NAVIGATION SOLUTIONS POWERED BY EUROPE

GALILEO OPEN SERVICE NAVIGATION MESSAGE AUTHENTICATION (OSNMA) RECEIVER GUIDELINES FOR THE TEST PHASE

Issue 1.0, November 2021

TERMS OF USE AND DISCLAIMERS

1. Authorised Use and Scope of Use

The European GNSS (Galileo) Open Service Navigation Message Authentication (OSNMA) Receiver Guidelines for the Test Phase Issue 1.0 (hereinafter referred to as OSNMA Receiver Guidelines for the Test Phase) and the information contained herein is made available to the public by the European Union (hereinafter referred to as Publishing Authority) as part of the OSNMA Test Phase for information, standardisation, research and development and commercial purposes for the benefit and the promotion of the European Global Navigation Satellite Systems programmes (European GNSS Programmes) and according to terms and conditions specified thereafter.

2. General Disclaimer of Liability

With respect to the OSNMA Receiver Guidelines for the Test Phase and any information contained in the OSNMA Receiver Guidelines for the Test Phase , neither the EU as the Publishing Authority nor the generator of such information make any warranty, express or implied, including the warranty of fitness for a particular purpose, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information hereby disclosed or for any product developed based on this information, or represents that the use of this information would not cause damages or would not infringe any intellectual property rights. No liability is hereby assumed for any direct, indirect, incidental, special or consequential damages, including but not limited to, damages for interruption of business, loss of profits, goodwill or other intangible losses, resulting from the use of the OSNMA Receiver Guidelines for the Test Phase or of the information contained herein. Liability is excluded as well for consequences of the use and / or abuse of the OSNMA Receiver Guidelines for the Test Phase or the information contained herein.

3. Copyright

The OSNMA Receiver Guidelines for the Test Phase is protected by copyright. Any alteration or translation in any language of the OSNMA Receiver Guidelines for the Test Phase as a whole or parts of it is prohibited unless the Publishing Authority provides a specific written prior permission. The OSNMA Receiver Guidelines for the Test Phase may only be partly or wholly reproduced and/or transmitted to a third party in accordance with the herein described permitted use and under the following conditions:

- the present “Terms of Use and Disclaimers” are accepted, reproduced and transmitted entirely and unmodified together with the reproduced and/or transmitted information;
- the copyright notice “© European Union 2021” is not removed from any page.

- ISBN: 978-92-9206-056-5
- ISSN: 1831-9424
- DOI: 10.2878/230609

4. Miscellaneous

No failure or delay in exercising any right in relation to the OSNMA Receiver Guidelines for the Test Phase or the information contained therein shall operate as a waiver thereof, nor shall any single or partial exercise preclude any other or further exercise of such rights.

5. Updates

The disclaimers contained in this document apply to the extent permitted by applicable law.

The OSNMA Receiver Guidelines for the Test Phase is expected to be subject to modification, update and variations. Those modification, updates and variations will reflect, between others, the result of the execution of the OSNMA Test Phase.

The publication of updates will be subject to the same terms as stated herein unless otherwise evidenced.

Although the Publishing Authority will deploy its efforts to give notice to the public for further updates of OSNMA Receiver Guidelines for the Test Phase, it does not assume any obligation to advise on further developments and updates of the OSNMA Receiver Guidelines for the Test Phase, nor to take into account any inputs, comments proposed by interested persons or entities, involved in the updating process.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Scope of the Document

The scope of this document is to provide guidelines for the user segment implementation of the OSNMA functionality, as defined in the OSNMA ICD [AD.2] and should be considered strictly as a complement to it. The document provides requirements, interfaces and steps to be followed in order to verify the authenticity of the Galileo navigation message. These guidelines are drafted in a generic way and are not tailored for any specific platform or application.

The document also provides test vectors and sample data, supporting the verification of OSNMA functionality implementation.

The guidelines provided in this document enable an implementation of the OSNMA protocol, providing navigation data authentication. For the interested reader, examples of techniques that can be additionally exploited to obtain a more robust PVT solution are provided in Annex 4.

The current version of the OSNMA receiver guidelines still contains some TBD and TBC values that will be consolidated during the OSNMA test phase and fixed in successive updates of the document. Additionally, users should expect the possibility of tag failures during the OSNMA test phase. A quarterly report will be published on the GSC webserver, which will provide an indication of the tag failure rate and further information. This rate will be reduced in view of service provision phase, in part through evolutions of the OSNMA ICD [AD.2].

1.2 What is OSNMA

The Galileo Open Service (OS) is providing a Navigation Message Authentication (NMA) capability, allowing the users to confirm that received Galileo Open Navigation Data originated from the Galileo system and has not been modified.

The authentication concept is based on two main principles:

- The use of different keys from a single one-way chain shared by the Galileo satellites through a Timed Efficient Stream Loss-tolerant Authentication (TESLA) protocol [1][2][3].
- The possibility to authenticate satellites which do not transmit OSNMA data with the data retrieved from satellites transmitting OSNMA, referred to as cross-authentication.

Both principles reduce the computation and communication overhead, and increase the service availability and robustness to data loss.

From a receiver perspective, the process of the OSNMA data can be described at a high level by the following steps, illustrated in Figure 1:

- The receiver receives the **navigation data** and the corresponding OSNMA data (**tag**, **TESLA chain key** and **TESLA root key**). The **tag** authenticates the **navigation data** and is received before its associated **TESLA chain key**.
- The **TESLA root key** is authenticated by means of its digital signature using a **public key** that shall be available at the receiver.
- The receiver authenticates the **TESLA chain key** with the **TESLA root key** or with a previously authenticated key from the TESLA chain.
- The receiver re-generates locally the **tag** with the verified **TESLA chain key** and the **data**, and checks whether it coincides with the received **tag**.

If the result of all these steps is successful the user shall consider the **navigation data** as authentic.

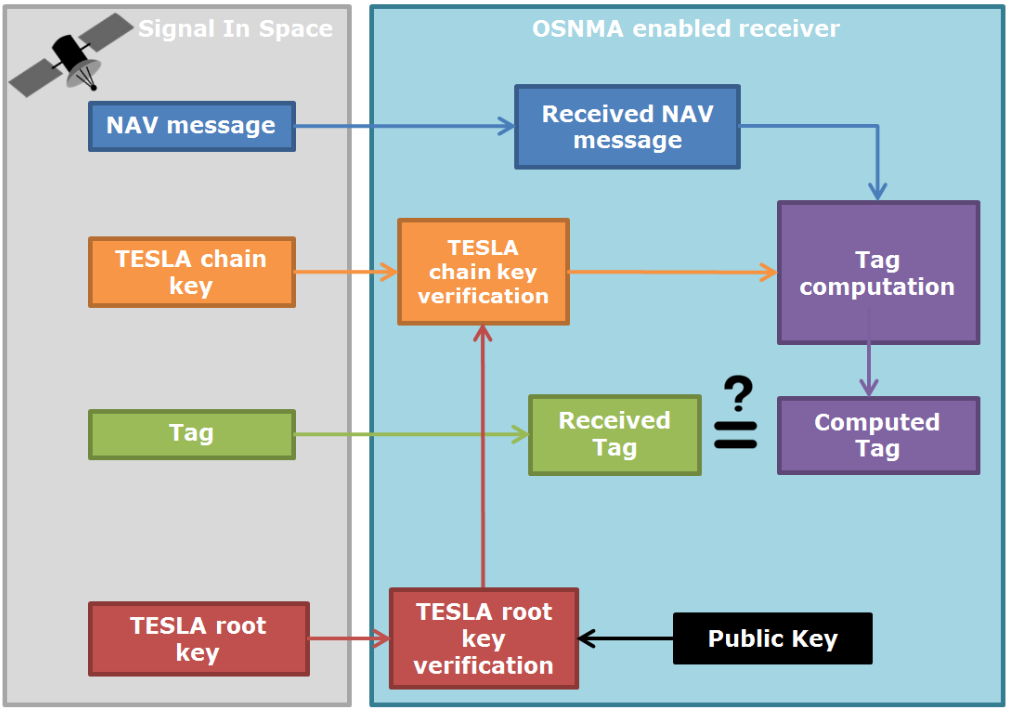


Figure 1. OSNMA processing logic

The retrieval of the data and operations required to perform these verification steps are further detailed in these guidelines. Specific strategies regarding the use of the information originated from the OSNMA verification steps, in the positioning process, are out of the scope of this document.

1.3 Structure of the Document

The document is structured as follows. After a short introduction in section 1, the requirements the receiver shall comply with to exploit OSNMA capabilities are presented in section 2. The data that need to be retrieved to exploit the scheme are presented in section 3. Section 4 shows how this data fits in the OSNMA verification workflow. The cryptographic operations that have to be applied to this data are presented in section 5.

Additional aspects are covered in Annexes:

- Annex 1 provides test vectors that can be used to verify the correct implementation of the cryptographic operations,
- Annex 2 describes OSNMA data structures for the Test Phase configurations,
- Annex 3 discusses ways for the receiver to fulfil the time synchronisation requirement,
- Annex 4 presents additional steps that can be taken by the receiver to obtain a more robust PVT solution.

1.4 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 integers as per Galileo OS SIS ICD [AD.1].

1.5 Applicable Documents

- AD.1 European GNSS (Galileo) Open Service, Signal-In-Space Interface Control Document, Issue 2, 2021

- AD.2 Galileo OSNMA User ICD for the Test Phase, Issue 1.0

2 Receiver Requirements

This section presents the requirements the receiver shall fulfil in order to exploit Galileo OSNMA capabilities.

2.1 Time Synchronisation Requirement

To ensure the security of the TESLA protocol and guarantee the authenticity of the data, the receiver must ensure it has received the navigation data and associated tag before the corresponding TESLA chain key is disclosed by the system. This implies that the receiver must be synchronised with a given accuracy to the Galileo System Time (GST) before receiving and processing OSNMA information. The time synchronisation requirement $\mathrm{T_L}$ is set to 30 sec [TBC]. If the receiver verifies this condition, all tags for all authentication types can be used.

If the condition is not verified, slow MACs, i.e. messages whose associated TESLA chain key is transmitted with an extra delay, may be exploited. A receiver synchronised to GST with an accuracy better than $\mathrm{T_L} + 300\ sec$, can process slow MAC with a 10 sub-frame delay (ADKD12 from [AD.2]).

If none of the above conditions on the receiver time synchronisation to GST is verified, the OSNMA protocol shall not be used.

Additional considerations on the receiver time synchronisation uncertainty with respect to GST the can be found in Annex 3.

2.2 Required Cryptographic Functions

The receiver shall be able to perform all the variants of the cryptographic functions listed in Table 1, according to their current standards. Future revisions of this document may consider additional algorithms. These functions are used for the verification of the OSNMA data, as further detailed in their indicated associated sections.

Function	Ref.	Variants	Applicable section
Elliptic Curve Digital Signature Algorithm (ECDSA)	[4]	ECDSA P-256 ECDSA P-521	Verification of the TESLA root key (5.2)
Secure Hash Algorithm 2 (SHA-2)	[5]	SHA-256 SHA-512	Verification of the TESLA root key, TESLA chain key and Public key (5.1, 5.2, 5.4)
Secure Hash Algorithm 3 (SHA-3)	[6]	SHA3-256	Verification of the Public Key and TESLA chain key (5.1, 5.4)
Hash-based Message Authentication Code (HMAC)	[7]	HMAC-SHA-256	Verification of the tag (5.5)
Cipher-based Message Authentication Code (CMAC)	[8], [9]	CMAC-AES	Verification of the tag (5.5)

Table 1. Required cryptographic functions

2.3 Integrity of the Cryptographic Material and Functions

The receiver or the system containing the receiver is responsible for ensuring the integrity of the cryptographic material stored in its memory and of the processing of OSNMA data at a security level corresponding to its needs. This security level shall be defined in accordance with the receiver or the system protection profile, taking into account the application it is supporting and the environment it is operated in. Integrity breaches shall be notified to the systems or applications which use the output of the receiver.

