds to the 104 bits of DSM that are transmitted in a given I/NAV sub-frame, as per Figure 2 above. The 4-bit NB value does not express directly the number of blocks and needs to be converted as per the table below.

NB value	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Nb. of Blocks	rsvd	7	8	9	10	11	12	13	14	15	16	rsvd	rsvd	rsvd	rsvd	rsvd
DSM length (bits)	n/a	728	832	936	1040	1144	1248	1352	1456	1560	1664	n/a	n/a	n/a	n/a	n/a

Table 8 – Nb. Of Blocks field, including DSM-KROOT total length (bits)

3.3.2 Public Key ID (PKID)

This field represents the ID of the public key used to verify the signature of the DSM-KROOT. It is assumed that the receiver already has a trusted public key and signature algorithm associated to the Public Key ID in its possession, either because it is installed from factory, or because it has obtained it from either a DSM-PKR message or the OSNMA Server of the European GNSS Service Centre (GSC) [1]. For the time being, and according to current standards [4], ECDSA with different key lengths is considered for the DSM generation. Future revisions may consider additional signature algorithms. The cryptographic operations to be performed by the receiver are described in Section 5.

The trusted public key, together with its IDs and signature algorithm, will be published by the OSNMA provider in the OSNMA Server, together with any information required to process the DSM-PKR message. Note that only one PK is in force at a certain time, which corresponds to the PK with the highest PKID of the PKs already published, and is the PK used in the DSM-KROOT. Public Keys will be published during service provision with increasing PKID values. See section 3.4 for more details.

[1] https://www.gsc-europa.eu/ . The OSNMA GSC Server will provide OSNMA assistance through HTTPS, in a format TBD at the moment, for the parameters required in this testing specification. At the time of SIS testing, the GSC website will provide the required guidance for OSNMA receiver manufacturers.

3.3.3 Chain ID of KROOT (CIDKR)

This field identifies the chain to which the signed KROOT belongs. Notice that *CIDKR* may not be the same as the *Chain ID* (*CID*) in the NMA Header, for example when a chain renewal process takes place (see section 6.4).

3.3.4 Nb. of MACK blocks (NMACK)

This field identifies the number of MACK blocks within a sub-frame. The table below shows the number of MACK blocks and size associated to each NMACK value.

NMACK field value	0	1	2	3
Number of MACK Blocks per sub-frame	rsvd	1	2	3
MACK Block length (bits)	n/a	480	240	160

Table 9 – Nb. of MACK blocks per sub-frame, and length

3.3.5 Hash Function (HF)

This field identifies the hash function used for the chain. It is interpreted as follows:

HF field value	0	1	2	3
Hash Function	SHA-256	SHA3-224	SHA3-256	rsvd

Table 10 – Hash Function field

SHA-2 family hashes (SHA-256) are defined in the latest FIPS publication [8]. SHA-3 family is implemented according to the Keccak algorithm [9].

3.3.6 MAC Function (MF)

This field identifies the MAC function used to authenticate the navigation data with the key disclosed later. It is interpreted as follows:

HF field value	0	1	2	3
Hash Function	HMAC-SHA-256	CMAC-AES	rsvd	rsvd

Table 11 – MAC Function field

HMAC-SHA-256 is standardized in [10] and CMAC-AES is standardized as Algorithm 5 in [11]. In addition [12] and [13] can be used as references for HMAC and CMAC implementation, respectively.

3.3.7 Key Size (KS)

This field identifies the size of the keys of the chain. It is interpreted as follows:

KS field value	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Key Size (bits)	96	104	112	120	128	160	192	224	256	rsvd	rsvd	rsvd	rsvd	rsvd	rsvd	rsvd

Table 12 – Key Size field (size expressed in bits)

3.3.8 MAC Size (MS)

This field identifies the degree of truncation of the MACs, according to the following definition:

MS field value	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
MAC Size (bits)	10	12	14	16	18	20	24	28	32	40	rsvd	rsvd	rsvd	rsvd	rsvd	rsvd

Table 13 – MAC Size field (size expressed in bits)

3.3.9 MAC Lookup Table (MACLT)

This 8-bit field corresponds to the entry of a lookup table that specifies the ADKD type sequence for the MACs provided within the MACK section. The lookup table can specify a sequence expanding for 1 or 2 MACK sections. When sequence for 2 MACK sections is applicable, the start of the sequence will match with the MACK section transmitted in the first 30 seconds of a minute, in GST.

The lookup table can identify up to 256 sequences. Each sequence specifies the positions that are *fixed*, and the associated ADKD type, and the positions whose ADKD type is *flexible*, and will be dynamically allocated on every MACK section and authenticated through MACSEQ field (see 5.6). The MACLT field is constant while a TESLA chain is in force.

Annex C provides the MAC Lookup Table bit interpretation and define two MAC sequences. Other sequences within the Table will be defined during the OSNMA experimentation phase.

3.3.10 Rsvd

Field reserved for future use.

3.3.11 MACK Offset (MO)

This field defines the time offset between MACK sections for different satellite groups.

MACK Offset field value	0	1	2	3
Interpretation	No Offset	Offset	rsvd	rsvd

Table 14 – Offsetting field

When set to '0', all MACK sections are transmitted in parallel by all transmitting satellites, without any time offsetting. When set to '1', some MACK sections are transmitted with a time delay as per section 6.9.

3.3.12 KROOT WN

This parameter provides the Week Number of the time associated to KROOT (T-KROOT) referred to Galileo System Time (GST). WN is represented in 12 bits as per the Galileo SIS ICD [7].

3.3.13 KROOT TOWH

8-bit field with the time of week, in hours, associated to KROOT. WN and TOWH are relative to the GST start epoch which is 0h UTC on Sunday, 22 August 1999 (midnight between 21 and 22 August), as per [7]. The combination of KROOT WN and TOWH gives an unambiguous time reference associated to KROOT in the following way: KROOT is the key immediately preceding the first key of the chain whose application time corresponds to the sub-frame starting at WN, TOWH. The KROOT is therefore derivable from the first key applicable at this time in just one step. KROOT WN and TOWH compose the time associated to KROOT, or KROOT Time, with a one-hour resolution.

3.3.14 α

This field includes the random pattern to be included in the hashing process of the chain, as per section 5.4.

3.3.15 KROOT

Root key associated to KROOT Time, and signed, together with the chain information, in the DSM-KROOT. Its size is defined by the Key Size (KS) field. Notice that several KROOTs of the same chain associated to different times, or *floating KROOTs*, can be transmitted by the system during the time the chain is in force, to facilitate the authentication of a TESLA key.

3.3.16 Digital Signature (DS)

This field includes the digital signature of the DSM-KROOT, as per section 5.1. The DS verification is performed according to the digital signature function associated to the key identified by PKID and known by the receiver.

3.3.17 Padding (P1)

Padding bits added to the DSM in order to fit to a total length multiple of one DSM block. The padding, including its size, is detailed in section 5.1.

3.4 DSM-PKR

This section presents the DSM structure for the provision and verification of public keys for DSM-KROOT authentication. Public keys are verified through a Merkle tree [5], where the leaves of the tree are obtained by the concatenation of the public key type, ID and the key, all provided through the DSM-PKR. See section 5.3 for details on the Merkle tree. In order to validate new public keys provided through the DSM-PKR message, the user shall be in the possession of the root of the tree. This information can be loaded in the receiver from factory or retrieved from the OSNMA Server. The hashing algorithm used to generate the nodes of the tree is SHA-256 as defined in [8].

DSM-PKR supports 16 key updates, which include new public keys or emergency service messages. The structure of the DSM-PKR message is as follows:

Nb. of Blocks	Message ID	Intermediate Tree Nodes	New Public Key type	New Public Key ID	New Public Key	Padding (P2)	Total (bits)
4	4	1024	4	4	v	v	v

Table 15 - DSM-PKR fields. 'v' stands for variable bit size.

DSM-PKR fields are described below.

3.4.1 Nb. of Blocks (NB)

This field is described in section 3.3.1.

3.4.2 Message ID (MID)

This field identifies which leaf of the Merkle tree is provided, as per Table 16, and the nodes transmitted in the Intermediate Tree Nodes field.

Message ID value	Merkle tree leaves	Intermediate Tree Nodes			
0	m_0	$x_{0,1}$	$x_{1,1}$	$x_{2,1}$	$x_{3,1}$
1	m_1	$x_{0,0}$	$x_{1,1}$	$x_{2,1}$	$x_{3,1}$
2	m_2	$x_{0,3}$	$x_{1,0}$	$x_{2,1}$	$x_{3,1}$
3	m_3	$x_{0,2}$	$x_{1,0}$	$x_{2,1}$	$x_{3,1}$
4	m_4	$x_{0,5}$	$x_{1,3}$	$x_{2,0}$	$x_{3,1}$
5	m_5	$x_{0,4}$	$x_{1,3}$	$x_{2,0}$	$x_{3,1}$
6	m_6	$x_{0,7}$	$x_{1,2}$	$x_{2,0}$	$x_{3,1}$
7	m_7	$x_{0,6}$	$x_{1,2}$	$x_{2,0}$	$x_{3,1}$
8	m_8	$x_{0,9}$	$x_{1,5}$	$x_{2,3}$	$x_{3,0}$
9	m_9	$x_{0,8}$	$x_{1,5}$	$x_{2,3}$	$x_{3,0}$
10	m_{10}	$x_{0,11}$	$x_{1,4}$	$x_{2,3}$	$x_{3,0}$
11	m_{11}	$x_{0,10}$	$x_{1,4}$	$x_{2,3}$	$x_{3,0}$
12	m_{12}	$x_{0,13}$	$x_{1,7}$	$x_{2,2}$	$x_{3,0}$
13	m_{13}	$x_{0,12}$	$x_{1,7}$	$x_{2,2}$	$x_{3,0}$
14	m_{14}	$x_{0,15}$	$x_{1,6}$	$x_{2,2}$	$x_{3,0}$
15	m_{15}	$x_{0,14}$	$x_{1,6}$	$x_{2,2}$	$x_{3,0}$

Table 16 - Message ID field and associated Merkle tree leaves and intermediate tree nodes

3.4.3 Intermediate Tree Nodes (ITN)

This field provides the four Merkle tree nodes necessary to authenticate the message identified by the Message ID. Each node is 256 bits long, for a total field size of 1024 bits. The nodes are sent as per the order defined in Table 16. E.g., for leaf m_0, ITN = ($x_{0,1}$ || $x_{1,1}$ || $x_{2,1}$ || $x_{3,1}$), where (X || Y) concatenates X and Y.

3.4.4 New Public Key type (NPKT)

This field represents the signature algorithm associated to the public key provided in the DSM-PKR message, as per the following table. Value 4 indicates that an Emergency Service Message is transmitted. If this is the case, the receiver is requested to discontinue the NMA service and connect to the OSNMA Server for further updates.

PK type value	0	1	2	3	4	5-16
Type of message	ECDSA P-224	ECDSA P-256	ECDSA P-384	ECDSA P-521	Emergency Service Message	rsvd

Table 17 - Public key type field

3.4.5 New Public Key ID (NPKID)

This field represents the ID of the new public key. If NPKT = 4 (Emergency Service Message), NPKID is set to 0.

3.4.6 New Public Key (NPK)

This field provides the new public OSNMA key. Keys are provided as compressed ECDSA keys, including sign field and rounded up to a whole number of bytes, as per [4]. Length will depend on the New PK type. For ECDSA P-224, ECDSA P-256, ECDSA P-384 and ECDSA P-521 length of the NPK will be 232, 264, 392 and 536 bits respectively.

In the case an emergency message is provided (field New Public Key Type set to 4) this field will contain a random sequence of bits allowing the authentication of the message. The NPK length will then be such that the whole DSM-PKR message fits in the number of blocks given by the NB field:

$$L = (104 \cdot NB) - 1040$$

Where *L* is the length of the NPK field when emergency message is provided, *NB* is the number of blocks of the DSM-PKR message as per 3.4.1, and 1040 is the size in bits of all the other fields (NB, MID, ITN, NPKT, and NPKID).

3.4.7 Padding (P2)

This field includes padding bits added to the DSM, when required, in order to fit to a total length multiple of one DSM block, as per in section 5.2.