It shall also be considered that all cryptographic material may be subject to renewal or revocation.

2.4 Interfaces

To exploit the OSNMA capabilities, the receiver has to interface with the Galileo OS SIS, as described in [AD.1]. In addition, the receiver may interface with the OSNMA server of the European GNSS Service Centre (GSC) to retrieve the cryptographic material. In case an OSNMA Alert Message (as defined in [AD.2]) is received, the receiver is required to connect to the GSC OSNMA server. Interactions with these systems shall respect the requirements defined in [AD.2].

2.5 Memory Requirement

The different elements required by the receiver to exploit the OSNMA protocol can be retrieved from the SIS or from the OSNMA Server. In addition to the Merkle root (see section 3.1), the receiver may store some other elements of the protocol, in particular the public key and the TESLA root key (or an intermediate TESLA chain key previously verified). These elements shall be stored according to the requirement stated in section 2.3.

3 OSNMA Data Retrieval

In order to exploit the OSNMA capabilities of Galileo Open Service, a receiver should retrieve different information:

- A Merkle root;
- A public key;
- A TESLA root key;
- TESLA chain keys;
- Tags and associated navigation data.

As specified in the following section and detailed in [AD.2], this information can be retrieved from the GSC OSNMA server or from the Signal In Space (SIS). The OSNMA data are transmitted within the Galileo SIS 40-bit OSNMA field, within the E1-B I/NAV navigation message. The OSNMA data is transmitted only by a subset of satellites; the remaining satellites have an OSNMA field filled with zeroes and their navigation data are cross-authenticated by the satellites transmitting OSNMA data. As a consequence, the user shall discard any OSNMA field which is set to zero (i.e. 40 bits set to zero).

It shall be noted that in case of non-continuous data reception, the user can retrieve the different elements of the protocol provided in the SIS individually (e.g. tags, keys, part of the DSM block), exploiting the way they are transmitted within the OSNMA fields, as defined in [AD.2]. For example, it is possible to retrieve the key by combining pages received from multiple satellites.

To be noted that, as specified in the Galileo OS Service Definition Document (Galileo OS SDD, Issue 1.1, May 2019), an authentication verification shall be performed only on navigation and OSNMA data for which a CRC checksum was successfully passed. The Galileo I/NAV CRC is described within [AD.1].

Additionally, several fields in the OSNMA data are reserved, as described in [AD.2]. The receiver shall be robust to any value being transmitted within the reserved fields.

3.1 Merkle Root Retrieval

In order to validate new public keys retrieved from the Signal In Space, the user must be in possession of the root of the Merkle tree and know which cryptographic function shall be used for the verification. This information can be loaded in the receiver after being retrieved from the GSC OSNMA server, as described in [AD.2]. The receiver shall be able to download new Merkle roots from the OSNMA Server during the OSNMA lifetime, following an eventual Merkle tree renewal. This renewal will be indicated through the use of an OSNMA alert message.

3.2 Public Key Retrieval

The receiver shall possess a public key, with its associated ID and signature algorithm. This information can be retrieved from:

- The GSC OSNMA server and loaded in the receiver. Details on the interface to this server can be found in [AD.2].
- The SIS, within the DSM-PKR message transmitted during public key renewal and revocation[1], as described in [AD.2]. The public key retrieved through the SIS shall then be authenticated, as per section 5.1. To be noted that Merkle tree renewal will induce a Public Key ID rollover. While retrieving the DSM-PKR, the sequencing of the DSM messages (described in [AD.2]) as well as their redundancy (i.e. same DSM ID block transmitted by several satellites) can be exploited. In addition, DSM blocks can be built using pages retrieved from different sub-frames. If the DSM-PKR message is not completed after one hour, the retrieved DSM blocks shall be discarded.
- A previously stored public key, provided that its applicability is verified as per section 4.1.1.2.

3.3 TESLA Root Key Retrieval

The TESLA root key shall be retrieved from the DSM-KROOT message transmitted by the SIS, as defined in [AD.2], and shall be verified as per section 5.2. Similarly to the DSM-PKR message, the

[1] Future updates of this documents might specify specific intervals at which DSM-PKR are transmitted.

defined transmission sequence of the DSM-KROOT message can be exploited to optimise its reception. If a DSM-KROOT message is not completed after 1 hour [TBC], the retrieved DSM blocks shall be discarded.

Instead of retrieving the TESLA root key from the SIS, the receiver can also use a previously stored TESLA root key, provided that its applicability is verified as per section 4.1.1.2.

Note that according to [AD.2], a DSM-PKR message can be alternated with a DSM-KROOT message, and two DSM-KROOT messages can also be transmitted alternatively, for example when a new chain is about to enter into force, as explained in section 4.2. Therefore, the receiver must be able to retrieve and store more than one DSM message in parallel.

3.4 TESLA Chain Keys Retrieval

TESLA chain keys are provided in the MACK message transmitted within the OSNMA message transmitted by the SIS, as defined in [AD.2]. The size of the TESLA chain key (l_K) is defined in the KS field of the DSM-KROOT message.

3.5 Navigation Data Retrieval

The navigation data shall be retrieved on a sub-frame basis, as a function of the ADKD type as per the descriptions provided in [AD.2].

In addition, the PRN of the satellite transmitting the authentication information, $\mathrm{PRN_A}$, to be used for the verifications in sections 5.5.3 and 5.5.5, shall be the one of the PRN code used for tracking, and shall not be retrieved from the navigation message.

3.6 Tags Retrieval

The tags and their associated Tag-Info are also provided in the MACK message transmitted in the SIS, as described in [AD.2]. Several tags can be sent within a MACK message. The number of tags per MACK message n_t can be computed as a function of the TESLA chain parameter, as defined in [AD.2]. Each tag is a truncation of a MAC generated with a given TESLA chain key, as further detailed in section 5.5.5.

4 OSNMA Workflow and Status Monitoring

4.1 *Workflow*

This section provides a description of the different operations that need to be performed in order to authenticate navigation data using the Galileo OSNMA protocol. The OSNMA workflow is divided in two steps:

- First, the initialisation, which consists in retrieving and verifying the public key and TESLA root key;
- Second, the processing of the authentication material to achieve navigation data authentication.

4.1.1 Initialisation

The OSNMA initialisation consists in the retrieval and verification of the public key and of the TESLA root key. As previously mentioned, this data can be either retrieved from the SIS and/or the GSC OSNMA server, referred as a factory start in the following section, or stored material can be used, referred as a warm start or hot start depending on the stored material.

4.1.1.1 Cold Start

In case the public key and TESLA root key in force are not available, the receiver shall first retrieve them, as illustrated in Figure 2. The public key can be retrieved directly from the GSC OSNMA server or from a DSM-PKR message. In this latter case, the public key must be verified making use of the Merkle root, as further detailed in section 5.1.

The verified public key can then be used to verify the TESLA root key and associated chain parameters, sent in the DSM-KROOT message, as per section 5.2. The verified TESLA root key can then be stored and used for the verification of the TESLA chain keys broadcast within the MACK message, as per section 5.4.

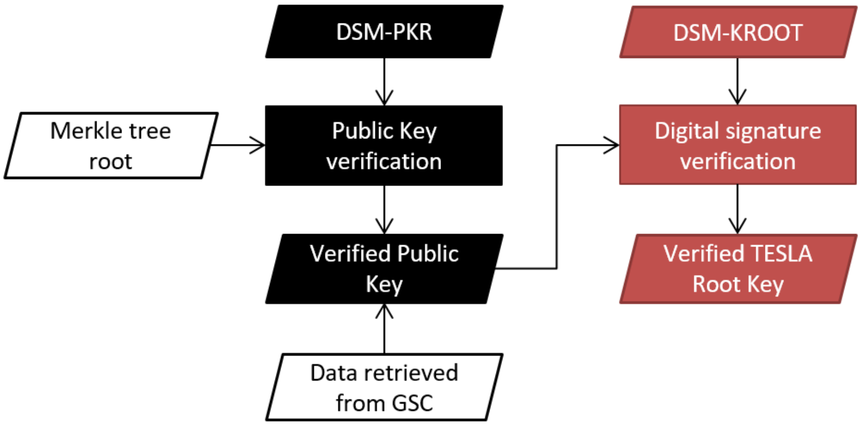


Figure 2. Verification of the public key and TESLA root key

4.1.1.2 Warm Start

If the receiver has stored a previously verified public key, and as long as it has been stored according to the requirements stated in section 2.3, the following steps shall be followed.

The receiver shall retrieve the DSM-KROOT message and verify that the public key ID (PKID) contained in the DSM-KROOT message corresponds to the one of the stored public key:

- If the PKID values are equal, the receiver can use the stored public key;
- If the values are different, the receiver shall discard the stored public key and retrieve a new one, from the GSC OSNMA server or from the DSM-PKR message. In the latter case, it shall also verify the public key as described in section 5.1.

If the PKID of the DSM-KROOT matches the one of the stored key, the receiver shall verify the DSM-KROOT with the stored key:

- If the verification fails, the receiver shall retrieve a new public key, from the GSC OSNMA server or from the DSM-PKR message. In the latter case, it shall also verify the public key as described in section 5.1.
- If the verification is successful, the receiver can use DSM-KROOT data (TESLA root key and chain parameters).

4.1.1.3 Hot Start

If, in addition of a stored public key, the receiver also possesses previously verified TESLA key and chain parameters, it does not have to retrieve the DSM-KROOT message. The receiver shall attempt to verify the received TESLA chain key with the stored key, being this either the TESLA root or a previously verified TESLA chain key, as per section 5.4.

- If the verification is successful, the receiver shall use the stored material for the data authentication.
- If the verification fails, the receiver shall retrieve the DSM-KROOT message and verify it with the stored public key, as per section 5.2.
    - If the verification is successful, the receiver can use the retrieved TESLA root key and KROOT parameters.
    - If the verification fails, the receiver shall retrieve a new public key, from the GSC OSNMA server or from the DSM-PKR message. In the latter case, it shall also verify the public key as described in section 5.1. Once this is done, it shall proceed to verify the retrieved DSM-KROOT message.

4.1.2 Processing of the Authentication Material

To authenticate a navigation data set, the user identifies the required tag and retrieves it, as explained in section 5.5.1. The ADKD type of the retrieved tag is verified against the ADKD sequence defined in the MAC look-up table (MACLT), as described in section 5.5.2. If some of these ADKDs are defined as flexible (i.e. they are not fixed for the given chain), their Tag-Info sections can be verified through the MACSEQ verification, as detailed in section 5.5.3. The TESLA key applicable for the tag verification is retrieved as per section 5.5.4 and verified as per section 5.4, with the verified TESLA root key or with a TESLA chain key previously verified. The verified TESLA chain key is then used for the verification of the tag, as per section 5.5.5. It can also be used for the verification of successive TESLA chain keys, acting recursively as new verified elements within the TESLA chain. The navigation data is authenticated as a result of successful tag verifications as per section 5.5.6.

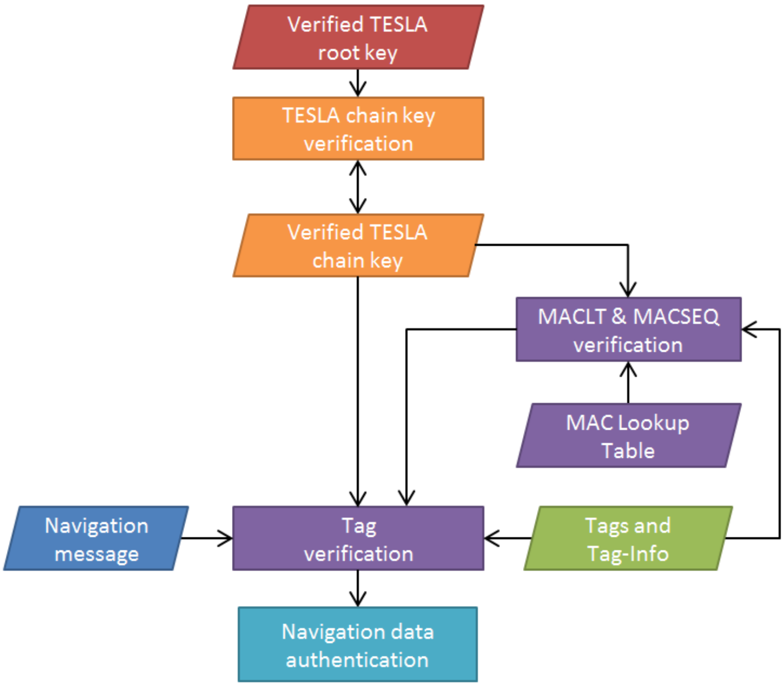


Figure 3. Verification of the TESLA chain key and tags

4.2 NMA Status

The result of the OSNMA authentication process is conditioned by the status of the NMA, which is reported in the NMAS field of the NMA Header. This value, which is the same for all satellites, shall be monitored by the receiver, and may require specific operations, as described below.

4.2.1 Monitoring of the NMA Status

The NMA status field can be set to three different values:

- "Test", to indicate that OSNMA is provided without any operational guarantees.
- "Operational", to indicate that OSNMA is provided according to the specifications.
- "Don't Use", to indicate that the receiver shall not perform navigation data authentication.

Before being interpreted, the NMA status shall be verified. The NMA status can be verified in two ways:

- In the tag verification, described in section 5.5. Thus, every time a tag is verified, the NMA status is verified as well.
- In the TESLA root key signature verification, described in section 5.2. As part of this verification, the Chain ID and CPKS flags, which may indicate that the cryptographic material is being renewed and revoked as defined in [AD.2], are also authenticated.

The user processing OSNMA data is required to verify the NMA status through the retrieval and verification of the TESLA root key signature at least once every hour [TBC]. In the case the NMA status is set to "Don't Use", the retrieval and verification of the TESLA root key signature, authenticating as well the Chain ID and CPKS flags, shall be immediately performed.

These flags may indicate that the cryptographic material relative to the TESLA chain and to the public key are being revoked or renewed, as detailed in [AD.2].

4.2.2 "Test" and "Operational" Status

The CPKS field can be set to five different values that shall be interpreted in combination with the NMA status. The possible combinations of CPKS and NMA status set to "Operational" or "Test", together with their interpretations and the operations the receiver shall perform, are presented in Table 2. This table shall be read together with the description of the renewal and revocation processes provided in [AD.2].

The cryptographic material defined in the table as ‘current’ corresponds to that applicable for the tag verification, as defined by the CID field for the TESLA chain in force (indicated in the NMA header) and by the PKID field associated to the public key in force (indicated in the DSM-KROOT), reported in the third and fourth columns. More detailed information of the renewal and revocation of the OSNMA cryptographic material can be found in [AD.2].

NMA status	CPKS	PKID	CID	Required operations
Operational/ Test	Nominal	*p*	*i*	The navigation data shall be authenticated using the current TESLA chain material (CID = *i*). The TESLA chain material consists of the TESLA root key and the TESLA chain parameters.
Operational/ Test	End of Chain (EOC)	*p*	*i*	The current TESLA chain (CID = *i*) is coming to an end. The receiver shall retrieve the new TESLA chain material (CID = *i’*) sent in the DSM-KROOT message and verify it with the public key (PKID=*p*) as per section 5.2. This material (CID = *i’*) will be used to authenticate the navigation data from its time of applicability, as defined in [AD.2]. Before that, while the CPKS flag is set to ‘EOC’, the navigation data shall still be authenticated using the current TESLA chain material (CID= *i*).
Operational/ Test	Chain Revoked (CREV)	*p*	*i'*	A previous TESLA chain (CID = *i*) has been revoked and shall be discarded. If the receiver does not possess the current TESLA chain material (CID = *i’*), it shall retrieve it in the current DSM-KROOT message and verify it with the public key (PKID=*p*) as per section 5.2. The navigation data shall be authenticated using the current TESLA chain material (CID = *i’*).
Operational/ Test	New Public Key (NPK)	*p*	*i*	The public key (PKID=*p*) is being renewed. The receiver shall retrieve the new public key (PKID=*p’*), either from the DSM-PKR message and verify it as per section 5.1, or from the GSC OSNMA server. When a DSM-KROOT message is retrieved, it shall be verified with the current public key (PKID=*p*) and the navigation data shall be authenticated using the current TESLA chain material (CID = *i*).
		p’	*i*	The public key (PKID=*p*) has been renewed. The receiver shall retrieve the new public key (PKID=*p’*), either from the DSM-PKR message or from the GSC OSNMA server. In the first case, the receiver shall verify DSM-PKR message as per section 5.1. The receiver shall discard the previous public key (PKID=*p*) and shall verify the retrieved DSM-KROOT message with the current public key (PKID = *p’*) as per section 5.2. The receiver can then authenticate the navigation data using the current TESLA chain material (CID = *i*).
Operational/ Test	Public Key Revoked (PKREV)	*p'*	*i'*	The previous public key *p* has been revoked and shall be discarded. The TESLA chain material associated to this public key (CID = *i*) shall also be discarded. If the receiver does not possess the current public key (PKID = *p’*), it shall retrieve it in the current DSM-PKR message and verify it as per section 5.1, or retrieve it from the GSC OSNMA server. If the receiver does not possess the current TESLA chain material (CID = *i’*) associated with (PKID = *p’*), it shall retrieve it in the current DSM-KROOT message and verify it as per section 5.2. The navigation data shall be authenticated using the current TESLA chain material (CID = *i’*).

Table 2. Operations associated with the NMA status “Operational/Test”

Note that while the NMA status is set to “Operational” or “Test”, the continuity of the navigation data authentication process is maintained. In addition, once a key with a certain PKID is in force, any key with a lower PKID is considered not in use by the system (as per [AD.2]). Therefore, DSM messages with a PKID lower than that in force shall be rejected.

Note that if a DSM-KROOT message is to be verified with a public key *p’* that the receiver does not possess, and such that PKID(*p’*) > PKID(*p*), *p* being the trusted public key stored in the receiver:

- If the CPKS flag is to NPK or PKREV, the receiver should get the new DSM-PKR message for *p’*, authenticate *p’* and replace the public key.
- If the CPKS flag is set to “Nominal”, the receiver may have been switched off for a long time and thus may have missed the PK update. The receiver may either get *p’* from the GSC OSNMA server, or get a DSM-PKR message transmitted by the SIS and verify it.

4.2.3 “Don’t Use” Status

When the NMA Status is verified to be set to “Don’t use”, the receiver shall not perform navigation data authentication. The retrieval of new OSNMA material can be carried out, as described in Table 3. This table shall be read together with the description of the renewal and revocation processes provided in [AD.2].

NMA status	CPKS	PKID	CID	Required operations
Don’t use	Nominal	*p*	*i*	The receiver shall not perform navigation data authentication.
Don’t use	Chain Revoked (CREV)	*p*	*i*	The current TESLA chain (CID = *i*) is revoked and the material associated to this chain shall be discarded. The receiver shall retrieve the new TESLA chain material (CID = *i’*) sent in a new DSM-KROOT message and verify it with the current public key (PKID=*p*), as per section 5.2. The receiver shall not perform navigation data authentication.
Don’t use	Public Key Revoked (PKREV)	*p'*	*i*	The previous public key (PKID=*p*) is revoked and shall be discarded, along with its associated TESLA chain material (CID = *i*). The receiver shall retrieve the current public key (PKID = *p’*), either from the GSC OSNMA server or in the DSM-PKR message and, in the latter case, verify it as per section 5.1.The receiver shall also retrieve the new TESLA chain material (CID = *i’*) and verify it as per section 5.2. The receiver shall not perform navigation data authentication.

Table 3. Operations associated with the NMA status “Don’t Use”

5 Overview of Cryptographic Operations

This section presents an overview of the cryptographic operations that shall be performed in order to authenticate navigation data using Galileo OSNMA data. Further details on these operations are provided in [AD.2].

5.1 Verification of the Public Key Retrieved from the SIS

The public key retrieved from the SIS within the DSM-PKR message (as explained in section 3.2) shall be authenticated against the stored Merkle root using the Secure Hash Algorithm (SHA). This procedure is summarised in Table 4.

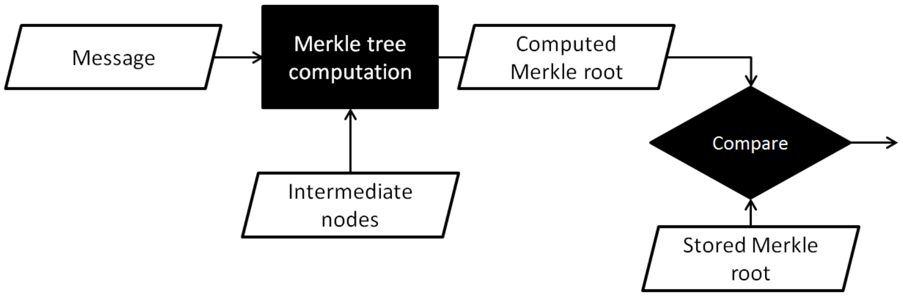
Public Key verification	
Inputs	• Message obtained by concatenating the public key type (NPKT), the public key ID (NPKID) and the public key (NPK) itself, as described in [AD.2] • Intermediate nodes received in the DSM-PKR message [AD.2] • Merkle root, stored in the receiver.
Cryptographic functions	Hash function used to build the Merkle tree: • SHA-256 [6],[10] • SHA3-256 [6][2]
Schematic	 Message Merkle tree computation Computed Merkle root Compare Intermediate nodes Stored Merkle root
Output	• If the computed tree root matches the stored root, the public key is verified. The public key can be used to verify the TESLA root key, as per section 5.2. • Otherwise, the verification fails. The public key shall be discarded. The navigation data cannot be authenticated. The event shall be logged and reported at application level.

Table 4. Description of the public key verification process

As explained in [AD.2], based on the message ID (MID) provided in the DSM-PKR message, the message is associated to a leaf m_i of the Merkle tree, and the associated intermediate nodes (ITN) $x_{j,i}$ can then be used to compute the Merkle root $x_{4,0}$, using the relationship between nodes described in [AD.2].

An example of the tree root computation is provided in Figure 4, assuming that the MID = 0, meaning that the message is associated to the first leaf $m_0$and that the provided intermediate nodes are ($x_{0,1}$, $x_{1,1}$, $x_{2,1}$, $x_{3,1}$), as defined in [AD.2].

[2] The system might use in future SHA3-256 for the tree generation.