4 MACK SECTION

The MACK section is transmitted in parallel to the HKROOT and contains the truncated MACs and time-delayed keys that authenticate the navigation data. It uses 480 bits per I/NAV 30-second sub-frame, out of total of 600 bits reserved for OSNMA. It is divided MACK blocks. Each MACK block is composed of several MACs, or tags, each followed by a 'MAC-Info' field, providing information about the data authenticated, and a TESLA key. The number of MACK blocks in an I/NAV subframe is fixed for each chain, as described in section 3.3.4.

Figure 3 summarises the structure of the MACK section. The different fields are grouped by type: all MAC and MAC-Info sections of a given MACK block are put in a column, the key is put in the next column, and so on. The order in which the information is received is given by reading each row from left to right, and from top to bottom. All MAC and MAC-Info fields follow the same format, except the first one in the first MACK section of the sub-frame, where the MAC (MAC0) authenticates the navigation data of the transmitting satellite, and the MAC-Info includes a field (MACSEQ) to authenticate the sequence of the flexible MACs (i.e. not fixed in MACLT, section 3.3.9) transmitted in the MACK section. The other MACs of a sub-frame can authenticate other data.

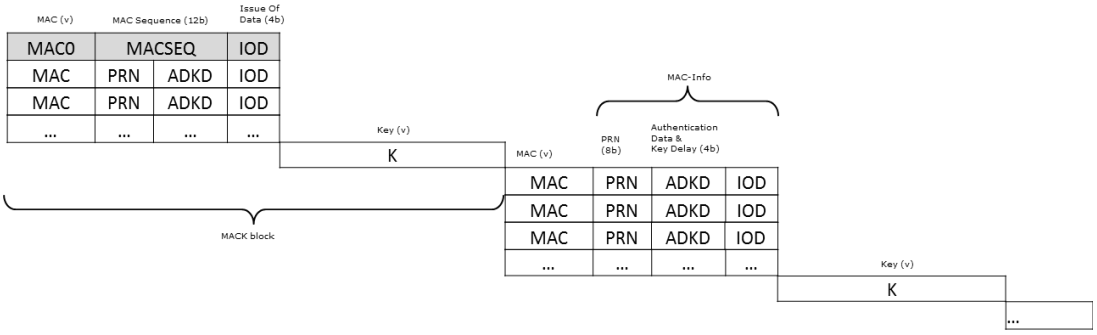

Figure 3 – MACK section structure. 'v' stands for variable bit length.

The number of MACs per MACK block is the maximum possible and can be calculated as:

$$n_M = floor\left(\frac{480/NMACK - KS}{MS + 16}\right)$$

Where n_M is the number of MACs per MACK block, and NMACK (number of MACK blocks), KS (key size) and MS (MAC size) are defined in section 3.3. *floor(A)* is the largest integer no greater than *A*. If the sum of MAC & MAC-Info sections and key does not equal the 160, 240 or 480-bit MACK block length, spare bits are set to '0' at the end of each MACK block. The MACK section fields are described in the following subsections.

4.1.1 MAC0

MAC truncated from the MSB, or tag, of length as defined in the MAC Size (MS) field of the DSM-KROOT of the chain in force, that authenticates the I/NAV words 1 to 5 (ADKD=0) of the transmitting satellite, according to section 5.5.

4.1.2 MACSEQ

MACSEQ is a 12-bit field that allows the receiver to authenticate the MAC-Info field for the MACs of the MACK section whose ADKD type is flexible ("*FLEX*" as per Table 30;

see also MACLT field, section 3.3.9). Section 5.6 describes the generation and verification of the MACSEQ field.

4.1.3 MAC

MAC truncated from the MSB, or tag, with the length defined in the MAC Size field of the DSM-KROOT of the chain in force. Section 5.5 describes the generation and verification of the tag.

4.1.4 MAC-Info

This section contains the fields PRN, Authentication Data & Key Delay (ADKD) and Issue Of Data (IOD), as shown in Table 18. They identify the data authenticated by the MAC field.

field	PRN	ADKD	IOD	*total length*
length	8	4	4	*16*

Table 18 – MAC-Info

4.1.4.1 PRN (8 bits)

PRN of the satellite transmitting the navigation data which is authenticated. By using 8 bits, up to 255 satellites could be authenticated, which allows for the authentication of satellite data from Galileo and other constellations (mainly GPS but also, GLONASS, Beidou or others) and regional systems such as SBAS. The convention used for the PRN field is provided in the table below.

PRN	Interpretation
0	Rsvd
1-36	36 Galileo PRNs as per Galileo signal specification [7], Satellite ID/SV_{ID} field[2], values 1-36.
37-63	Rsvd
64-95	32 GPS PRNs as per GPS signal specification [14], minus 63 (e.g. PRN 64 represents GPS PRN 1).
96-99	Rsvd
100-123	24 GLONASS PRNs reserved for future implementation (GLONASS satellite slot numbers as per GLONASS ICD [15], minus 99. E.g. PRN 100 represents GLONASS slot number 1).
124-135	Rsvd
136-172	37 BDS PRNs reserved for future implementation (37 ranging code numbers as per Beidou ICD [16], minus 135. E.g. PRN 136 represents Beidou ranging code number 1).
173-190	17 SBAS PRNs, reserved for future implementation (SBAS GEO PRNs, as per SBAS MOPS [17], minus 53. E.g. PRN 173 represents SBAS MOPS PRN 120).
191-251	Rsvd
252	Reserved for future implementation (general BEIDOU constellation information that does not relate specifically to a satellite)
253	Reserved for future implementation (general GLONASS constellation information that does not relate specifically to a satellite).
254	General GPS constellation information that does not relate specifically to a satellite.
255	General Galileo constellation information that does not relate specifically to a satellite.

Table 19 – PRN parameter definition

[2] SV_{ID} values 37 to 63 are not currently foreseen to be allocated.

In the case of SAR RLM authentication (ADKD=7) the PRN field corresponds to the 8 bits of the MSB-truncated 24-bit CRC[3] calculated over the concatenation of the SAR RLM Identifier bit and the 60-bit Beacon ID field as follows:

SAR PRN = trunc(8, CRC(SAR RLM Identifier || Beacon ID))

The function *trunc(L,I)* retains the *L* MSBs of the input *I*.

4.1.4.2 Authentication Data & Key Delay (4 bits)

This field describes the authenticated navigation data, as per the below table. It also describes, in the case of the *SLMAC* ("Slow MAC") values, the extra time between the MAC and the associated key disclosure, with respect to a standard MAC.

ADKD	Interpretation
0	Eph, Clk & Health
1	GPS LNAV Iono
2	Galileo Sub-frame / GPS Frame
3	Almanac
4	GST-UTC & GST-GPS
5	Other NAV Msg
6	RedCed of I/NAV improved processing
7	SAR RLM
8	Rsvd
9	Rsvd
10	Rsvd
11	SLMAC, ADKD=0 mask with 1 extra sub-frame delay (30s)
12	SLMAC, ADKD=0 mask with 10 extra sub-frames delay (5 min)
13	Rsvd
14	Rsvd
15	Rsvd

Table 20 – ADKD parameter definition

Table 21 below interprets ADKD values for Galileo and GPS, in accordance with the current Galileo infrastructure capabilities. ADKDs for other global or regional systems are left for future implementation at the moment.

[3] CRC computation as per checksum computation from Galileo OS SIS ICD [4].

ADKD	Galileo (PRN 1-36)	GPS (PRN 64-95)
'0'	Words Types 1-5 of [7]: the MAC authenticates the following I/NAV [7] data fields (concatenated in the order defined here): • Word 1: IODnav, Ephemeris 1/4; • Word 2: IODnav, Ephemeris 2/4; • Word 3: IODnav, Ephemeris 3/4, SISA (E1,E5b); • Word 4: IODnav, SVID, Ephemeris 4/4, Clock correction; • Word 5: Ionospheric correction, BGD(E1,E5a), BGD(E1,E5b), E5bHS, E1BHS , E5bDVS and E1BDVS). Reserved/Spare bits and GST are not included[4].	GPS Eph, Clk & Health: the MAC authenticates the following GPS L1 C/A [14] data fields (concatenated in the order defined here) extracted after removing the parity inversion: • Subframe 1: URA INDEX, SV HEALTH, IODC 2MSBs, TGD, IODC 8 LSBs, toc, af2, af1, af0; • Subframe 2: IODE, Crs, Δn, M0, CUC, e, CUS, $\sqrt{A}$, toe, fit interval flag, AODO; • Subframe 3: Cic, Ω0, Cis, i0, Crc, ω, $\dot{\Omega}$, IODE, IDOT.
'1'	N/A	*For future implementation.* (most recent ionospheric model information as transmitted in GPS sub-frame 4, page 18 , $\alpha_{0\text{-}3}$, $\beta_{0\text{-}3}$)
'2'	Sub-frame: the MAC authenticates the bits of the 'Page Type' and the 128-bit word data of the I/NAV nominal page of the 15 pages of the last full sub-frame in the order they are transmitted. In case of I/NAV Spare Word or Dummy Page only the 'Page Type' and the 'Word Type' fields are authenticated.	*For future implementation.* (last full GPS 30-s frame, including sub-frames 1 to 5, all words per sub-frame. Parity bits are excluded. All other bits are included)
'3'	Almanac: the MAC authenticates the almanac data from WT7, WT8, WT9 and WT10 of the last two E1B I/NAV subframes of the transmitting satellite. The MAC authenticates the following I/NAV data fields (concatenated in the order defined here): • Word 7: IODa, WNa, t_{0a}, SV_{SVID1} (1/2) • Word 8: IODa, SV_{SVID1} (2/2), SV_{SVID2} (1/2) • Word 9: IODa, WNa, t_{0a}, SV_{SVID2} (2/2), SV_{SVID3} (1/2) • Word 10: IODa, SV_{SVID3} (2/2)	*For future implementation.*
'4'	GST-UTC conversion parameters (WT6), TOW (WT6) and GST-GPS (WT10) authenticated as per the last WT6 and WT10 provided by the transmitting satellite in the E1-B I/NAV. Data is concatenated using words order: first word 6 and then word 10.	*For future implementation.*
'5'	Gal F/NAV: the MAC authenticates the F/NAV page 1 to 4, excluding GST, spare, CRC and tail bits.	*For future implementation.* (GPS CNAV)

[4] Note that the GST in WT5, which should be common to all satellites, is authenticated by authenticating any TESLA key, as a TESLA key used to authenticate the navigation must relate correctly to GST. The same occurs with GPS Time in GPS navigation authentication, which holds a constant and deterministic relationship (aside from GGTO, which can be estimated or authenticated from ADKD='4').

ADKD	Galileo (PRN 1-36)	GPS (PRN 64-95)
'6'	*For future implementation.* I/NAV RedCed: Reduced Clock and Ephemerides bits, according to I/NAV Improvements. Bit mask is TBD at the moment.	N/A
'7'	SAR. This MAC authenticates a SAR RLM previously transmitted in the E1-B I/NAV pages for the transmitting satellite. The MAC authenticates the concatenation of the 'Short/Long RLM Identifier' bit plus the 4 20-bit blocks (for short RLM) or 8 20-bit blocks (for long RLM) of 'SAR RLM Data'. The MAC authenticates a SAR RLM that started transmission (first block) in the previous I/NAV sub-frame. It is meant to be transmitted only when the RLM carries non spare information.	N/A
'11'	The receiver will get the key associated to that MAC exactly with 1 sub-frame (30 seconds) delay in addition to the delay with which it would have received it in normal conditions. MACs are generated following ADKD0 definition.	N/A
'12'	The receiver will get the key associated to that MAC exactly with 10 sub-frames (5 minutes) delay in addition to the delay with which it would have received it in normal conditions. MACs are generated following ADKD0 definition.	

Table 21 – ADKD interpretation for Galileo and GPS

4.1.4.3 Issue-Of-Data (4 bits)

Identification of the Issue Of Data of the authenticated information. It is interpreted in Table 22 as follows, for different ADKDs:

ADKD	Galileo (PRN 1-36)	GPS (PRN 64-95)
'0'	The first bit is set to '1' for the first MAC provided by the transmitting satellite after the authenticated data has changed, or set to '0' otherwise[5]. This can happen because there is a new IODnav, or because the WT5 authenticated information has changed. The WT5 authenticated corresponds to the last one transmitted in the I/NAV E1-B before the MAC (TBC). The following 3 bits are the LSB-truncated IODnav bits. Note that GST (WN, TOW) changes on every sub-frame, but this is not considered new data and thus not triggering the flag, as it is not part of the signed plain text.	The first bit is set to '1' for the first MAC provided by the transmitting satellite after the authenticated data has changed, or set to '0' otherwise[5]. The following 3 bits are the LSB-truncated IODE bits. The IODC of the authenticated clock corrections is that of the clock corrections transmitted with the last IODE.
'1'	N/A	N/A. Set to zero.
'2'	N/A. Set to zero.	N/A. Set to zero.
'3'	4 bits of the IODa for the almanac transmitted in the last E1B I/NAV subframe.	*For future implementation.*
'4'	N/A. Set to zero.	N/A. Set to zero.
'5'	The first bit is set to '1' for the first MAC provided per satellite after the authenticated data has changed, or set to '0' otherwise[5]. This can happen because there is a new IODnav, or because the ionospheric correction, GST-UTC information, GST-GPS information, or the SIS flags have changed. The following 3 bits are the LSB-truncated IODnav bits. Note that TOW changes on every page, but this is not considered new data and thus not triggering the flag.	For future implementation.
'7'	The IOD field corresponds to the SAR RLM message code.	
'11'and '12'	Same as ADKD='0'.	N/A. Not transmitted.

Table 22 – IOD interpretation as per ADKD and PRN (Gal/GPS)

4.1.5 Key

This field contains the TESLA chain key. The position in the chain depends on the satellite PRN and time as defined in section 5.4.

[5] This bit solves possible ambiguities in the identification of the data being authenticated in case of navigation data update with identical truncated IODnav/IODE to the one being previously authenticated.

5 RECEIVER CRYPTOGRAPHIC OPERATIONS

This section describes the cryptographic operations to be implemented for OSNMA. Cryptographic operations are divided in the following categories:

- DSM-KROOT, comprising the reception and verification of a root key for a TESLA chain.
- DSM-PKR, comprising the reception and verification of a new public key.
- TESLA key verification, comprising the verification of a chain key with a root key.
- MAC verification, comprising the navigation message authentication.

5.1 DSM-KROOT

The DSM-KROOT digital signature is produced by concatenating the fields described below, where the MSB is represented to the left. It is verified with the public key in the possession of the receiver and identified by the *Public Key ID* field. The message *M* to be signed is:

$$M = (NMA\_Header \,||\, CIDKR \,||\, NMACK \,||\, HF \,||\, MF \,||\, KS \,||\, MS \,||\, MACLT \,||\, Rsvd \,||\, MO \,||\, KROOT\ WN \,||\, KROOT\ TOWH \,||\, \alpha \,||\, KROOT)$$

Where

- (X || Y) concatenates bitset X to bitset Y bits, with X at the MSB
- *NMA_Header* corresponds to the 8-bit NMA Header field as per section 3.1, transmitted in parallel to the DSM-KROOT. Note that, while a given DSM ID is transmitted, the *NMA Header* is kept constant. If the *NMA Header* changes, a new DSM will be transmitted.
- The remaining fields are described in section 3.3.

Following the KROOT, the digital signature DS is transmitted.

$$DS = signature(M \,||\, P0)$$

Where *P0* is a number of padding bits (0 to 7) set to zero so that (*M* || *P0)* fits to a whole number of bytes. The algorithm for the digital signature is ECDSA [4], supporting signature lengths of 448, 512, 768, and 1056 bits, with key lengths of 224, 256, 384, and 521 bits respectively. The couples of curves and hash functions used to generate the DS are respectively as follows: P-224/SHA-224, P-256/SHA-256, P-384/SHA-384, and P-521/SHA-512. These parameters are statically and univocally associated with the *Public Key ID* in force, and known by the receiver. Once a *PKID* is in force, any lower *PKID* is considered not in use by the system and therefore DSMs with a *PKID* lower than that in use should be rejected.

The Padding field (P1) is added as follows:

$$P1 = trunc(L, hash(M \,||\, DS))$$

$$L = (104 \cdot NB) - length(M) - length(DS)$$

Where *P1* is the Padding field as per 3.3.17, and *L* is the difference in number of bits between the total block length and the length of *DS* and *M*. *trunc(n,p)* is the truncation function that retains the *n* MSBs of the input *p*; *hash* is SHA-256 [8] irrespective of the signature scheme or length; and *NB* is the *Nb. of Blocks* as defined in 3.3.1. Note that *NB* is the smallest value such that L>=0, i.e. the DSM will use the minimum number of blocks possible.

5.2 DSM-PKR

The DMS-PKR will provide the user either with a new ECDSA public key, together with the values to correctly interpret and apply the new public key, or an emergency service message, plus the information allowing their authentication. The user will authenticate the ECDSA public key or service message against a Merkle root using the hashing algorithm that was used for the tree generation (SHA-256, [8]). The user will re-generate the leaf of the tree as follows:

$$m_i = (NPKT \,||\, NPKID \,||\, NPK)$$

Where m_i is the leaf of the tree and *NPKT, NPKID* and *NPK* are the *New Public Key Type, New Public Key ID* and *New Public Key* DSM-PKR message fields as detailed in section 3.4.

The next section details the process to validate m_i against the root of the tree by means of the distributed tree nodes. The Merkle tree root and hashing algorithm can be loaded in the receiver from factory or retrieved from the OSNMA Server.

The padding field (*P2*) is added as follows:

$$P2 = trunc(L, hash(Root \,||\, m_i))$$

$$L = (104 \cdot NB) - length(M)$$

Where *P2* is the Padding field as per 3.4.7; *trunc(n,p)* is the truncation function that retains the *n* MSBs of the input *p*; *Root* is the Merkle tree root, m_i is the tree leaf, hash is SHA-256, *NB* is the Nb. of Blocks as defined by 3.4.1, and *M* is the message from Table **15** excluding the padding bits. Note that *NB* is the smallest value such that *L*>=0, i.e. the NMA message will use the minimum number of blocks to transmit a given DSM-PKR.

5.3 Public Key verification through the Merkle tree

A Merkle tree can be used to authenticate a set of $N = 2^n$ items, using only a cryptographic hash function *f*, which is defined as SHA-256 in this specification as per section 3.3.5. The *N* items need to be known at the time of generating the tree. We denote the *N* items by m_i, with $i = 0, \ldots, N - 1$ and compute the base nodes of the tree as follows:

$$x_{0,i} = f(m_i);\ i = 0, \ldots N - 1$$

The other nodes are computed as follows:

$$x_{j,i} = f\,(x_{j-1;2i} \,\|\, x_{j-1;2i+1})$$

Where $j = 1, \ldots n$ denotes the level in the tree, and for each j: $i = 0, \ldots 2^{n-j} - 1$. The value $x_{n,0}$ is the root of the tree. It needs to be distributed to all verifiers. Galileo OSNMA uses a Merkle tree of $N = 16$ ($n = 4$) for public key renewal, as depicted in Figure 4.

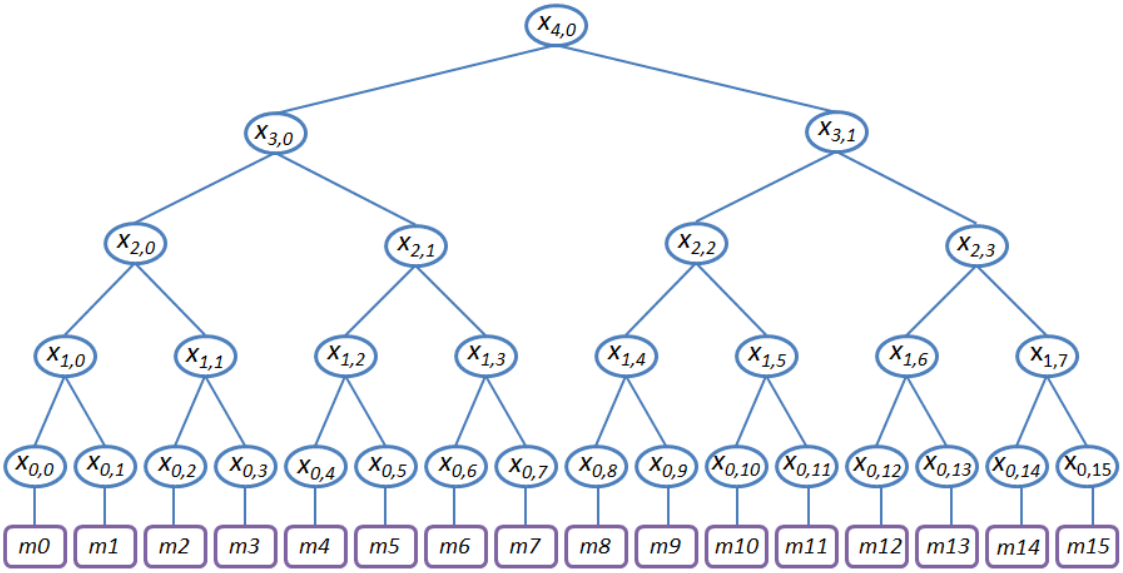

Figure 4 - Merkle-tree with $N = 16$

In order to distribute m_0, i.e. the leaf of the tree associated to $x_{0,0}$, the sender transmits m_0 together with the required information to allow the verifier to compute $x_{n,0}$. That is $x_{0,1}$, $x_{1,1}$, $x_{2,1}$, $x_{3,1}$.

In order to authenticate m_0, the verifier computes $x_{0,0} = f(m_0)$, then computes $x_{1,0} = f(x_{0,0} \| x_{0,1})$, etc. until the root $x_{n,0} = f(x_{n-1,0} \| x_{n-1,1})$ is obtained and compared to the stored one. Note that with this information no other m_i can be authenticated. Later, in order to distribute m_1, the sender transmits m_1 together with $x_{0,0}$, $x_{1,1}$, $x_{2,1}, \ldots x_{n-1,1}$. In a similar way, each of the N items can be authenticated by transmitting $n = log_2 N$ nodes of the tree.

5.4 TESLA Key generation and verification

The TESLA chain starts with a random seed key K_n, which is secret and only known by the OSNMA provider, and ends with a root key K_0 that is public and certified through the DSM-KROOT. K_n and K_0 relate as follows:

$$K_0 = F^n(K_n)$$

Where F^n means recursively applying n times the function F, so each element of the chain can be constructed by applying F to the previous element. The key K_n is a random number truncated, if necessary, by removing the LSBs to the Key Size (l_k below) as defined in 3.3.7. The function F is applied for iteration m as follows:

$$K_m = F(K_{m+1}) = trunc(l_k, hash(K_{m+1} \,||\, GST_{SF} \,||\, \alpha \,||\, P3))$$

where K_m is the key to be generated from K_{m+1}, the previous key in the chain; *trunc(n,p)* is the truncation function that retains the n MSBs of the input p; l_k is the chain key size as per 3.3.7; *hash* is the selected hash function as per the HF field in 3.3.5; α is the unpredictable chain pattern defined in 3.3.14 and *P3* is a number of padding bits (0 to 7)

set to zero so that the message input to the hash function fits to a whole number of bytes. GST_{SF} is the Galileo System Time at the start of the 30-second interval in which the sub-frame in which the key K_m will be transmitted begins transmission. For E1, this is the E1 sub-frame start time minus 1 second. For the key hashing process, GST_{SF} is a 32-bit number as per [7].

Figure 5 presents how the TESLA keys are assigned in time and transmitter and how they relate to each other.

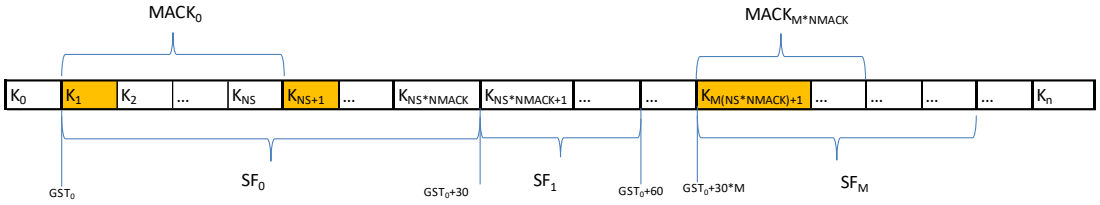

Figure 5 –TESLA one-way chain with different keys from different senders, 2 MACK blocks per I/NAV sub-frame

The key generator must produce NS·NMACK keys per sub-frame, where NS stands for maximum number of satellites with different keys per MACK block. The first key of block (e.g. K_1, K_{NS+1}, $K_{M(NS*NMACK)+1}$) is used to generate all the MACs transmitted in that block, except for the SLMAC case, and the case when MACK Offset = 1, detailed in section 6.9. It will also be transmitted by PRN 1, while the following (K_2, K_{NS+2}, $K_{M(NS*NMACK)+2}$…) by PRN2, and so on. For testing purposes NS should be configurable between 1 and 36, as per section 7. Due to the fact that all keys belong to the same chain a missed key broadcasted by a satellite can be recomputed by the hashing process of any subsequent key from the chain delivered by the same or other satellite. This allows, for example, the receiver to use the OSNMA scheme with the reception of keys from a single satellite.