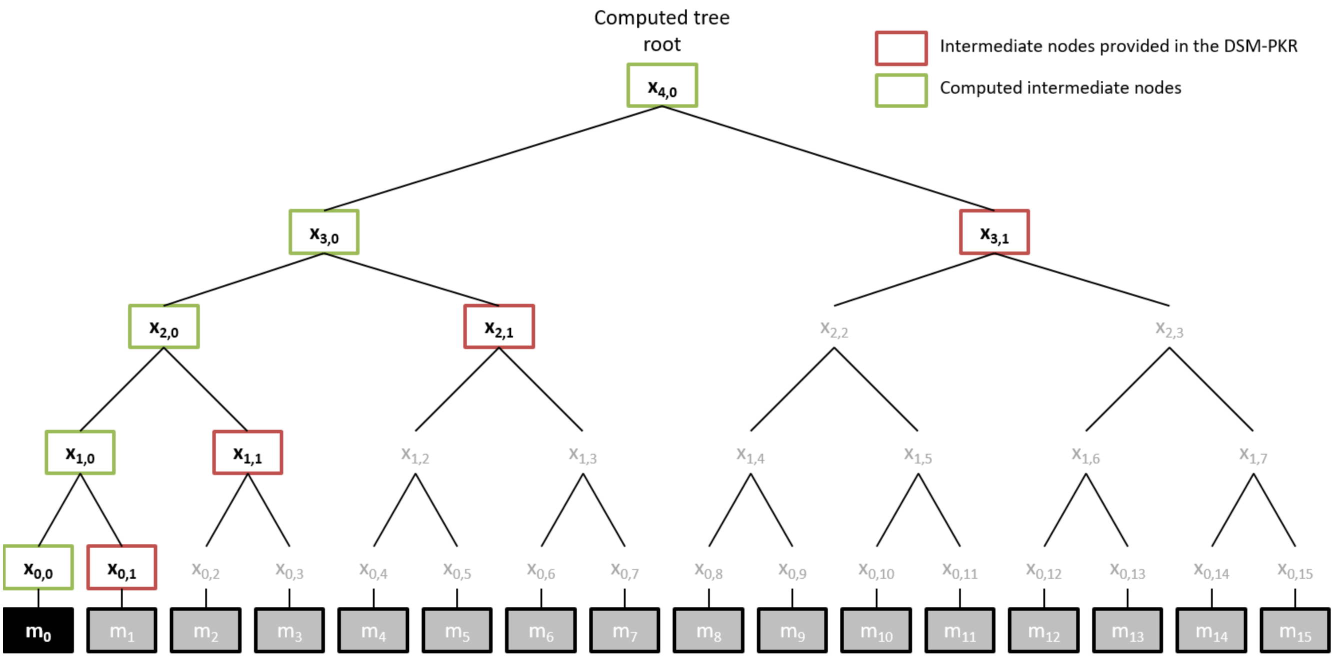


Figure 4. Verification of the public key through the use of Merkle tree

5.2 *Verification of the TESLA Root Key*

The TESLA root key, chain cryptographic functions, TESLA chain key length, tag length, and other TESLA chain parameters are received in the DSM-KROOT message and verified using the Elliptic Curve Digital Signature Algorithm (ECDSA), as summarised in Table 5.

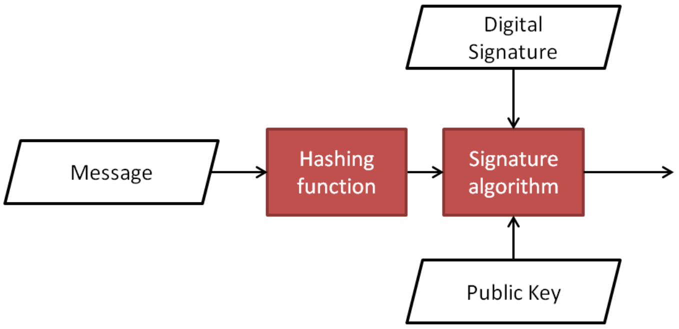
Root Key verification	
Inputs	• Message produced by concatenating some of the fields from the DSM-KROOT message, as described in [AD.2]. • Digital Signature (DS) obtained within the DSM-KROOT message, as described in [AD.2] • Public key (PK), retrieved as per section 3.2.
Cryptographic functions	Signature algorithm / Hash function combinations: • ECDSA P-256 / SHA-256 [4] • ECDSA P-521 / SHA-512 [4] The applicable function is specified in the New Public Key Type (NPKT) field of the DSM-PKR message [AD.2] and is provided on the GSC OSNMA server in case the public key is retrieved there.
Schematic	 Digital Signature Message Hashing function Signature algorithm Public Key
Output	• If the signature algorithm confirms that the received digital signature is valid, the verification is successful. The NMA Status, chain ID and CPKS fields are verified, as are the TESLA root key and the TESLA chain parameters. These parameters can be used to verify the TESLA chain keys (as per section 5.4). • Otherwise, the verification fails. The TESLA root key, TESLA chain parameters and flags are not verified and shall be discarded. The navigation data cannot be authenticated. The event shall be logged and reported at application level.

Table 5. Description of the TESLA root key verification process

5.3 Determination of the GST Sub-frame

As further detailed in sections 5.4 and 5.5, the GST of the sub-frame GST_{SF} in which the TESLA chain key and the tags are retrieved is needed for their verification. GST_{SF} can be computed in one of two ways, as per the next sub-sections.

5.3.1 GST Retrieval and Verification from the SIS

The GST of the sub-frame, GST_{SF}, can be derived from the GST^{SIS} retrieved from the SIS, either from the I/NAV word type 5 or from the word type 0 (Spare Word). For E1, this is the E1 sub-frame start minus 1 second. The Secondary Synchronisation Pattern (SSP) may also be exploited [AD.1].

The retrieved GST^{SIS} value shall be verified against the receiver local realisation GST^{Rx}, to ensure that:

$$|\mathrm{GST}^{SIS} - \mathrm{GST}^{Rx}| < \frac{T_L}{2} \qquad \text{Eq. 1}$$

Where T_L is the receiver time synchronisation requirement defined within the section 2.1. If this condition is not verified, the user shall not process OSNMA data.

The GST of the sub-frame, GST_{SF}, used in the TESLA chain key and tag verifications, can then be computed as:

$$\mathrm{GST}_{SF} = \mathrm{GST}_0 + 30 \cdot \left\lfloor \frac{\mathrm{GST}^{SIS} - \mathrm{GST}_0}{30} \right\rfloor \qquad \text{Eq. 2}$$

Where:

- $\lfloor n \rfloor$ is the floor operator, indicating the greatest integer less than or equal to *n*;
- GST_0 is the time of applicability of the TESLA chain in force, as per [AD.2], expressed in seconds.

5.3.2 GST Sub-frame Propagation

The GST of the sub-frame can also be propagated internally by the receiver, based on the fixed structure of the OSNMA data. In that case the GST_{SF} will be incremented by 30 seconds every time a new sub-frame is being retrieved. The initial value of GST_{SF} to be propagated shall be computed from the GST^{SIS} retrieved from the SIS and verified, as per section 5.3.1

5.4 *Verification of the TESLA Chain Key*

5.4.1 TESLA Chain Properties

As explained in [AD.2], the keys used for the verification of the tags are part of a one-way chain. Keys from the same chain are related in such a way that each key can be re-constructed by applying a function F to the next transmitted key, as in the following equation:

$$K_I = F(K_{I+1}) = trunc\left(l_k, \mathrm{hash}_{\mathrm{chain}}\left(K_{I+1} \| \mathrm{GST}_{SF,I} \| \alpha\right)\right) \qquad \text{Eq. 3}$$

Where:

- I is the index of the key in the TESLA chain, computed as per [AD.2];
- l_k is the TESLA chain key size defined [AD.2];
- $\mathrm{hash}_{\mathrm{chain}}$ is the specific hash function used for the TESLA chain as indicated in the HF field defined in [AD.2][3];
- $GST_{SF,I}$ is the Galileo System Time at the start of the 30-seconds sub-frame in which the TESLA chain key K_I is transmitted, computed as per section 5.3 and in the format WN (12bit) and TOW (20bits);
- α is the unpredictable chain pattern defined in [AD.2];
- $trunc(L, I)$ is the truncation function retaining the L most significant bits (MSB) of the input I.

5.4.2 Number of Recursive Operations

Given the previously mentioned properties of the TESLA chain, a TESLA chain key K_I can be verified against a previously authenticated key belonging to the same TESLA chain, K_J, where J <I, by recursively applying the function F. The number of times the function shall be applied to perform the verification is equal to the difference between the positions of the two keys in the chain, such that:

$$K_J = F^{I-J}(K_I) \qquad \text{Eq. 4}$$

Where:

- I is the index of the key to be verified in the TESLA chain, computed as per [AD.2]. The GST_{SF} used in the calculation is derived as per section 5.3
- J is the index of the previously verified key in the TESLA chain, computed as per [AD.2]. In the case the TESLA root key K_0 is used, $J = 0$.

This concept is further illustrated in Figure 5.

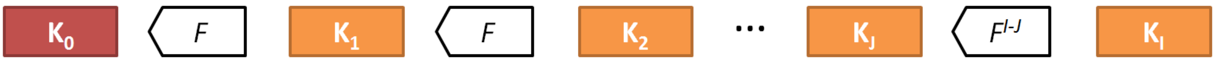


Figure 5. TESLA chain key

The function F shall be applied recursively exactly $(I - J)$ times to consider the verification successful.

5.4.3 Verification Process

The TESLA chain key verification is performed by means of a SHA cryptographic function, as summarised in Table 6.

TESLA key verification	
Inputs	• Message produced by concatenating the TESLA chain key provided in the MACK message, the Galileo System time *GST*$_{SF}$ computed as per section 5.3 and expressed in the format WN (12bit) and TOW (20bits), and the unpredictable chain pattern *α*, as defined in [AD.2] • A verified TESLA key: the TESLA root key (KROOT) verified as per section 5.2 or a previously authenticated TESLA key from the same chain.

[3] Note that $\mathrm{hash}_{\mathrm{chain}}$ takes as input a message that fits an integer number of bytes, and if that is not the case, it needs to be zero-padded, as explained in [AD.2].

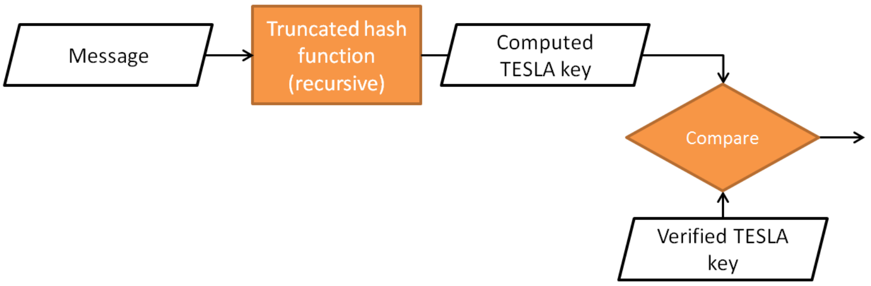
TESLA key verification	
Cryptographic functions	Hash functions: • SHA-256 [5] • SHA3-256 [6] The applicable function is specified in the Hash Function (HF) field of the DSM-KROOT message [AD.2].
Schematic	 Message Truncated hash function (recursive) Computed TESLA key Compare Verified TESLA key
Output	• If the computed TESLA key matches the verified TESLA key, the verification is successful. The verified TESLA chain key can be used for the verification of the tag as per section 5.5 and for the verification of successive TESLA chain keys. • Otherwise, the verification fails. The TESLA chain key is not verified and shall be discarded. The event shall be logged and reported at application level.

Table 6. Description of the TESLA chain key verification process

5.5 Verification of the Tag and Authentication of Navigation Data

As mentioned in section 4.1.2, the authentication of the navigation data is carried out in several steps:

- The user identifies the tag required to authenticate the navigation data and retrieves it, as per section 5.5.1;
- The user verifies the ADKD of the retrieved tag against the fixed sequence defined by the MACLT, as per sections 5.5.2 and 5.5.3;
- The TESLA chain key, applicable for the verification, is retrieved and verified as per section 5.5.4;
- The verification of the tags and the authentication of the navigation data are then performed as per sections 5.5.5 and 5.5.6.

Based on the successful verification of the tags, the data navigation can be authenticated as per section 5.5.6.

5.5.1 Identification of the Applicable Tag

The navigation data retrieved in sub-frame transmitted at time GST_{SF} is authenticated by the tags transmitted in the next sub-frame transmitted at time (GST_{SF} + 30 sec). Different parts of the navigation data are authenticated by different tags, as specified by the tag ADKD type, for different satellites, as specified by the tag $\mathrm{PRN_A}$ field.

Provided that all the prescriptions described here and in [AD.2] are respected, and in particular the fixed link between a tag and the data it authenticates (as discussed above), users can exploit all the properties of the Galileo I/NAV message as they are described in [AD.1] in order to optimise the performance. An example is the repetition of clock and ephemeris data (unambiguously identified by an $\mathrm{IOD_{nav}}$) and including Reed-Solomon information provided within word types 17-20. In any case, a tag received in a certain sub-frame shall never be used to authenticate navigation data retrieved after the transmission of the last bit of the last Tag-Info field received in that sub-frame.

To be noted, that depending on its needs, a receiver may process all transmitted tags or only a subset of them. As an example, a user can decide to process only tags with an ADKD = 0 in order to authenticate only I/NAV ephemeris, clock and satellite health data.

5.5.2 Tags Sequence Verification

In order to verify a tag, the receiver shall first verify that its associated ADKD corresponds to the one predefined in the MAC Look-up Table [AD.2]. The entry value to the MAC Look-up Table is provided by the MACLT field of the DSM-KROOT message and is fixed for a given TESLA chain.

The MAC Look-up Table can define some slots in the tag sequence as flexible (‘FLX’), which means that the ADKD values of these tags are not fixed and can be allocated dynamically. The Tag-Info field of these tags is then authenticated using the MACSEQ, as described in section 5.5.3.

If the ADKD type of a received tag differs from the one defined in the MAC Look-up Table sequence, the tag shall be discarded.

5.5.3 Flexible Tags Sequence Verification

To exploit the tags defined as flexible in the MAC Look-up Table, the MACSEQ verification shall be performed to verify the authenticity of the associated Tag-Info sections. The MACSEQ is verified with the same key and MAC function as the $\mathrm{Tag_0}$ in the same MACK message [AD.2], as summarised in Table 7.

MACSEQ verification	
Inputs	• Message produced by concatenating fields as described in [AD.2] • MACSEQ received, as described in [AD.2] • Applicable TESLA chain key, as specified in 5.5.4, verified as described in section 5.4

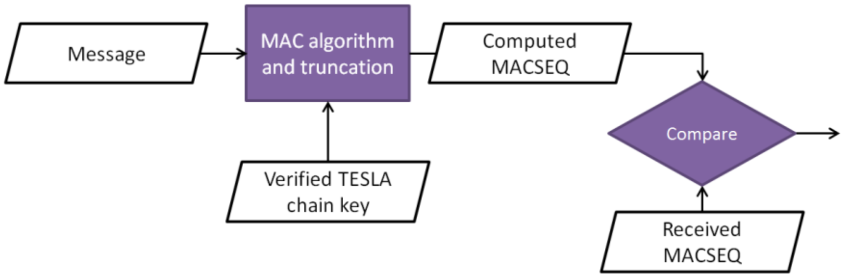
MACSEQ verification	
Cryptographic functions	MACSEQ verification algorithm: • HMAC-SHA-256 [7] • CMAC-AES [8], [9] The applicable function is specified in the MAC Function (MF) field of the DSM-KROOT message [AD.2].
Schematic	 Message MAC algorithm and truncation Computed MACSEQ Compare Verified TESLA chain key Received MACSEQ
Output	• If the locally computed MACSEQ matches the received MACSEQ, the verification is successful. The FLEX tags sequence is verified. • Otherwise, the verification fails. The related flexible tags shall be discarded. The event shall be logged and reported at application level.

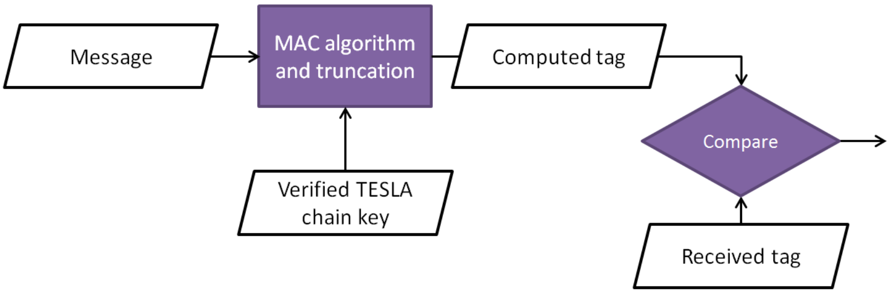
Table 7. Description of the MACSEQ verification process

5.5.4 Identification of the Applicable TESLA Chain Key

A tag retrieved in a certain MACK message has to be verified with the TESLA chain key broadcast in the next MACK message, as specified in [AD.2]. It is important to note that, given the properties of the TESLA chain keys mentioned in section 5.4.1, even if the receiver did not retrieve the key required for the verification from the SIS, it can compute this key from any retrieved key which has a subsequent position in the chain. A TESLA chain key shall be verified as per section 5.4 before being used for the tag verification.

5.5.5 Tag Verification

The tag verification is performed to verify the authenticity of the navigation data it is associated to. A summary of the tag verification scheme is provided in Table 8.

Tag verification	
Inputs	• Message produced by concatenating fields, as described in [AD.2]. • Tag received in the MACK message, as described in [AD.2] • Applicable TESLA chain key, as specified in 5.5.4, verified as in section 5.4
Cryptographic functions	MAC verification algorithm: • HMAC-SHA-256 [7] • CMAC-AES [8], [9] The applicable function is specified in the MAC Function (MF) field of the DSM-KROOT message [AD.2].
Schematic	 Message MAC algorithm and truncation Computed tag Compare Verified TESLA chain key Received tag

Tag verification	
Output	• If the locally computed tag matches the received tag, the verification is successful. • Otherwise, the verification fails. The tag shall be discarded. The event shall be logged and reported at application level.

Table 8. Description of the tag verification process

5.5.6 Navigation Data Authentication

Navigation data authentication is achieved after verifying a minimum number of tag bits, L_t^{min}, associated to the same navigation data set. The value of the minimum equivalent tag length L_t^{min} is currently equal to 80 bits. The minimum equivalent tag length L_t^{min} may be reduced in future updates of this document and therefore this parameter should be configurable in the receiver.

If the tag being used for the authentication has a length l_t such that $l_t \geq L_t^{min}$, then the navigation data is authenticated if the tag is verified as per section 5.5.5.

If the tag length l_t is such that $l_t < L_t^{min}$, tag accumulation shall be performed. The steps below shall be followed:

- The receiver shall accumulate N_t tags, such that $l_t \cdot N_t \geq L_t^{min}$. All tags to be accumulated must correspond to the same data to be authenticated. For elements covered by IOD_{nav}, the user can take advantage of the I/NAV message structure, as presented in section 5.5.1. The receiver shall also register the sub-frame time GST_{SF} (computed as per section 5.3) in which the tags are retrieved.
- The receiver shall then ensure that the registered GST_{SF} values are coherent with the sub-frames in which the tags are retrieved, e.g. two tags retrieved in two consecutive sub-frames will have an offset of 30 seconds between their GST_{SF}.
- Once N_t tags have been retrieved and the coherency between the registered GST_{SF} is verified, the receiver shall verify the tags using their associated and previously authenticated TESLA chain keys, as described in section 5.5.5.
- If one of the verifications fails, all the tags shall be discarded, the navigation data is not authenticated. If all the tags are verified, the navigation data is authenticated.

If a tag authentication fails, new navigation data shall be retrieved and tag accumulation restarted.

References

[1] International Organization for Standardization, “ISO/IEC 29192-7:2019 Information security — Lightweight cryptography — Part 7: Broadcast authentication protocols.” 2019.

[2] Method and system to optimise the authentication of radionavigation signals, Patent application PCT/EP2015/056120, 23/03/2015, European Union represented by European Commission

[3] Digitally-signed satellite radio-navigation signals, Patent application PCT/EP2014/064285, 04/07/2014, European Union represented by European Commission

[4] National Institute of Standards and Technology, “FIPS PUB 186-4: Digital Signature Standard (DSS).” 2013.

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

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

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

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

[9] 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.

[10] National Institute of Standards and Technology, “NIST Special Publication 800-208: Recommendation for Stateful Hash-Based Signature Schemes.” 2020.

List of abbreviations

ADKD	Authentication Data & Key Delay
AES	Advanced Encryption Standard
BID	Block ID
CID	Chain ID
CMAC	Cipher-based Message Authentication Code
CPKS	Chain and Public Key Status
CRC	Cyclic Redundancy Check
CREV	Chain Revoked
DSM	Digital Signature Message
DSM-KROOT	DSM for a KROOT
DSM-PKR	DSM for a PKR
ECDSA	Elliptic Curve Digital Signature Algorithm
EOC	End Of Chain
GSC	European GNSS Service Centre
GST	Galileo System Time
HF	Hash Function
HKROOT	Header and KROOT
HMAC	Hash-based Message Authentication Code
ICD	Interface Control Document
IOD	Issue of Data
ITN	Intermediate Tree Node
KROOT	Root Key
MAC	Message Authentication Code
MACK	MAC and Key
MACLT	MAC Look-up Table
MACSEQ	MAC Sequence
MF	MAC Function
MID	Message ID
MSB	Most Significant Bit
NB	Number of Blocks
NMA	Navigation Message Authentication
NPK	New Public Key
NPKID	New Public Key ID
NPKT	New Public Key Type
OS	Open Service
OSNMA	Open Service Navigation Message Authentication
PK	Public Key
PKID	Public Key ID
PKR	Public Key Renewal
PKREV	Public Key Revocation
PRN	Pseudo Random Noise
PVT	Position Velocity Time
SHA	Secure Hash Algorithm
SIS	Signal In Space

TESLA	Timed Efficient Stream Loss-Tolerant Authentication
TOW	Time of Week
WN	Week Number

Annex 1. Test Vectors

This Annex provides a set of OSNMA test vectors with the purpose of supporting developers in their implementation of the OSNMA protocol with regards to data parsing and verifications using cryptographic operations.

A1.1 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 in hexadecimal and binary values are always in big-endian bytes 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.

A1.2 OSNMA Configuration

The test vectors correspond to the OSNMA configuration 7, presented in Annex 2 and summarised in the following table.

Parameters	**Settings**
Tag size	40 bits
Key size	128 bits
Digital Signature Algorithm	ECDSA P-256
Hash Function	SHA-256
MAC Function	HMAC-SHA-256
MACLT	33

A1.3 NMA Header

This section reports the NMA Header to be used for the verifications presented in the following sections.

NMA Header = `0b01010010` (8 bits)

As per [AD.2], the header can be interpreted as follows.