The value of the *GST* to be used in *F( )* can be computed as follows:

$$GST_{SF} = GST_0 + 30 \cdot floor(\frac{m-1}{NS \cdot NMACK})$$

Where GST_0 is the start time of the chain as defined by KROOT TOWH and KROOT WN from DSM-KROOT, *m* is the chain link position (e.g. K_1 in Figure **5** has *m*=1), *NS* is the number of keys per MACK block, *NMACK* is the total number of MACK blocks per sub-frame, and *floor(A)* is the largest integer no greater than *A*. Note that the key index decrements from the seed value down to 1, which is the first key that will be used. To provide a trusted root for the entire chain, the key chain must be continued to K_0, as this is the key that is signed when a new chain is disseminated. Note that for evaluating K_0, $GST_{SF} = GST_0 - 30$ seconds, where GST_0 is the start time associated with the chain.

Note also that when a new KROOT is disseminated for a chain renewal, and therefore prior to the entry into force of its associated chain at GST_0, the KROOT is nominally the last key from the preceding time-slice ($GST_0 - 30$), and it will never be used for MAC production as it relates to a time before the chain enters into force. For simplicity, all intermediate roots that are signed and transmitted via a DSM-KROOT (after their applicability has passed) will conform to the same model, so the time of applicability associated with the intermediate key will be the time of applicability of the first key after the one that is sent. This means that a receiver can use the exact same algorithm to validate

a key against a signature irrespective of whether it is a root key of a new chain or a "floating root key" of the current chain.

An event of TESLA key verification failure might indicate a CID rollover missed by the receiver in case, for example, of being switched off for prolonged time. Receivers shall regularly check the DSM for confirmation of the applicable KROOT and associated CID.

5.5 MAC verification

MACs are generated and verified as explained in this section. For MACs different than MAC0:

$$m = (PRN\_N \,||\, PRN\_A \,||\, GST_{SF} \,||\, CTR \,||\, NMA\_Status \,||\, navdata \,||\, P3)$$

$$tag = trunc(MS, mac(K,m))$$

For MAC0:

$$m_0 = (PRN\_A \,||\, GST_{SF} \,||\, CTR \,||\, NMA\_Status \,||\, navdata \,||\, P3)$$

$$tag_0 = trunc(MS, mac(K,m_0))$$

Where

- *m* is the message to be authenticated (m_0 in the case of MAC0).
- *PRN_N* is the satellite ID whose data is *being authenticated,* as defined in the MAC-info section, PRN field.
- *PRN_A* is the satellite ID *providing authentication* (it can only be a Galileo satellite), as per the Galileo OS SIS ICD [7].
- GST_{SF} is the Galileo system time (seconds), coded in a 32-bit field as per [7], of the start of the thirty second cycle in which the sub-frame starts in which the MAC is due to be transmitted, not accounting for offsetting delays. For transmission in E1 this is the sub-frame start time minus 1 second.
- *CTR* is an 8-bit unsigned integer identifying position of the MAC within a single MACK block; it has a value of '1' for the first MAC (or MAC0) at the MSB, incrementing by one for each subsequent MAC towards the LSB of that block.
- *NMA_Status* is the 2-bit *NMA Status* field in the NMA Header transmitted in the current sub-frame.
- *navdata* is the navigation data authenticated as defined in the MAC-Info section, ADKD field.
- *P3* is a number of padding bits (0 to 7) set to zero so that the message input to the *mac* function fits to a whole number of bytes.
- *tag* is the truncated message authentication code transmitted in the signal (tag_0, in the case of MAC0).
- *trunc(n,p)* is the truncation function that retains the *n* MSBs of the input *p*.
- *MS* is the truncated MAC length for the chain in force, as defined in the DSM.

- *mac* is the MAC function used for the chain in force, as defined in the DSM-KROOT (MF).
- *K* is the key from the one-way chain used for the MAC generation. Note that *K* may not be the key transmitted in the MACK block to which the MAC belongs, which may need to be hashed through the chain as explained in section 5.4, and 6.9 in case of MAC offsetting.

Note that the MAC input parameters are unique for each MAC. Note also that the GST is verified by ensuring the authenticity of a key vs the KROOT: if a key of a certain sub-frame is authenticated using the TOW of that sub-frame, the TOW must be correct too as otherwise the key verification process would not lead to the KROOT.

In order to perform the message authentication, a tag has to be stored and compared with the one re-generated according to the MAC-Info section once a valid key is received and the data is available.

5.6 MACSEQ verification

The 12-bit MACSEQ field provides a MAC for the concatenation of the MAC-Info fields for the MACs of the MACK section whose ADKD type is flexible and to be dynamically allocated as per applicable MAC sequence (see MACLT field, section 3.3.9), so the receiver can verify their authenticity. The MACSEQ is generated with the same key and MAC function as the rest of the MACs in the same MACK block.

$m = (PRN\_A \,||\, GST_{SF} \,||\, MFLEX_1 \,||\, MFLEX_2 \,||\, \ldots \,||\, MFLEX_N)$

$$MACSEQ = trunc(12, mac(K,m))$$

Where

- *m* is the message to be authenticated.
- *PRN_A* is defined as per section 5.5.
- GST_{SF} is defined as per section 5.5.
- $MFLEX_i$ are the authenticated MAC-Info fields. $MFLEX_1$ represents the MAC-Info field for the first MAC in the current MACK section with flexible ADKD type. $MFLEX_N$ represents the MAC-Info field for the last MAC in the current MACK section with flexible ADKD type. If there are no flexible ADKDs in the chain, *m* will include only $(PRN\_A \,||\, GST_{SF})$.
- *trunc(n,p)* is the truncation function that retains the *n* MSBs of the input *p*. In the case of MACSEQ truncation is done to 12 bits.
- *mac* is the MAC function used for the chain in force, as defined in the DSM-KROOT (MF).
- *K* is the key from the one-way chain used for the MAC generation.

6 GUIDELINES FOR IMPLEMENTATION

This section provides some non-exhaustive guidelines for OSNMA implementation and testing, including some details on how the OSNMA information is transmitted, in order to complement the previous sections.

6.1 Processing the OSNMA field

The OSNMA 40-bit field is filled in only for satellites connected to the Galileo ground segment. The maximum current system capacity for ground connection at Full Operational Capability is 20 links, so out of the 24 satellites nominally composing the Galileo constellation, only a maximum of 20 will be transmitting OSNMA simultaneously. As the uplink antennas switch between satellites, the transmission of OSNMA information may be interrupted at any time. When a satellite is not connected to ground and therefore not transmitting OSNMA, the OSNMA field will be filled with zeros.

6.2 DSM transmission and reception

This section provides some guidelines for the reception and processing of the DSM:

- The transmission of blocks of the same DSM ID is scattered across different satellites. For optimum reception, the receiver must be able to store DSM blocks from different satellites, until a full DSM is received. Every satellite transmits all the blocks of a DSM. This implies that receiver can also reconstruct the whole DSM from either a single satellite or a subset of satellites.
- Two DSMs can be transmitted alternately, e.g. when a new chain is about to enter into force. Therefore, the receiver must be able to store and build more than one DSM in parallel. At a given sub-frame, all satellites transmit blocks belonging to the same DSM ID.
- A DSM ID shall always be transmitted in its entirety, before switching to another DSM ID. This means that satellites with DSM offsetting = 0, as per section 6.13, will transmit the full DSM from BID=0. Therefore, in degraded conditions, decoding a single DSM ID/Block ID byte allows determining the Block ID of the remaining satellites.
- If the transmission of a DSM is interrupted, the receiver must delete DSM blocks associated to incomplete DSM IDs. A maximum of 1 hour before deletion is recommended.
- During the transmission of a DSM-KROOT, the NMA Header remains constant, and it coincides with the NMA Header information authenticated by the DSM-KROOT.
- DSM-PKR, when transmitted, is alternated with DSM-KROOT. Both DSM-PKR and DSM-KROOT are transmitted entirely, before switching to the other DSM type.

6.3 Use of floating KROOTs

During the time a chain is in force, the distance between the initial root key K_0 and the transmitted keys K_m can increase to a point where the CPU consumption in the receiver may be too high. The receiver can replace the root key K_0 by an already authenticated key K_m in the chain from the MACK section, and use K_m to authenticate the subsequent keys. DSM-KROOTs associated to more recent keys than the initial K_0, or *floating KROOTs*, will be regularly transmitted in order to facilitate the key authentication process for receivers that switch on in the middle of a TESLA chain.

6.4 Chain renewal and revocation

This section presents the TESLA key chain renewal and revocation processes. The chain renewal process is the usual process to follow when a TESLA chain is coming to an end.

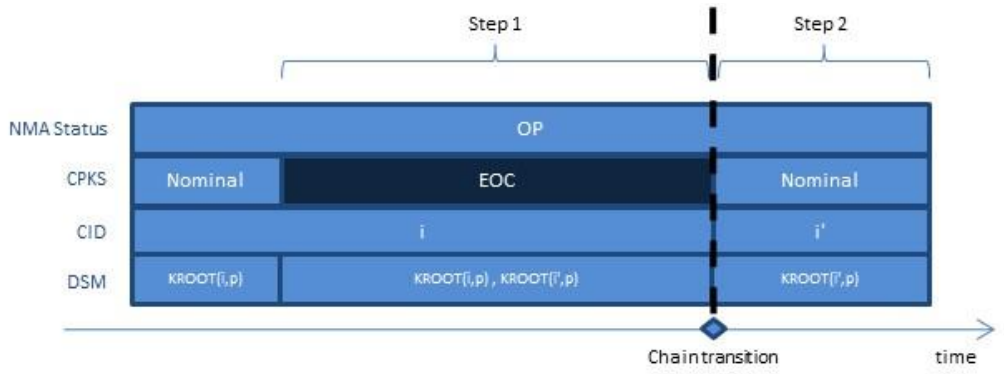

Figure 6 –Chain renewal

The chain renewal process is depicted in Figure 6 and comprised by the following steps:

- **Step 1**: The CPKS (Chain and Public Key Status) flag is set to EOC (End of Chain), reporting that the chain in force *i* is coming to an end. A new DSM-KROOT for the new chain *i'* is transmitted (note that, in general, CIDs will follow a sequential order *mod* 4). During this step, the DSM alternates two DSM-KROOTs: one for the chain *i* currently in force, KROOT(*i,p*), where *p* is the public key in force, and one for the next chain, KROOT(*i',p*). The parameters KROOT WN/TOWH set the start time of the new chain.
- **Step 2**: At the transition time set by KROOT WN/TOWH, the new chain *i'* becomes in force. CPKS is set to Nominal, CID is set to *i'*, and the DSM transmits only a KROOT for the new chain KROOT(*i',p*). The previous chain *i* is considered as expired. Receiver shall discard any KROOT related to the previous chain.

The chain revocation process is followed only when the chain in force is revoked, e.g. because it has been compromised, and therefore it is not expected in nominal operation.