Field	**Field value**	**Interpretation**
NMAS	`0b01`	Test
CID	`0b01`	1
CPKS	`0b001`	Nominal
Reserved	`0b0`	N/A

The header indicates that:

- the OSNMA is in Test mode (i.e. provided without operational guarantees)
- the TESLA chain in force is the one with ID 1

- the public and chain in force are nominal (i.e. no renewal or revocation processes are taking place).

A1.4 DSM-KROOT

The DSM-KROOT message is reported below.

DSM-KROOT =
`0x2150492104790025D3964DA3A2540A4830D139B710A4951D73C19DA22D3612E32DD`
`C522FD248C7EA8DD271C757A35039F810405BDDE0528FFE261389A1643B879E1BDCB8`
`ADB529333B42D6C387E41EB7DF91AE20889BC37CCE7B86BE3C023AFCD8D6E7C0EDC67`
`D83` (832 bits)

A1.4.1 DSM-KROOT interpretation

The following chain parameters are extracted from the first 104 bits of the DSM-KROOT message.

Field	Field value	Interpretation
NB_{DK}	0x2	8 blocks
PKID	0x1	1
CIDKR	0b01	1
Rsvd1	0b01	N/A
HF	0b00	SHA-256
MF	0b00	HMAC-SHA-256
KS	0x4	128 bits
TS	0x9	40 bits
MACLT	0x21	33
Rsvd	0x0	N/A
WN_K	0x479	1145
$TOWH_K$	0x00	0 hours
α	0x25D3964DA3A2	0x25D3964DA3A2

The remaining bits of DSM-KROOT contain the KROOT itself, its digital signature (DS) and a padding sequence (P_{DK}).

KROOT = `0x540A4830D139B710A4951D73C19DA22D` (128 bits)

DS =
`0x3612E32DDC522FD248C7EA8DD271C757A35039F810405BDDE0528FFE261389A1643`
`B879E1BDCB8ADB529333B42D6C387E41EB7DF91AE20889BC37CCE7B86BE3C` (512 bits)

P_{DK} = `0x023AFCD8D6E7C0EDC67D83` (88 bits)

A1.4.2 DSM-KROOT verification

In order to verify the DSM-KROOT content, the message *M* is produced by concatenating some of the DSM-KROOT fields, as described in [AD.2]. The message to be signed is:

M = `0x525049210479002 5D3964DA3A2540A4830D139B710A4951D73C19DA22D` (232 bits)

The digital signature can be verified with the following public key, provided here in PEM format:

```
-----BEGIN EC PARAMETERS-----

BggqhkjOPQMBBw==

-----END EC PARAMETERS-----

-----BEGIN PUBLIC KEY-----

MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAE+Q2wvmvfdQg1sQF6OmCEy8skCSiu79vBn
RrKmaPpCJnaMOOvm26Us6ELhebL+q75MAyWAXJjlyRZZwp68gSAHw==

-----END PUBLIC KEY-----
```

The corresponding hexadecimal value of the public key is:

`0x04F90DB0BE6BDF750835B1017A3A6084CBCB240928AEEFDBC19D1ACA99A3E90899D`
`A30E3AF9B6E94B3A10B85E6CBFAAEF9300C96017263972459670A7AF204801F` (520 bits)

Note that the public key is broadcast via SIS in compressed ECDSA format, as provided in section A1.7.1.

A1.4.3. P_{DK} verification

This section describes the verification of the DSM-KROOT padding bits, P_{DK}. As this verification is not mandatory, it is not covered in the OSNMA Receiver Guidelines. The description below is provided for the sake of completeness and in the interest of the developer.

As per [AD.2], the message hashed for the verification is composed of the concatenation of the message *M* used for the DSM-KROOT verification and of the digital signature DS:

`0x5250492104790025D3964DA3A2540A4830D139B710A4951D73C19DA22D3612E32DD`
`C522FD248C7EA8DD271C757A35039F810405BDDE0528FFE261389A1643B879E1BDCB8`
`ADB529333B42D6C387E41EB7DF91AE20889BC37CCE7B86BE3C` (744 bits)

The message is then hashed with the SHA-256 function, leading to:

`0x023AFCD8D6E7C0EDC67D83857D956FDFFB070F728BC9CE056281980618C12015`
(256 bits)

This result is then truncated to the length of P_{DK} and compared bit-by-bit to the received field.

`0x023AFCD8D6E7C0EDC67D83` (88 bits)

A1.5. TESLA Chain Key

A1.5.1. TESLA Chain Key Interpretation

This section reports some of the TESLA chain keys and their associated index in the chain. It shall be noted that the key with index 0 corresponds to the root key transmitted over the DSM-KROOT section, as explained in section A1.4. The WN and TOW corresponding to the GST_{SF} of the sub-frame in which the TESLA chain key is *transmitted* is also reported.

Key Index	WN	TOW	Key (128 bits)
2879	1145	86340	0xDA7A30B12CF716B00BA31C6D9B2D21DA
2878	1145	86310	0x8AC8F29832EE2EB6C6CCF08F6BD416FC

Key Index	WN	TOW	Key (128 bits)
…	…	…	…
1441	1145	43200	`0x1256B87E98288C7657ACEB9E0291F523`
1440	1145	43170	`0x281A4ED883D8CED907A10EDAED595A41`
…			
12	1145	330	`0xD7DEF915D2863BDEA81A9E2480FD4662`
11	1145	300	`0xE41CD213C9FE2D2E5B4127857FE3912C`
…	…	…	…
3	1145	60	`0x8A50D8884FD0A6298B380EBDEA7C45F2`
2	1145	30	`0x4235FF797019E2EFD3CB72780E861FED`
1	1145	0	`0x17B98FD42A4AFD0EA36D1DA2DE406B93`
0 (KROOT)	1144	604770	`0x540A4830D139B710A4951D73C19DA22D`

Note that as per [AD.2], the GST_{SF} associated to the KROOT is GST_0 – 30 seconds, where GST_0 is the time provided in the DSM-KROOT, in the form of WN_K and $TOWH_K$ (see section A1.4).

A1.5.2. TESLA Chain Key Verification

This section describes the verification of the second key of the TESLA chain, K_2, against the root key, K_0.

As a first step, K_2 is used to reconstruct K_1, the previous key in the chain. The message to hash for this step is the concatenation of K_2, GST_{SF}, and α. Note that, as per [AD.2], GST_{SF} corresponds to the time of the sub-frame in which the key being computed, K_1, was transmitted (here WN = 1145 and TOW = 0 sec).

`0x4235FF797019E2EFD3CB72780E861FED4790000025D3964DA3A2` (208 bits)

Note that as the length of the message fits an integer number of bytes, no padding is needed. The message is then hashed with the HF function (SHA-256) and the following hash is obtained:

`0x17B98FD42A4AFD0EA36D1DA2DE406B93B069B4565EC6DDD47953C94249B669BA`
(256 bits)

The first key of the chain, K_1, is obtained by truncation, retaining the first KS = 128 bits of the hash:

`0x17B98FD42A4AFD0EA36D1DA2DE406B93` (128 bits)

Another step has to be performed to generate the local replica of the KROOT, K_0. This time, the message to hash is the concatenation of K_1, GST_{SF} and α. GST_{SF} is the time associated to K_0, corresponding to WN = 1144 and TOW = 604770 sec.

`0x17B98FD42A4AFD0EA36D1DA2DE406B9347893A6225D3964DA3A2` (208 bits)

The message is hashed and the following hash is obtained:

`0x540A4830D139B710A4951D73C19DA22D66BC5AED3B67CC874E9D5BCBE1E13CF5`
(256 bits)

K_0 is obtained after truncation:

`0x540A4830D139B710A4951D73C19DA22D` (128 bits)

The computed K_0 local replica can now be verified against the root key obtained from the DSM-KROOT, performing a bit-by-bit comparison. Note that following these steps, K_2 is verified. This implies that the following key, K_3, can be verified against K_2, without having to hash back to K_0.

A1.6. Tags

A1.6.1. MACK Message Interpretation

The tags are transmitted within the MACK message. The MACK messages transmitted by the satellite E01 at WN = 1145, TOW = 0 seconds and at WN = 1145, TOW = 30 sec are reported below.

`0xE094B3FBA533A6A6B55DFED505055EBEC4E3A8FF40B580AEB151190578A85B87930`
`1C6B3E3315F471B0517B98FD42A4AFD0EA36D1DA2DE406B930000` (480 bits)

`0x25CAEECF3EEDC68A2BA7BD7A1B05354C3819942405FF5C3DDA4801C61D0C9AB88E1`
`A0674FEF6A5B221C14235FF797019E2EFD3CB72780E861FED0000` (480 bits)

Six tags can be extracted from each of this MACK message, they are reported in the following table along with their position within the MACK message (CTR field), their associated PRN_D and ADKD values.

TOW	CTR	Tag	PRN_D	ADKD
0	1 (Tag_0)	`0xE094B3FBA5`	1	0
0	2	`0xA6B55DFED5`	5 (0x05)	0 (0x0)
0	3	`0x5EBEC4E3A8`	255 (0xFF)	4 (0x4)
0	4	`0xB580AEB151`	25 (0x19)	0 (0x0)
0	5	`0x78A85B8793`	1 (0x01)	12 (0xC)
0	6	`0xB3E3315F47`	27 (0x1B)	0 (0x0)
30	1 (Tag_0)	`0x25CAEECF3E`	1	0
30	2	`0x8A2BA7BD7A`	27 (0x1B)	0 (0x0)
30	3	`0x354C381994`	36 (0x24)	0 (0x0)
30	4	`0xFF5C3DDA48`	1 (0x01)	12 (0xC)
30	5	`0x1D0C9AB88E`	26 (0x1A)	0 (0x0)
30	6	`0x74FEF6A5B2`	33 (0x21)	12 (0xC)

In addition, one MACSEQ is transmitted in each MACK block, corresponding to 0x33A at TOW = 0 sec and 0xEDC at TOW = 30 sec.

A1.6.2. Tag Sequence Verification

As part of the tag verification, its ADKD shall be verified against the one predefined in the MAC look-up table. The sequence corresponding to the configuration 7 (MACLT 33) is recalled here.

ID	Msg	n_t	Sequence	
33	2	6	00S, 00E, 04S, 00E, 12S, 00E	00S, 00E, 00E, 12S, 00E, 12E

It can be seen that the ADKD sequence matches the one of the MACLT, and that self-authenticating tags (indicated with ‘S’ in the MACLT) are associated with PRN 1, or PRN 255 in case of ADKD4. Also, it can be seen that the order of the two MACK messages is respected, as the sequence transmitted in the first 30 seconds of the GST minute corresponds to the first part of the MACLT sequence.

A1.6.3. Associated Navigation Data

As an example, some of the tags retrieved in the sub-frame WN = 1145, TOW = 0 second are verified. These tags are authenticating the navigation data transmitted in the previous sub-frame, at WN = 1144 and TOW = 604770 seconds. The following table reports the E1-B I/NAV pages transmitted by E01, at the reported times of transmission. They are used in the following section for the tag verification.

WN	Time of transmission	Page part index	E1-B I/NAV page (120 bits)
1144	604771	1	0x021BBA49F42FC9F3FE51AE45A0EE80
1144	604772	2	0xBF2254A3C97EE1AAAAAA6861E3BF40
1144	604773	3	0x041B810008002B9D33F67B1067FF40
1144	604774	4	0xB1806D5E3975812AAAAA54AE5ABF40
1144	604775	5	0x0600000001FFFFFF12907889E25240
1144	604776	6	0xB4CE804EDCBE30EAAAAA5FF003FF40
1144	604777	7	0x09E3EE55555555555500AAAAAAAA80
1144	604778	8	0xAAAAA106C166BB6AAAAA4D689B3F40
1144	604779	9	0x0AEAAAAAAAAAAAAAAAAAAA006BFFC0
1144	604780	10	0x800E569751A8892AAAAA6D8E2BFF40
1144	604781	11	0x0095555555555555555555555551E240
1144	604782	12	0x8E9B7D417076E42AAAAA58DF90FF40
1144	604783	13	0x0095555555555555555555555551E240
1144	604784	14	0x8E9BD9B30A0071AAAAAA64DB69FF40

WN	Time of transmission	Page part index	E1-B I/NAV page (120 bits)
1144	604785	15	0x0095555555555555555555551E240
1144	604786	16	0x8E9C72BF900F8E6AAAAA597152FF40
1144	604787	17	0x0095555555555555555555551E240
1144	604788	18	0x8E9CC7C64681AEEAAAAA4AF18F7F40
1144	604789	19	0x0095555555555555555555551E240
1144	604790	20	0x8E9D68B1749DAFAAAAAA60F4047F40
1144	604791	21	0x011BA74CE15ACB1B001BDB92AA04C0
1144	604792	22	0x96ACC108705502AAAAAA78C848BF40
1144	604793	23	0x031BBFF0CB8825C276C2FB8711C300
1144	604794	24	0x90DADF120C344E6AAAAA4FF83DBF40
1144	604795	25	0x0521A0F80CF000200808F1274F6A80
1144	604796	26	0xAAAAB06DC429256AAAAA77E608FF40
1144	604797	27	0x0095555555555555555555551E240
1144	604798	28	0x8E9F65475CF0676AAAAA5002053F40
1144	604799	29	0x0095555555555555555555551E240
1145	0	30	0x8E9FDA288B40002AAAAA54625D3F40

A1.6.4. MACSEQ Verification

As no flexible tags are transmitted, the MACSEQ verification is not mandatory. It can however be performed. As per [AD.2], the message *m* used for the verification is obtained by concatenating the PRN of the satellite transmitting the OSNMA data, PRN_A, and the GST at the start of the sub-frame, GST_{SF}, corresponding to WN = 1145 and TOW = 0 sec.

m = `0x0147900000` (40 bits)

The MACSEQ is verified with the key of the next sub-frame, K_2, applying the MAC function indicated by the MF field, HMAC-SHA-256. The result of the MAC function is:

`0x33AF2A08A913D9EFFC4BD2598D2AC832F06DD8A3620DCEDC831EEFD596EE574E`
(256 bits)

This result is then truncated to the 12 more significant bits, leading to 0x33A. The value is compared bit-by-bit with the retrieved MACSEQ. As the values are the same, the verification is successful.

A1.6.5. Tags Verification

A1.6.5.1. Tag0 Verification

This section describes the verification of the tag received in first position (Tag_0) in the first of the two MACK messages provided in section A1.6.1 (WN = 1145, TOW = 0 sec). As per [AD.2], Tag_0 is associated to ADKD0. The navigation data to be used for the verification shall be retrieved in the previous sub-frame, reported in section A1.6.3 and is the following:

ADKD0 navigation data =
`0x1BA74CE15ACB1B001BDB92AA04D6AC1BBA49F42FC9F3FE51AE45A0EEBF221BBFF0C`
`B8825C276C2FB8711C310DAC6E04002000AE74CFD9EC419FFDC6008683E033C000802`
`00` (552 bits, 549 bits without padding)

Note that the ADKD0 navigation data has been right zero padded in order to be expressed in hexadecimal format.

The message *m* used for the verification is obtained by concatenating the different fields described in [AD.2]. It is reported below.

m =
`0x01479000000146E9D33856B2C6C006F6E4AA8135AB06EE927D0BF27CFF946B91683`
`BAFC886EFFC32E209709DB0BEE1C470C436B1B810008002B9D33F67B1067FF718021A`
`0F80CF00020080` (600 bits)

The message is hashed with the key of the next sub-frame, K_2, applying the MAC function indicated by the MF field, HMAC-SHA-256. The result of the MAC function is:

`0xE094B3FBA5426E2B414EDB1FDB2182B82C16B89EA2EB1121DC65C191C33C37CC`
(256 bits)

This result is finally truncated to the tag length indicated by the TS field, 40 bits.

`0xE094B3FBA5` (40 bits)

The value computed locally is then compared bit-by-bit with the retrieved Tag_0. As the values are the same, the verification is successful.

A1.6.5.2. ADKD4 Verification

This section describes the verification of the ADKD4 tag received in 3rd position in the first of the two MACK messages provided in section A1.6.1 (WN = 1145, TOW = 0 sec). The navigation data used for ADKD 4 verification shall also be retrieved from the previous sub-frame reported in section A1.6.3 and is the following:

ADKD 4 navigation data = `0x00000001FFFFFF12907889E25274CE00D7FF801C8` (164 bits, 161 without padding)

The message *m* used for the verification of the tag associated with ADKD4, as described in [AD.2], is reported below. Note that in the case of ADKD4, $PRN_D = PRN_A$ is used instead of PRND = 255. Also note that as the message does not fit an integer number of bytes, a 4-bit zero padding sequence is added.

m = `0x010147900000034000000007FFFFFC4A41E2278949D338035FFE00720` (224 bits)

The message is hashed with the chain key of the next sub-frame, K_2, applying the HMAC-SHA-256 function. The result of the MAC function is:

`0x5EBEC4E3A85050C8D73AF075AF9D66E7B24AB5B3EF06A1469731905BD28FFFF5`
(256 bits)

This result is finally truncated to the tag length indicated by the TS field, 40 bits.

`0x5EBEC4E3A8` (40 bits)

The value computed locally is then compared bit-by-bit with the retrieved tag. As the values are the same, the verification is successful.

A1.6.5.3. ADKD12 Verification

This section describes the verification of the ADKD12 tag received in 5th position in the first of the two MACK messages provided in section A1.6.1 (WN = 1145, TOW = 0 sec). The navigation data used for ADKD12 verification is the same as the one used for ADKD0, reported in section A1.6.5.1.

The message *m* used for the verification of the tag associated with ADKD 12 is reported below.

m =
0x0101479000000546E9D33856B2C6C006F6E4AA8135AB06EE927D0BF27CFF946B91683BAFC886EFFC32E209709DB0BEE1C470C436B1B810008002B9D33F67B1067FF718021A0F80CF00020080 (608 bits)

The message is hashed with the chain key transmitted with 10 sub-frame additional delay, corresponding to K_{12}, applying the HMAC-SHA-256 function. The result of the MAC function is:

0x78A85B8793FC38BBD9F5815D85B7DE468E1BC48460EF7E8008FE6AE0520BD453 (256 bits)

The result is finally truncated to the tag length, 40 bits.

0x78A85B8793 (40 bits)

The value computed locally is then compared bit-by-bit with the retrieved tag. As the values are the same, the verification is successful.

A1.7. DSM-PKR

The DSM-PKR is reported below:

DSM-PKR =
0x70AA1A8B68E5DB293106B5BC8806F9790E8ACF8DC2D28A6EF6C1AC7233A9813D3F86E53A50D345FBDAD49835F3363EE4A7262DB738CBDFC399229AE2803679300D6FB21E4DDF3F8E517A5C5B1C6D843F9236707FF11D96F9BA954BFEAA3A44E56BC8314BA8084E0CA101E595E88F170012F1F5CE71EEEFAB27334283E15935E8E61103F90DB0BE6BDF750835B1017A3A6084CBCB240928AEEFDBC19D1ACA99A3E9089962AD4833A51E (1352 bits)

A1.7.1. DSM-PKR Interpretation

The following public key parameters are extracted from the DSM-PKR message.

Field	Field value	Corresponding value
NB_{DP}	0x7	13 blocks
MID	0x0	Merkle tree leaf m_0
NPKT	0x1	ECDSA P-256
NPKID	0x1	1
P_{DP}	0x62AD4833A51E	0x62AD4833A51E

The message also contains the compressed ECDSA public key value:

0x03F90DB0BE6BDF750835B1017A3A6084CBCB240928AEEFDBC19D1ACA99A3E90899 (264 bits)

It also contains a set of four Merkle tree intermediate nodes. As per [AD.2], the intermediate nodes transmitted with the Merkle tree leaf m_0 are $x_{0,1}$, $x_{1,1}$, $x_{2,1}$ and $x_{3,1}$. They are reported below.

Idx	Node (256 bits)
$x_{0,1}$	0xAA1A8B68E5DB293106B5BC8806F9790E8ACF8DC2D28A6EF6C1AC7233A9813D3F
$x_{1,1}$	0x86E53A50D345FBDAD49835F3363EE4A7262DB738CBDFC399229AE2803679300D
$x_{2,1}$	0x6FB21E4DDF3F8E517A5C5B1C6D843F9236707FF11D96F9BA954BFEAA3A44E56B
$x_{3,1}$	0xC8314BA8084E0CA101E595E88F170012F1F5CE71EEEFAB27334283E15935E8E6

A1.7.2. DSM-PKR Verification

In order to perform the verification of the DSM-PKR, the received message is verified against a known Merkle tree root, which is retrieved from the GSC OSNMA server and is reported below:

`0xC5B2A3BD24E819EF82B17ACE83C0E7F41D34AC9B488CB7CE4D765FDE7DCA0297`
(256 bits)

As a first step of the verification, the Merkle tree leaf identified by the MID, m_0, is obtained by concatenating the NPKT, NPKID and NPK fields, as per [AD.2].

m_0 =
`0x1103F90DB0BE6BDF750835B1017A3A6084CBCB240928AEEFDBC19D1ACA99A3E90899` (272 bits)

This message is then hashed to obtain the intermediate node $x_{0,0}$.

$x_{0,0}$ =
`0x40CAA1D70F7B1D370219674A25721311170A49DE4E4A0CE4FE328674E01CF750`
(256 bits)

This initial node is then used together with the received $x_{0,1}$ value to compute $x_{1,0}$, as per the formula in [AD.2]. The operation is iterated until the Merkle tree root, $x_{4,0}$, is obtained. The computed intermediate tree nodes are listed below.

$x_{1,0}$ =
`0x1E12368CFC91AD8560CBF5CA03E8EC2B19206DAFBE14092D9A4F0F3827428677`
(256 bits)

$x_{2,0}$ =
`0xE5F968BD94AF9B4ACCAA34BE782C3F9633D490549DC7889E0ADE61669AD85D26`
(256 bits)

$x_{3,0}$ =
`0x707519B194C1513866AD22FD32B228B01B77D479D55F7A27BF0DF625096FBD1B`
(256 bits)

$x_{4,0}$ =
`0xC5B2A3BD24E819EF82B17ACE83C0E7F41D34AC9B488CB7CE4D765FDE7DCA0297`
(256 bits)

The locally generated Merkle tree root, $x_{4,0}$, is compared bit-by-bit to the stored value. As the values are the same, the received public key and its associated parameters are verified.

A1.7.3. Verification of the P_{DP}

This section describes the verification of the DSM-PKR padding bits, P_{DP}. As this verification is not mandatory, it is not covered in the OSNMA Receiver Guidelines. The description below is provided for the sake of completeness and in the interest of the developer.

The message to be hashed for this verification is the concatenation of the Merkle tree root and the Merkle tree leaf, m_0, transmitted in the DSM-PKR:

`0xC5B2A3BD24E819EF82B17ACE83C0E7F41D34AC9B488CB7CE4D765FDE7DCA0297110 3F90DB0BE6BDF750835B1017A3A6084CBCB240928AEEFDBC19D1ACA99A3E90899` (528 bits)

The message is then hashed with the SHA-256 function, leading to:

`0x62AD4833A51EED2F0A5521CDF7280A0C3C7B3788D0BA54755F4B5B2455D89381` (256 bits)

This result is further truncated to the length of P_{DP} and compared bit-by-bit to the received field.