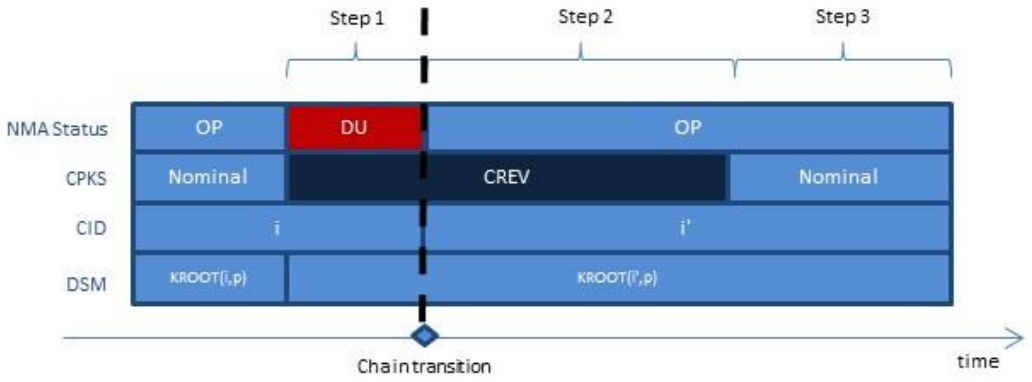

Figure 7 –Chain revocation

The chain revocation process is depicted in Figure 7 and comprised by the following steps:

- **Step 1**: The OSNMA Status flag is set to "Don't Use" and the CPKS flag is set to CREV, reporting that the chain in force *i* is revoked. A new DSM-KROOT(*i',p*), with

the root key of a new chain *i'* is transmitted. Its KROOT WN/TOWH fields may refer to a time in the past, even though the chain is not yet in force because CID has not yet been updated. The receiver may store the new KROOT but it must wait until the new chain is set in force in Step 2. Receiver shall discard any KROOT related to the previous chain.

- **Step 2**: The CID field is set to *i'*, reporting that the new chain is in force. The OSNMA Status is set back to "Operational". The CPKS is maintained to CREV, to report (in combination with NMA Status, as described in 3.1.3) that the previous chain has been revoked. The receiver can restart the NMA service. For that, it must perform the required number of chain steps to authenticate the new keys with the newly received KROOT.
- **Step 3**: The CPKS flag is set to nominal.

6.5 Public Key renewal and revocation

This section presents the public key renewal and revocation processes. As explained in section 3.3.2, the user shall be in the possession of a trusted public key, identified by a PKID, and associated to the DS algorithm to authenticate a DSM-KROOT message. Even though a public key is expected to be in force for several years, a public key renewal mechanism is foreseen. Note that the receiver can expect to receive public keys through DSM-PKR at a low rate even if no renovation or revocation process is on-going. Note that a public key renewal does not imply a TESLA chain renewal.

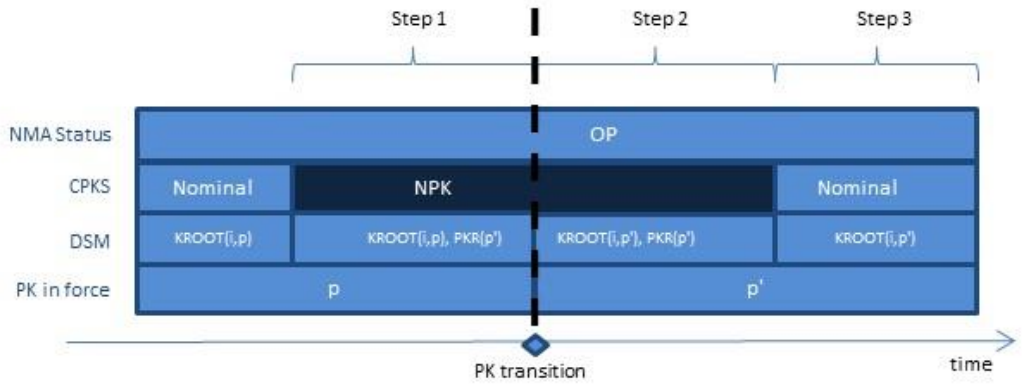

Figure 8 – Public key renewal

The public key renewal mechanism is depicted in Figure 8 and comprised by the following steps:

- **Step 1**: CPKS is set to NPK (New Public Key), reporting that the public key in force *p* is going to be replaced. During this step, the DSM alternates two messages with different DSM ID: a DSM-KROOT verifiable with *p*, KROOT(*i,p*), and a DSM-PKR for the new key *p'*, PKR(*p'*), where PKID(*p'*) > PKID(*p*).
- **Step 2**: *p'* enters into force by the transmission of a new KROOT(*i,p'*) transmitted in a new DSM-KROOT, and verified with *p'*. The DSM alternates this new DSM-KROOT with PKR(*p'*) [6] from Step 1. CPKS is maintained to NPK. When *p'* enters

[6] The transmission of PKR(*p'*) in Steps 1 and 2 is maintained for a sufficient time for NMA users to authenticate the new key *p'*, which may imply its transmission during several days.

into force, *p* and any other public key with a PKID < PKID*(p')* is declared not in use, so at a given time only one public key is in force. From this time the receiver shall discard any previously stored public key.

- **Step 3:** CPKS is set back to Nominal, and only the DSMs of KROOT(*i,p'*) are transmitted.

Note that if a DSM-KROOT is signed with a key *p'* not in the possession of the receiver, and where PKID(*p'*) > PKID (*p*), *p* being the trusted public key:

- If the CPKS flag is set to NPK or PKREV, the receiver should get the new DSM-PKR(*p'*), authenticate *p'* and replace the public key.
- If the CPSK flag is set to Nominal, the receiver may have been switched off for a long time and have missed the PK update. The receiver may either get *p'* from the trusted server and authenticate it, if it has a network connection, or get a DSM-PKR transmitted by the SIS. DSM-PKR will be regularly alternated with DSM-KROOT but with a very low latency.

The public key revocation process is followed only when the public key in force is revoked, e.g. because it has been compromised, and therefore it is not expected in nominal operation.

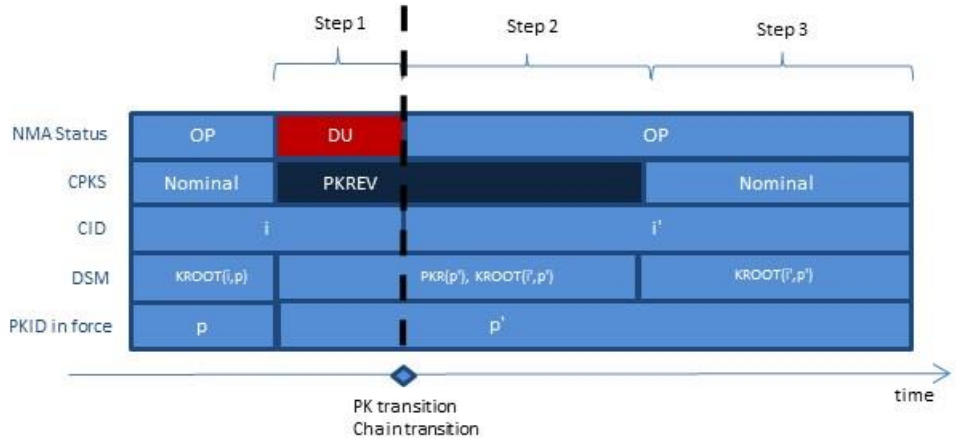

Figure 9 – Public key revocation

The public key revocation process is depicted in Figure 9 and comprised by the following steps:

- **Step 1**: The OSNMA Status flag is set to "Don't Use" and the CPKS flag is set to PKREV, reporting that the public key *p* is revoked. A DSM-PKR with the new public key *p'* and a new DSM-KROOT with a root key for a new chain *i'*, authenticated with *p'*, start to be broadcasted. By transmitting KROOT(*i',p'*), the new key *p'* enters into force.
- **Step 2**: The NMA Status is set back to "Operational". The PKREV flag reports that the previous public key has been revoked. The chain *i'* entries into force at the same time. From this time the receiver shall discard any previous public key and any KROOT associated to the previous TESLA chain.
- **Step 3**: CPSK is back to Nominal and the DSM-PKR stops being continuously transmitted.

6.6 Emergency Service Message

The DSM-PKR can transmit an Emergency Service Message through a specific value of the New Public Key Type (NPKT) of the DSM-PKR (NPKT = 4). In case such a message is received, and its authenticity is verified, the receiver shall stop NMA operation and check the OSNMA Server for further information. Receiver shall also discard any previous OSNMA cryptographic material.

6.7 Alarm conditions

This section is to be completed in the future and will include cases in which OSNMA data is not as expected and an alarm must be raised by the receiver. Specific actions to be taken from the alarm is out of the scope of this specification. Some conditions are already identified:

- An authentication verification event shall be performed over data that has been validated after CRC checking as per the Galileo OS SIS ICD [7]. A failed MAC authentication or DSM verification shall raise and alarm.
- The navigation message reference time given by GST differs from the internal reference time by an amount higher than the loose time maximum error allowed by the TESLA protocol.
- A DSM-KROOT is signed with a key p" different than the one in force by the receiver (p), where PKID(p") < PKID(p').
- Others TBC.

6.8 Chain cryptoperiods and lengths

The crypto algorithms, their parameters, and the cryptoperiods will be reevaluated periodically by the NMA Service Provider, taking into account changes in standards, improvement of cryptanalytic techniques, and decrease in hardware costs. As future changes in those are indeed foreseeable, it is recommended to include tests for different parameters in the test period.

At the moment, it is recommended that the chain be replaced after releasing 2^{25} to 2^{26} keys. For example, using 128-bit keys, NS=36 and 2 MACK blocks per sub-frame, a chain cryptoperiod between 5 and 10 months can be allowed.

6.9 MACK Offsetting

The key allocation described above corresponds to the case with no MACK Offsetting (i.e. Offsetting = '0' in DSM-KROOT). However, when set to '1', satellites can have an offset to relate its 30 second MACK cycle to the E1-B 30 second sub-frame. The purpose of this offset is to reduce the time between authentication events at user level, and improve TTFAF.

The offset values per PRN are defined in the Offsetting parameter, section 3.3.11. When the field is set to '1', PRNs 1 to 18 will have a zero offset and will use the key transmitted by PRN=1 for the MACs, and setting PRNs 19 and above to use *the key transmitted by PRN=19 for the MACs,* and have an offset defined as follows:

$$\Delta = round(\frac{18}{2 \cdot NMACK})$$

Where Δ is the offset, in number of I/NAV pages (1 page = 2 seconds), *round(A)* is the closest integer to *A* (the lowest in case two integers are the same close), and *NMACK* is the number of MACK blocks in a sub-frame. For example, if *NMACK* = 2, Δ = 4 (8 seconds), and if *NMACK* = 3, Δ = 2 (4 seconds). Note that MACK offsetting is only compatible with NS equal or greater than 2 (see section 7).

6.10 Use of I/NAV Page CRC and tag accumulation

MACs for a given navigation data set are provided with high frequency[7], including MACs for cross authenticated satellites, increasing the probability of MAC reception and minimizing the impact of missed MAC even in challenging environments.

It is recommended that the receiver applies the CRC first and then the authentication, in order to discern between data reception issues, and potential data spoofing attacks. The receiver can use the I/NAV CRC to ensure that an HKROOT byte in an I/NAV page is properly received, and recompose a DSM based on a DSM Header and several bytes from several satellites. Once DSM ID and BID are properly obtained from the DSM Header (I/NAV page 2), a receiver can accumulate 1-byte sub-blocks from CRC-valid pages of the same DSM ID, if reception conditions prevent continuously receive a Block ID from the same transmitter.

If the receiver application requires longer tags, the receiver can accumulate more than one tag for the same authenticated navigation dataset. This may not have a high impact in the time to authenticate, as several tens of tags can be received in a sub-frame, and the tags will mostly authenticate satellites visible by the receiver.

6.11 Management of different IODnav between I/NAV Words and OSNMA

Even if the OSNMA information will be generated with a latency of some seconds, the IODnav transmitted in the I/NAV message in E1-B (Words 1,2,3,4), once completed, will correspond, in nominal operation, with the IODnav authenticated by the tags transmitted at the same time, even when a new IODnav is transmitted. However, in some cases, the receiver may dispose of different IODnavs for navigation and OSNMA. For example, a receiver may be switched on in cold/warm start, receive the authentication of a new IODnav, but take some more sub-frames to decode the WT-1-4. In these cases, the NMA-PVT must be based on the last authenticated IODnav, as long as it is still valid according to [7].

6.12 Protection against replay attacks

This specification covers NMA only. Protection against replay attacks is subject to receiver implementations and therefore out of the main scope of this specification. Nevertheless, it can use the unpredictable symbols encoding the NMA message.

---

[7] For example, 8 MAC per sub-frame would be provided per transmitting satellite in a configuration with 2 MACK blocks per sub-frame, 128-bit keys and 12-bit MAC. User with 4 connected satellites in view would obtain up to 8 keys and 32 MAC per sub-frame.

6.13 DSM block sequencing and transmission

This subsection presents two preconditions aimed at improving the reception of the DSM in noisy environments. With these two preconditions, a receiver can reconstruct a full DSM just by receiving a single DSM Header:

- The DSM blocks are transmitted sequentially (i.e. BID=0, BID=1…) with a predefined offset between satellites, as per Table 23. The offsetting value *(*Δ_{DSM}) can also be calculated as Δ_{DSM} = *mod(PRN,8)*, where *mod(a,b)* is the *modulo* operation and returns the reminder of *a / b*. This precondition allows a receiver to determine the BIDs of a DSM transmitted at a given sub-frame by different satellites, by just receiving one DSM header from one satellite.
- A DSM associated to a given DSM ID shall be transmitted in its entirety before the OSNMA service starts transmitting another DSM. That means that an 8-block DSM will be fully transmitted at least once from BID=0 to BID=7 by a PRN with Δ_{DSM} = 0 (e.g. PRN=8), from BID=1 to BID=0 by a PRN with Δ_{DSM} = 1, etc. This precondition allows a receiver to determine the preceding and/or incoming BIDs from a given satellite, without having to receive them.

Offsetting:	0	1	2	3	4	5	6	7
PRNs	8, 16, 24, 32	1, 9, 17, 25, 33	2, 10, 18, 26, 34	3, 11, 19, 27, 35	4, 12, 20, 28, 36	5, 13, 21, 29	6, 14, 22, 30	7, 15, 23, 31

Table 23 – DSM offsetting values TBC

7 CONFIGURABLE PARAMETERS, GUIDELINES FOR SIS TESTING AND OTHER CONSIDERATIONS

Many parameters are configurable as part of the operational specification, in order to cope with changes over the lifetime of the service. In addition, there are some parameters that should be fixed in the operational specification but must be configurable in both the OSNMA data generator and the OSNMA test receivers during the SIS OSNMA transmissions, and others which can be configurable in the final specification but can be fixed for testing purposes:

- NS (Number of Satellites) and symbol unpredictability: The main reason for associating a different key to different satellites is to add frequent and continuous unpredictability to the symbols in the NMA field, in order to increase protection against replay attacks, in combination with receiver measures and under certain conditions. In exchange, a high NS increases the distance between a given key with a root key, implying a higher receiver resource consumption. Therefore, the value of NS is a parameter to be fixed during the OSNMA SIS experimentation with support from the user and receiver community. For the SIS experimentation, NS, described in section 5.4, must be set by default to 36, but it must be configurable between 1 and 36. The NS values used for OSNMA experimentation will be reported by Galileo through other means than the SIS (e.g. at the GSC website), although they can be implicitly detected from the SIS as well. When NS is lower than the number of satellites, keys will be repeated as necessary (e.g. if NS=4, the keys will be repeated as follows: [PRN=1 -> K_1; PRN=2 -> K_2… PRN=4 -> K_4; PRN=5->K_1…].
- Similarly, the time elapsed between the MAC and associated key transmission drives the loose time synchronization requirement in the receivers. The MACK blocks between a tag and its associated key (do not confound with the MACK offset) must be configurable between 1 and 0 and set to '0' by default (i.e. the tag and associated key are transmitted in the same MACK block). A value of '1' implies that a tag will be authenticated with a key broadcast in the *next* MACK block. Note that this is a receiver configuration parameter for testing purposes and is not part of the OSNMA SIS message specification.
- The cryptographic functions implemented can be reduced to those allowing the proof of concept, provided that the non-implemented functions will provide similar results.
- The mask of authenticated bits for each ADKD should be configurable in the receiver implementation, allowing its modification it if needed.
- Note that the cryptographic functions proposed are based on current standards. The resistance of some of these functions to quantum cryptography is not confirmed (e.g. ECDSA). Future standards resistant to quantum cryptography will be taken into account when available.

ANNEX A – BITMASK DEFINITIONS AND TEST VECTORS

A.1 TEST VECTORS

This section provides a set of OSNMA test vectors with the purpose of supporting OSNMA implementation in receivers. Unless otherwise stated, the sample data has been generated using the following OSNMA configuration:

```
NS (Number of satellites) = 36 satellites
KS (Key size) = 128 bits
MACLT (MAC lookup table) = 11
NMACK (Number of MACK blocks per section) = 2
HF (Hash function) = SHA-256
MF (MAC function) = HMAC-SHA-256
MS (MAC size) = 12 bits
MACK offset = No Offset
Seed key (hex): 0xFEAE8911880EFFFCC81FFA45F427F6EA
α pattern (hex): 0xF1CA3856A975
Key chain start/end time:
 - Week number 947, Day of week 5
 - Week number 947, Day of week 6
```

A.1.1 Binary data representation conventions

The hexadecimal string values in the following sections are right padded (LSB) with zeroes whenever the number of bits of the value to be represented is not a multiple of 8. Strings of hexadecimal and binary values are always in big-endian byte order with the bits transmitted first in the MSB position (which appears to the left of the string). Binary values are prepended by the '0b' prefix, hexadecimal values are prepended by the '0x' prefix, decimal numbers have no prefix.

A.1.2 Key chain test vectors

This section reports "intermediate" keys generated for the first 3 MACK blocks (36 * 3 = 108 keys) and their associated chain link position. It shall be noted that the key with index 0 corresponds to the root key transmitted over the DSM KROOT section. The last key of the chain key (subframe starting at 947/604770) has index 414720 and the following value:

$K_{SEED} = K_{414721} =$ 0xFEAE8911880EFFFCC81FFA45F427F6EA
$K_{414720} =$ 0xA8F6692E5C1258E3CCF941ADBAF21615
$K_{ROOT} =$ 0xEE6772D9AB8396866DC57EADA1D29637

Key index	WN	TOW	Key bits
414720	947	604770	0xA8F6692E5C1258E3CCF941ADBAF21615
414719	947	604770	0xF7CFDA81E1C4E83B227F18F0F226ADC6
414718	947	604770	0xD35422AB710779BE8ADF24013D9230A6
...	...	...	...
108	947	432030	0x5EA3A18FB127D4C7B31812C382D4C96D
...	...	...	...
73	947	432030	0x4E0E2DA7F80F547B874D4A2533316389
72	947	432000	0x47F767BFDC6674B6F108BE17A0198751
71	947	432000	0x3CA9190D0B21026D70E7FF8BAD6C6ED0
...	...	...	...

Key index	WN	TOW	Key bits
2	947	432000	0x22B30FBEE8C6C4A43480AF28A67D4A65
1	947	432000	0x81AEE575195E13C06961A705A191B9CD
0	947	431970	0xEE6772D9AB8396866DC57EADA1D29637

Table 24: Key chain test vector.

A.1.3 Key verification

This section describes the verification of the key $\mathbf{K}_2$ with chain position link equal to 2 (received from satellite E02) against the root key, $\mathbf{K}_0$. **The message to hash is composed concatenating $\mathbf{K}_2$,** GST_{SF}, α and *P3*:

K_2 (hex): 0x22B30FBEE8C6C4A43480AF28A67D4A65
GST_{SF} (hex): 0x3B369780
α (hex): 0xF1CA3856A975
P3 (hex): n/a

Note that the GST_{SF} is equal to 947/432000 and that the P3 length is equal to zero (as the message to be hashed is already a multiple of 8 bits). The result of the concatenation is the following:

0x22B30FBEE8C6C4A43480AF28A67D4A653B369780F1CA3856A975

The message is hashed and the following hash is obtained:

0x81AEE575195E13C06961A705A191B9CDBD04D00EF3867C1B82D2E62FB7357B19

$\mathbf{K}_1$ is obtained retaining the first KS=128 MSB bits of the hash:

0x81AEE575195E13C06961A705A191B9CD

Another step shall be performed to generate a local replica of $\mathbf{K}_0$. **The message to hash is composed concatenating $\mathbf{K}_1$,** GST_{SF}, the α and the *P3* values:

K_2 (hex): 0x81AEE575195E13C06961A705A191B9CD
GST_{SF} (hex): 0x3B369762
α (hex): 0xF1CA3856A975
P3 (hex):

Note that, even if the associated time of applicability of the root key is 947/432000, the GST_{SF} value to be used in the chain verification is equal to 947/431970. The result of the concatenation is the following:

0x81AEE575195E13C06961A705A191B9CD3B3697623B369762

The message is hashed and the following hash is obtained:

0xEE6772D9AB8396866DC57EADA1D29637F96908257E39FAD6898BB8230327E714

The $\mathbf{K}_0$ is obtained retaining the first KS=128 MSB bits of the hash:

0xEE6772D9AB8396866DC57EADA1D29637

The computed $\mathbf{K}_0$ local replica can now be verified against the root key obtained from DSM KROOT, performing a bit-by-bit comparison.

A.1.4 DSM-KROOT verification

This section reports test vector for the DSM-KROOT message. The following table reports the result of the decoding of a DSM-KROOT message having this binary value:

```
0x2020410B03B378F1CA3856A975EE6772D9AB8396866DC57EADA1D2963715E81EE289C9F
6F54869405F5E115E424777D11D598D2451CC576C2837A3984715B22FD153EF85179EA6D4
BD0101DB1C0E363A19DCA1625034F2CCF9D0E763E3A442FF8199A7D3C8CEF9B2
```

Field	Value
NB	0b0010
PKID	0b0000
CIDKR	0b00
NMACK	0b10
HF	0b00
MS	0b00
KS	0b0100
MS	0b0001
MACLT	0b00001011
Rsvd	0b00
MO	0b00
WN	0b001110110011
TOWH	0b01111000
Alpha	0xF1CA3856A975
KROOT	0xEE6772D9AB8396866DC57EADA1D29637
DS	0x15E81EE289C9F6F54869405F5E115E424777D11D598D2451CC576C2837A39847 15B22FD153EF85179EA6D4BD0101DB1C0E363A19DCA1625034F2CCF9D0E763E3
P1	0xA442FF8199A7D3C8CEF9B2

Table 25: Decoded DSM-KROOT message.

The message M is produced by concatenating the fields described in section 5.1. The message to be signed is:

```
0x8220410B03B378F1CA3856A975EE6772D9AB8396866DC57EADA1D29637
```

Note that the following NMA Header data has been used for the generation of M:

```
NMA Header: 0b10000010
```

Signature can be verified with the following public key (here provided in PEM format).

```
-----BEGIN PUBLIC KEY-----
MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEc7RAsKw8qkRfvlVvn4p4la1XX0K0
rTSveYwqGMW8OovntszzLiy4dTGcSkvwo0gejSbRjkRnMpQBta/RDLnWOA==
-----END PUBLIC KEY-----
```

A.1.5 MAC0 verification

At time (WN/TOW) 947/432033, the user receives from satellite E18 the following MAC0 authentication tag:

```
[MAC,MACSEQ,IOD]: [0b111001011000,0b110000100100,0b0000]
```

The authentication tag is received in the first MACK block of the subframe starting at 432031 (and thus having GST_{SF} equals to 432030); it's the first tag of the block and as consequence the CTR is equals to 1. The message to be verified is composed by concatenating the following binary values:

PRN_A = 18 (8 bits): 0b00010010
GST_{SF} Week number = 947 (12 bits): 0b001110110011
GST_{SF} Time of week = 432030 (20 bits): 0b01101001011110011110
CTR = 1 (8 bits): 0b00000001
NMA_Status_Header = Operational (2 bits): 0b10
navdata (549 bits): Extracted from I/NAV pages reported in Table **26**
P3 (1 bit): 0

The resulting message to be authenticated is the following:

0x123B36979E018507080CD1C003400000002A812D29050A1EFEA9227D27D280000000000
005000000000000000000000000000019141200000000708000000000000000320000000000
000000

The message is hashed with key having chain link position equal to 73 (reported in Table **24**) and corresponding to the one transmitted by satellite E01. The result of the MAC function is:

0xE5819AB582F2269E68FCB6AB5B368041E4E87F25320905E08EB48113ADEE997A

The hashed message is truncated retaining the MSB, obtaining the following local replica of the authentication tag:

0b111001011000

The verification is performed by checking if the local replica matches with the received tag.

The first step for the verification of the MACSEQ is the generation of the message to be verified by concatenating the following data:

PRN_A = 18 (8 bits): 0b00010010
GST_{SF} Week number = 947 (12 bits): 0b001110110011
GST_{SF} Time of week = 432030 (20 bits): 0b01101001011110011110
$MFLEX_1$ = 0b1111111101000000
$MFLEX_2$ = 0b0001001010110000

As defined in the MAC Lookup Table (row having ID equals to MACLT, which in this example is 11) the $MFLEX_1$ and $MFLEX_2$ are the MAC-Info of the second and the third tags transmitted in the first MACK block. The following message to be authenticated is the generated:

0x123B36979EFF4012B0

The message is hashed with key having chain link position equal to 73 (reported in Table **24**) and corresponding to the one transmitted by satellite E01. The result of the MAC function is:

0xC24C7E4C55A29896758A48657E858425C396C648B3AAC2B199E594F88F40E4B3

The hashed message is truncated retaining the MSB, obtaining the following local replica of the MACSEQ. The verification is performed by checking if the local replica matches with the received MACSEQ.

```
0b110000100100
```

A.1.6 MAC verification

At time (WN/TOW) 947/432039, the user receives from satellite E17 the following authentication tag:

```
[MAC,MAC-Info]: [0b011101100001,0b0001001000000000]
```

The authentication tag is received in the first MACK block of the subframe starting at 432031 (and thus having GST_{SF} equals to 432030); it's the fourth tag of the block and as consequence the CTR is equals to 4. The following information is extracted from the MacInfo:

```
PRN_A: 18
ADKD: 0
IOD: 0
```

The message to be verified is composed by concatenating the following binary values:

PRN_N = 18 (8 bits): 0b00010010
PRN_A = 17 (8 bits): 0b00010001
GST_{SF} Week number = 947 (12 bits): 0b001110110011
GST_{SF} Time of week = 432030 (20 bits): 0b01101001011110011110
CTR = 4 (8 bits): 0b00000100
NMA_Status_Header = Operational (2 bits): 0b10
navdata (549 bits): Extracted from I/NAV pages reported in Table **26**
P3 (1 bit): 0

The resulting message to be authenticated is the following:

```
0x12113B36979E048507080CD1C003400000002A812D29050A1EFEA9227D27D2800000000
0000500000000000000000000000000001914120000000070800000000000000032000000000000000
```

The message is hashed with key having chain link position equal to 73 (reported in Table **24**) and corresponding to the one transmitted by satellite E01. The result of the MAC function is:

```
0x76164E0DB6504991E064D47AAAD165372E204ACF23A0051BE9125C637769E77D
```

The hashed message is truncated retaining the MSB, obtaining the following local replica of the authentication tag:

```
0b011101100001
```

The verification is performed by checking if the local replica matches with the received tag.

WN	Time of transmission	Page part index	Data
947	432001	1	0x0214287BFAA489F49F4A0000000000
947	432002	2	0x80006091123D022AAAAA4CCEED08C0
947	432003	3	0x0414120000000070800000000000000
947	432004	4	0x800040D2C280096AAAAA63755748C0
947	432005	5	0x06000000000000001278B3B1424D00
947	432006	6	0xAF0AB3B444011EEAAAAA785A6608C0
947	432007	7	0x070ED04C0000000000007BA2800580
947	432008	8	0x93355740403A62AAAAAA4FD814C8C0
947	432009	9	0x0800000000028000000000003DD00
947	432010	10	0x940029A1FCFB512AAAAA52EB5848C0
947	432011	11	0x0C020408102040810204080000000
947	432012	12	0x80008E5CE339EBAAAAAA51D24948C0
947	432013	13	0x0E000000000000000000000000000
947	432014	14	0x80002C68E8E3446AAAAA47B452C8C0
947	432015	15	0x0B880204081020408102040810204
947	432016	16	0x8081794BEE39C86AAAAA646AB788C0
947	432017	17	0x0D0000023B36978000000000000
947	432018	18	0x800039C800AA80EAAAAA4DC46CC8C0
947	432019	19	0x0F0000000000000000000000000680
947	432020	20	0x97935F80003C682AAAAA68067BC8C0
947	432021	21	0x01141C203347000D00000000AA0480
947	432022	22	0xB4A48540AD06C02AAAAA5C8FF248C0
947	432023	23	0x0314000000000000000000000000
947	432024	24	0x8006707E8D4EF8EAAAAA765C45C8C0
947	432025	25	0x0532000000000000000766D2F32A80
947	432026	26	0xAAAA9844C78F936AAAAA7121B188C0
947	432027	27	0x009555555555555555555554ECD80
947	432028	28	0xA5E6E8666C215C6AAAAA403ED148C0
947	432029	29	0x009555555555555555555554ECD80
947	432030	30	0xA5E74F4AD42E5DAAAAAA5E2B6D88C0

Table 26: Satellite E18, E1B I/NAV data.

A.1.7 DSM-PKR

This section reports test vector for the DSM-PKR message. The following table reports the result of the decoding of a DSM- PKR message having this binary value:

```
0x70A5E09C16A42D37D584D63797D684ED5D24F12CF99553033B01FACBBC79EEBF9C743A5BC5089
7F9A5E78FB0733D425B541874398ABB0E12DD6C2D585035ECBF09C978D80C3F476D3D5B7129003F
735CB5019E995BB9FB6CF7045CCFF0039965F775943C3286BA8222E1B6437D12507436C0BF38BBB
B5FD856D9D948EF8FB3BAEC0002D25BDF123D1CB876022BD071BC2372E4132DC62E627C1988D4E7
272619148C51B7F0EED951EA
```

Field	Value in binary format
NB	0b0111
MID	0b0000
ITN[0]	0xA5E09C16A42D37D584D63797D684ED5D24F12CF99553033B01FACBBC79EEBF9C
ITN[1]	0x743A5BC50897F9A5E78FB0733D425B541874398ABB0E12DD6C2D585035ECBF09
ITN[2]	0xC978D80C3F476D3D5B7129003F735CB5019E995BB9FB6CF7045CCFF0039965F7
ITN[3]	0x75943C3286BA8222E1B6437D12507436C0BF38BBBB5FD856D9D948EF8FB3BAEC
NPKT	0b0000
NPKID	0b0000
NPK	0x02D25BDF123D1CB876022BD071BC2372E4132DC62E627C1988D4E72726
P2	0x19148C51B7F0EED951EA

Table 27: Decoded DSM-PKR message.

The MID is equal to 0 and as consequence the intermediate nodes are the following: $x_{0,1}$, $x_{1,1}$, $x_{2,1}$, $x_{3,1}$. The NPK can be verified with the following steps. The m_0 is composed as defined in section 5.2, m_0 is hashed to recover the $x_{0,0}$. The $x_{0,0}$ is prepended to the received $x_{0,1}$ value, the concatenation is hashed to generate the $x_{1,0}$. The $x_{1,0}$ is prepended to the received $x_{1,1}$ value, the concatenation is hashed to generate the $x_{2,0}$. The $x_{2,0}$ is prepended to the received $x_{2,1}$ value, the concatenation is hashed to generate the $x_{3,0}$. The $x_{3,0}$ is prepended to the received $x_{3,1}$ value, the concatenation is hashed to generate the $x_{4,0}$. A bit-by-bit comparison between the $x_{4,0}$ and the preloaded Merkle tree root is performed to ensure the two matches.

m_0 = 0x0002D25BDF123D1CB876022BD071BC2372E4132DC62E627C1988D4E72726

The $x_{j,i}$ values are reported in section Table **28**; Table **29** reports test vector for the leaves of the tree.

Idx	Node (Hex)
$x_{4,0}$	0x5E53B01CC55A978180040E95AB129F2E2C4B65CBDFA849E4DE9E26AC7315A49D
$x_{3,0}$	0xEC676ADB02BD018DFC67E0DC4C15321AA2E9E1EF120C4BE43827162DC7CE2AE6
$x_{3,1}$	0x75943C3286BA8222E1B6437D12507436C0BF38BBBB5FD856D9D948EF8FB3BAEC
$x_{2,0}$	0x6EB60E650EC92B951C51167F3A88C089E8BB6E5A98250270962FE1FEE818BD32
$x_{2,1}$	0xC978D80C3F476D3D5B7129003F735CB5019E995BB9FB6CF7045CCFF0039965F7
$x_{2,2}$	0x1F86DB6AE3596AC8CB5FECBAED58213A4CCAC59E96B11F29B2760AF0C9489E7A
$x_{2,3}$	0x1E10CA440544A7BCFEFEC71E5E65052D8B182A4DE72139521F9B9431AE5B05A8
$x_{1,0}$	0x097C2DC621590E80EFF8B4278BCCC35F016828D44EEB8421DA3D2DE15D2444BE
$x_{1,1}$	0x743A5BC50897F9A5E78FB0733D425B541874398ABB0E12DD6C2D585035ECBF09
$x_{1,2}$	0xFBC67E885397F9258581FF155571F69CE53477B22FC2A2CA0ED475ED7446403B
$x_{1,3}$	0xC0F37BF54F4CA4D849D6EE61B77BCFD01CEA8D72B80F076F271072D2247F484D
$x_{1,4}$	0x490638E121DE28F5584223A6D03D5B612E0FA8DCC1A023DF96ABF10C8B48DCE5
$x_{1,5}$	0x5552DC56A54D8E554B818C4CF50772039F3E4521B1B762B1F952AA0EE049E3BD
$x_{1,6}$	0x341E0636E6A62B96D8B93D8DBEB55F0252D3F1BDC27599BF728127BAFB6130F2
$x_{1,7}$	0xB90FCEDEFC01A1528D3836FDD272333862E00789E0AC452498D6CDB1BA19CB47
$x_{0,0}$	0x6F826E2AD5CF4292B8402AAEE16D01AFB6C8A2EB6603D1272BE992229CBBD0E8
$x_{0,1}$	0xA5E09C16A42D37D584D63797D684ED5D24F12CF99553033B01FACBBC79EEBF9C
$x_{0,2}$	0x0394A6102822FAC4C348E5FED6C3B3AB07E09930987DC48B196AACDE30C78136
$x_{0,3}$	0x1CC95319D4494B52893CD293D3E449A83806CF01BE4EB1D48455E724528B1581
$x_{0,4}$	0xFCB3D7B6E9EC4237E0AA611ED31191F4041D672F2E278B6E1EBE1D4EC762F056
$x_{0,5}$	0x5EFD91A8AE07EA2298684106A5F455DDFEDFF12EB96A7EF3A9D29DC46AB2A8A8
$x_{0,6}$	0x662DFE866C0B50A266AD912F9810C0B3849E6792FDFDCE8E314BD89E2FC57BD6
$x_{0,7}$	0xB5E54677E1D76415AC658F7DF98CF5F4873D49E07D96E12C85660EE9D3D7C155
$x_{0,8}$	0xF90298304C7F751B6F4A6CC66AD9C465572BC19FDE8BC0C263563F29367A6B8C
$x_{0,9}$	0x9A6355979ADCB14823EC73B1B4026710CFCCCFA3A2ECA346EA79313A78D6102D
$x_{0,10}$	0x9D4DC478AE5CEE073A22ECC0FC335CBEC86F439B13262913759A648A828C2CE7
$x_{0,11}$	0x04A576E2CAEE22B4A9346B0FA0B4808D07BFED2A7FB8DCFB96A7FE7F6FE2D041
$x_{0,12}$	0xE7FFDB9E5FC193E420F7E0C07E4988EE5C1DD5C38BC4737EC23F10211544ED27
$x_{0,13}$	0xB6101E23C55DD76F0A541F1BA26026547D5A5EFAA7E4AC34E16E9539186D11B8
$x_{0,14}$	0xD95C43CBCC3493EBC10299B35ECC188567FAE224852DAED25E0809575E804897
$x_{0,15}$	0x82D99D41A166C338289294A00E4775DEC4712A70E4274CCE5583D72630FBCB89

Table 28: Merkle tree nodes.

MID	NPKT	NPKID	Data
0	0	0	0x0002D25BDF123D1CB876022BD071BC2372E4132DC62E627C1988D4E72726
1	0	1	0x010275D0AABCA64E8FB46950ECB05829A3C6B3DBC18D1F21394AB7E4D551
2	0	2	0x0202D87CFFCD808389E0734DBC5D8FDED33AC88D52266D11573F54D6CC27
3	0	3	0x030211CC9CC878052C549BA3258083A745D8EDD9DA347647D8EB214F3103
4	1	4	0x14029A9B98F15F2515D9C93A0F465B96FF64EF5D8A1FAB4028A3FA03F452 30866DB6
5	1	5	0x15037BE971E04DF17A16D7A7A253FC5BF0B67B5CAC8E3678CCE2B887B21C 2D5BA2DD
6	1	6	0x1603E457289C1BE099F2051F0F49EBF99145588B9D7768CA756BA1B760AD A50D59A7
7	1	7	0x1703BD770F7B68F6AD2F681D0F9734D259F3F34B0FE476847C905AC0D645 F8690725
8	2	8	0x2802F17161022DADEB5EF25F5B03D044558F019F967B213171254D7F2B7F 7D60A744B8E482B4B56B28B375B9E389038DF0DD
9	2	9	0x2902FB00F20FA341FD48080012C1CDE2286034957A49953C611F10CCD244 B62B9B5353922DC2538FB0688ACFA2B8EBD71D10
10	2	10	0x2A03822B3667317D8EDF6FEC4A6A25C185D19603609AB1E0FF124EA36F44 7875DCFA4DB4979FFC83A99F96DD9A22360A44CC
11	2	11	0x2B02F7C33744AC74251A3AFC44220023FEA60CFC9523B4F451495660E6C4 9EE9D663CD8AD1AE7FC8C4BEC8A16A9905684F31
12	3	12	0x3C02002F26BC3F4DE9EABA42C2D8391B2EB2A090E6137A3CAACB4FB8C1B1 4EA2C224D4A899D0D6FE8F3F3192FE240CF68C45686F2D05203BE5B28C9973 8D20D8252BFBDD
13	3	13	0x3D03002C672BF21D70F55BD07D1EC72EBBFCE528B09A23FFC7AA57524FA5 C4862876D6940DC85063FEC7FE5C9BB07FD091A5D78C11CDA34037B0F41F09 955404187A5CC9
14	3	14	0x3E030112ABD96E36AD88508B0E594019DB5FBE718F6AF76540751ABADFD1 9A7334F68C8DBDF481BCACF64919F0104933C210F61BC64F36FCD8DE3FE43F 08CDFF6085F6ED
15	4	15	0x4F2A2A2A454D455247454E435920534552564943452044553534147452A 2A2A2A0A

Table 29: Merkle tree leaves.

ANNEX B - MAC LOOKUP TABLE

The current OSNMA specification allows flexibility to authenticate different data from different satellites. The MAC sequence is partially fixed for each chain through a lookup table, in order to protect against its potential modification. Table 30 defines the MAC lookup table and presents some ADKD sequences. Only 24 out of 256 possible sequences are defined at the moment. More sequences may be added during OSNMA SIS experimentation.

The first column (ID) is the entry value to the table as per DSM-KROOT MACLT, section 3.3.9. It is maintained for the duration of the chain. The second column (Sct) specifies the number of MACK sections (1 or 2) for which the sequence is defined. The third column (NMACK) specifies the number of MACK blocks per section, and is shall be the same as the DSM-KROOT NMACK, section 3.3.4. The fourth column (MACs/blc) shows the *maximum* number of MACs that can be provided in a block, for a given NMACK. The last column specifies the MAC slot sequence, and is interpreted as follows:

- Every slot is represented by three characters:
  - Flexible slots are represented as '`FLX`'. They are not fixed in the lookup table and their MAC-Info data is authenticated as per 5.6.
  - All other slots are fixed slots. Their first 2 characters define the ADKD (from '`00`' to '`12`', as per 4.1.4.2), and the last character means '`S`' for Self-authentication, '`E`' for Galileo cross-authentication, and '`G`' for GPS cross-authentication. For example: '`12S`' means ADKD=12, self-authentication; and '`00E`' means Galileo cross-authentication. Cross-authenticated satellites are transmitted from closest to farthest from a given type (GPS, Galileo). The first element of the sequence is always fixed to '`00S`' and corresponds to MAC0, section 4.1.1.
- Blocks containing flexible slots are coloured in light orange. Only the first block of a section can contain flexible slots.
- If the NMACK/KS/MS triplet of the chain provides less MACs than the maximum, the sequence is truncated from the left, ignoring the last positions. For example, for ID=17 (MACs/blc=5), and a NMACK/KS/MS triplet of 2/112/16, which provides only 4 MACs per block, the sequence shall be truncated to the first 4 elements defined for each block: `00S,FLX,FLX,00G` (for block 1) and `00S,00E,00E,12S` (for block 2).

ID	Sct	NMACK	MACs/ Blc	Sequence					
0	1	1	14	00S,00E,00E,00E,00E,00E,00E,00S,00E,00E,00E,00E,00E,00E					
1	1	1	14	00S,00G,00G,00G,00G,00E,00E,00E,00E,00E,00E,00S,00G,00G					
2	1	1	14	00S,FLX,FLX,FLX,FLX,FLX,FLX,FLX,FLX,FLX,FLX,FLX,FLX,FLX					
3	1	1	14	00S,04S,03S,00E,00E,00E,00E,FLX,FLX,FLX,11S,12S,FLX,FLX					
4	1	1	14	00S,04S,03S,00G,00G,00E,00E,FLX,FLX,FLX,11S,12S,FLX,FLX					
5	1	1	14	00S,03S,05S,00E,FLX,04S,05G,00E,FLX,05E,12S,11S,FLX,FLX					
6	1	1	14	00S,03S,05S,00E,04S,05E,FLX,12S,FLX,FLX,FLX,FLX,FLX,FLX					
7	1	1	14	00S,FLX,FLX,FLX,00E,00E,00E,FLX,FLX,FLX,11S,12S,FLX,FLX					
8	1	2	5	00S,00E,00E,00E,00E			00S,00E,00E,00E,00E		
9	1	2	5	00S,00G,00G,00G,00G			00S,00E,00E,00E,00E		
10	1	2	5	00S,FLX,FLX,FLX,FLX			00S,00E,00E,00E,00E		
11	1	2	5	00S,FLX,FLX,00E,11S			00S,00E,00E,00E,12S		
12	1	2	5	00S,FLX,FLX,00G,11S			00S,00G,00E,00E,12S		
13	1	2	5	00S,03S,05S,FLX,00E			05E,00E,04S,00E,12S		
14	1	2	5	00S,03S,05S,FLX,00E			05E,05G,04S,00G,12S		
15	1	2	5	00S,04S,FLX,FLX,11S			00S,00E,00E,00E,12S		
16	1	2	5	00S,12S,00E,00E,FLX			00S,11S,00E,00E,00E		
17	1	2	5	00S,FLX,FLX,00G,FLX			00S,00E,00E,12S,11S		
18	2	3	2	00S,00E	00E,00E	00E,00E	00S,00E	00E,00E	00E,00E
19	2	3	2	00S,00G	00G,00G	00G,00G	00S,00E	00E,00E	00E,00E
20	2	3	2	00S,FLX	00E,00E	00E,00E	00S,FLX	00E,00E	00E,00E
21	2	3	2	00S,FLX	00E,00E	00E,11S	00S,FLX	00E,00E	00E,12S
22	2	3	2	00S,FLX	00G,00G	00G,11S	00S,FLX	00E,00E	00E,12S
23	2	3	2	00S,FLX	05S,03S	00E,11S	00S,FLX	04S,00E	05E,12S
24	2	3	2	00S,FLX	05S,03S	00G,11S	00S,FLX	04S,00G	05E,12S
25	…			…					
…									
255	…			…					

Table 30 – MAC Lookup Table (preliminary)

ANNEX C – LIST OF ACRONYMS

ADKD	Authentication Data & Key Delay
AGC	Automatic Gain Control
α_S	αSize
BGD	Broadcast Group Delay
C/N_0	Carrier to Noise ratio
CID	Chain ID
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
CREV	Chain Revocation
CTR	Counter
DS	Digital Signature
DSID	Digital Signature ID
DSM	Digital Signature Message
DSMH	Digital Signature Message Header
DVS	Data Validity Status
DSM BID	Digital Signature Message - Block ID
EDBS	External Data Broadcast Service
EOC	End of Chain
GPS	Global Positioning System
GSC	European GNSS Service Centre
GST	Galileo System Time
HF	Hash Function
HKROOT	Header and KROOT
HS	Health Status
ICD	Interface Control Document
IOD	Issue Of Data
IODC	Issue Of Data - Clock
IODE	Issue Of Data – Ephemeris
IRNSS	Indian Radio Navigation Satellite System
KROOT	Root key
KS	Key Size
LSB	Least Significant Bit
MAC	Message Authentication Code
MACK	MAC & Key (section)
MACLT	MAC Lookup Table
MF	MAC Function
MS	MAC Size
MSB	Most Significant Bit
n/a	Not Applicable
NB	Number of Blocks
NS	Number of Satellites
MOPS	Minimum Operational Performance Standards
NMACK	Number of MACK Blocks
OS	Open Service
OSNMA	Open Service Navigation Message Authentication
PK	Public Key
PKID	Public Key ID

PKR	Public Key Renewal
PRN	Pseudo Random Noise
RFI	Radio Frequency Interference
RLM	Return Link Message
RLSP	Return Link Service Provider
rsvd	Reserved
SAR	Search and Rescue
SBAS	Satellite-Based Augmentation Systems
SHA	Secure Hash Algorithm
SIS	Signal In Space
SISA	Signal-In-Space Accuracy
SVID	Space Vehicle ID
TBD	To Be Defined
TLM/HOW	Telemetry/Handover Word
TOW	Time Of Week
URA	User Range Accuracy
WN	Week Number

ANNEX D – BIBLIOGRAPHY

[1] European Commission , *COMMISSION IMPLEMENTING DECISION (EU) 2017/224 of 8 February 2017 (CS Implemeting Act),* 2017.

[2] European Commission, *COMMISSION IMPLEMENTING DECISION (EU) 2018/321 of 2 March 2018 amending Implementing Decision (EU) 2017/224 (CS Implemeting Act).,* 2018.

[3] A. Perrig, R. Canetti, J. Tygar and D. Song, "Efficient Authentication and Signing of Multicast Streams over Lossy Channels," *IEEE Symposium on Security and Privacy,* pp. 56-73, May 2000.

[4] National Institute of Standards and Technology, "FIPS PUB 186-4 - Digital Signature Standard (DSS)," U.S. Department of Commerce, 2013.

[5] R. C. Merkle, "Method of providing digital signatures". Patent US4309569A, 1979.

[6] I. Fernandez-Hernandez, V. Rijmen, G. Seco-Granados, J. Simon, I. Rodriguez and J. D. Calle, "A Navigation Message Authentication Proposal for the Galileo Open Service," *Journal of the Insitute of Navigation,* no. Spring, pp. pp. 85-102, 2016.

[7] European Union, "OSSISICD: Open Service Signal In Space Interface Control Document, Issue 1.3," 2016.

[8] National Institute of Standards and Technology, "FIPS PUB 180-4: Secure Hash Standard (SHS)," 2012.

[9] National Institute of Standards and Technology, "FIPS PUB 202: SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions," 2015.

[10] H. Krawczyk, M. Bellare and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication," *Network Working Group,* 1997.

[11] International Organization for Standardization, "ISO/IEC 9797-1:2011: Information technology - Security techniques - Message Authentication Codes (MACs) - Part 1: Mechanisms using a block cipher," 2011.

[12] National Institute of Standards and Technology, "FIPS PUB 198-1: The Keyed-Hash Message Authentication Code (HMAC)," 2008.

[13] National Institute of Standards and Technology, "NIST Special Publication 800-38B: Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication," 2004.

[14] The US Government, “GPS Interface Specification IS-GPS-200,” 2014.

[15] Russian Institute of Space Device Engineering, “Glonass Interface Control Document - Navigational Radiosignal In Bands L1, L2 (Edition 5.1),” 2008.

[16] China Satellite Navigation Office, “BeiDou Navigation Satellite System Signal In Space - Interface Control Document - Open Service Signal B1I (Version 1.0),” 2012.

[17] RTCA SC-159, “Minimum Operational Performance Standards for Global Positioning System - Wide Area augmentation System Airborne Equipment - DO229D (with Change 1, Feb 2013),” 2006.

[18] China Satellite Navigation Office, “BeiDou Navigation Satellite System Signal In Space - Interface Control Document - Open Service Signal - Version 2.0,” China Satellite Navigation Office, December 2013.

End of the Document