`0x62AD4833A51E` (48 bits)

Annex 2. OSNMA Configuration Examples

This section describes the structure of the HKROOT and MACK messages for OSNMA configuration examples.

A2.1. Example 1

The parameters of the TESLA chain for example 1 are summarised in Table 9.

Parameters	Settings
Tag size	40 bits
Key size	128 bits
Number of tags per MACK message (n_t)	6
Digital Signature algorithm	ECDSA P-256
Nb. of blocks in the DSM-HKROOT (NB_{DK})	8
Hash Function	SHA-256
MAC Function	HMAC-SHA-256
MACLT examples	27, 33

Table 9. OSNMA parameters for Example 1

The structure of the associated DSM-KROOT message is shown in Figure 6. One line represents the data sent over one sub-frame (30 sec), corresponding to 8 bits every 2 sec.

The time indicated on the left corresponds to the transmission time of the first bit of the odd page in which the OSNMA data is transmitted. As per [AD.2], GST_{SF} is the Galileo System Time at the start of the 30-seconds sub-frame in which the data is transmitted. For Galileo E1 I/NAV, this is the E1 sub-frame start time minus 1 second.

As per [AD.2], all the blocks start with the NMA header and the DSM header.
The NMA header contains:

- NMA Status (NMAS)
- Chain ID (CID)
- Chain and Public Key Status (CPKS)
- A reserved field (Rsvd)

The DSM header contains:

- Identifier of the DSM associated with the current block (DSM ID)
- DSM Block ID which identifies the position of the block in the overall DSM message (DSM BID)

In the case of the first block (BID = 0), the remaining bits provides the parameters of the TESLA chain, including:

- Number of DSM-KROOT blocks (NB_{DK})
- Public Key ID (PKID)
- KROOT chain ID (CIDKR)
- Reserved1 (Rsvd1)
- Hash Function (HF)
- MAC Function (MF)
- Key Size (KS)
- Tag Size (TS)
- MAC Look-up Table (MACLT)
- KROOT Week Number and Time of Week (WN_K and TOW_K)
- Random pattern α

The following blocks (BID = 1 to NB_{DK}-1) contain the digital signature (DS). When required, the last block may contain padding bits. The different fields are described in [AD.2].

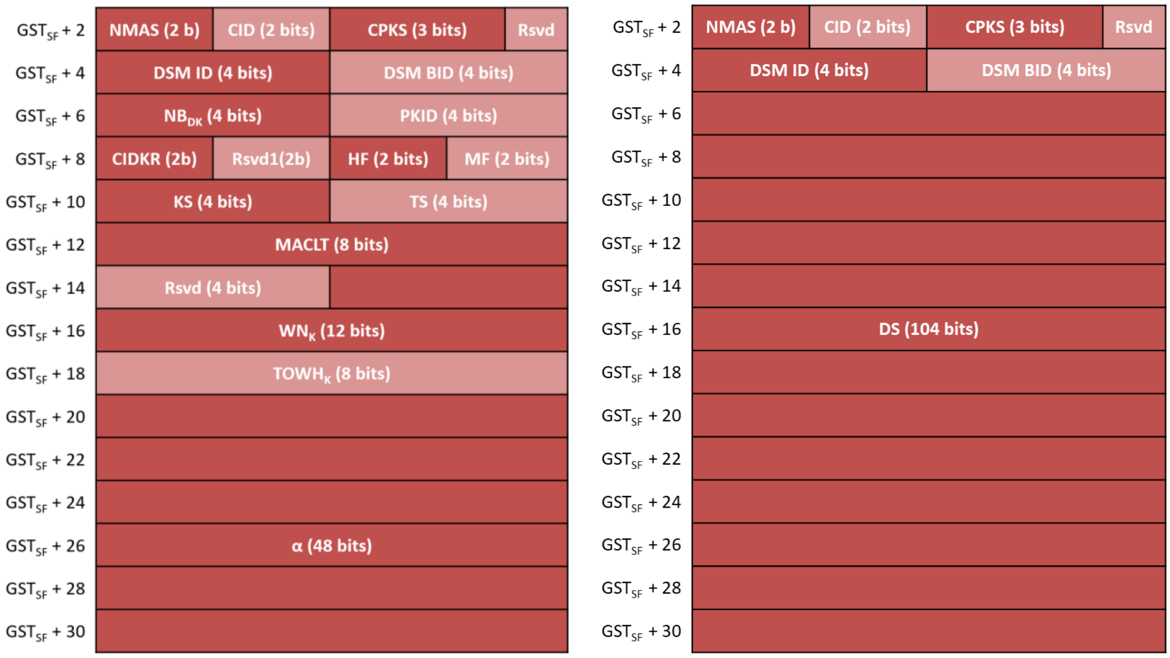


Figure 6. HKROOT message structure for Block ID BID = 0 (left) and for BID = 1 to NB$_{DK}$-1 (right)

The structure of the MACK message is shown in Figure 7. It represents the data sent over one sub-frame (30 sec), corresponding to 32 bits every 2 sec. As for the KROOT message, the time at which the data is sent within the sub-frame is indicated on the left side of the line.

The MACK message contains:

- Tag$_0$ field
- MACSEQ
- Reserved2 (Rsvd2)
- Tags
- PRN of the satellite transmitting the navigation data which is authenticated by the tag (PRN$_D$)
- Authentication Data and Key Delay (ADKD)
- TESLA chain key
- Padding bits

These different fields are described in [AD.2].

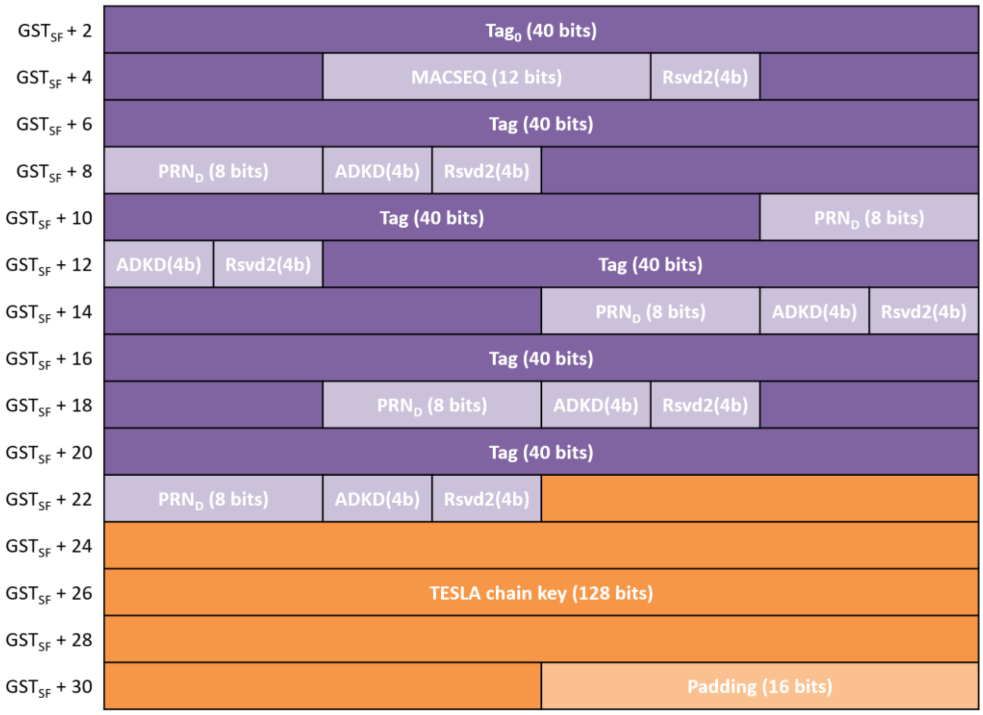


Figure 7. MACK message structure for Example 1

A2.2. Example 2

The parameters of the TESLA chain for example 2 are summarised in Table 10.

Parameters	Settings
Tag size	20 bits
Key size	120 bits
Number of tags per MACK message (n_t)	10
Digital Signature algorithm	ECDSA P-256
Nb. of blocks in the DSM-HKROOT (NB_{DK})	8
Hash Function	SHA-256
MAC Function	HMAC-SHA-256
MACLT example	28

Table 10. OSNMA parameters for Example 2

The structure of the KROOT message is similar for all the configurations (except for the value of NB_{DK}) and is not reminded here. The structure of the MACK message is shown in Figure 8.

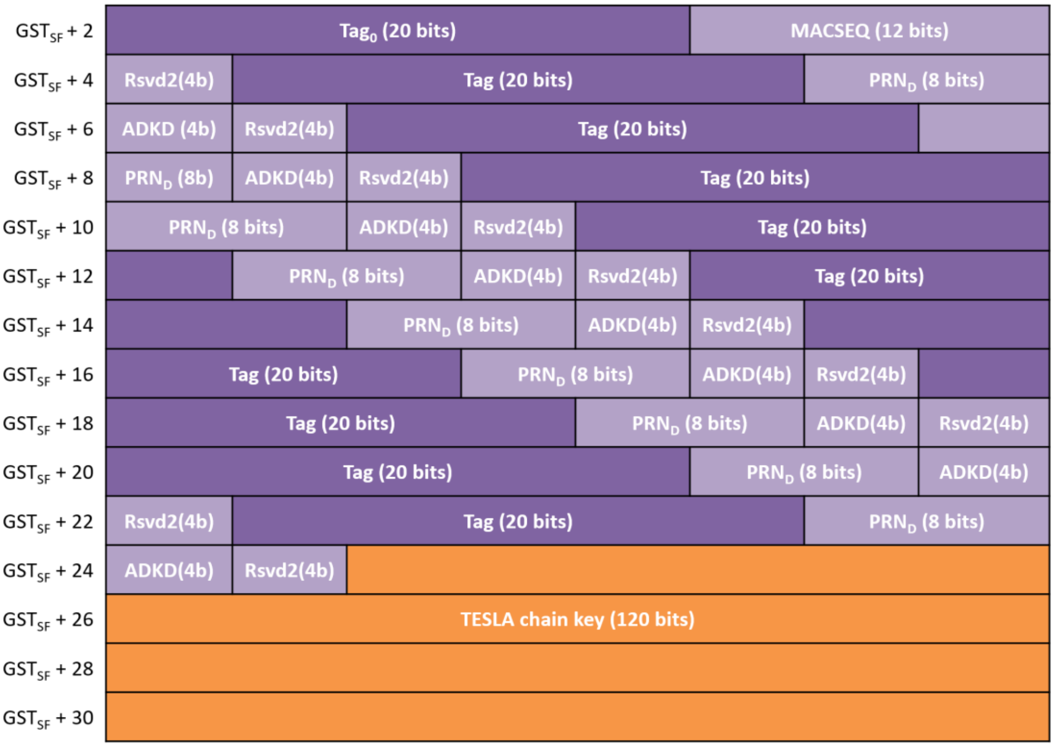


Figure 8. MACK message structure for Example 2

A2.3. Example 3

The parameters of the TESLA chain for the example 3 are summarised in Table 11.

Parameters	Settings
Tag size	40 bits
Key size	192 bits
Number of tags per MACK message (n_t)	5
Digital Signature algorithm	ECDSA P-521
Nb. of blocks in the DSM-HKROOT (NB_{DK})	13
Hash Function	SHA-256
MAC Function	HMAC-SHA-256
MACLT example	31

Table 11. OSNMA parameters for Example 3

The structure of the MACK message is shown in Figure 9.

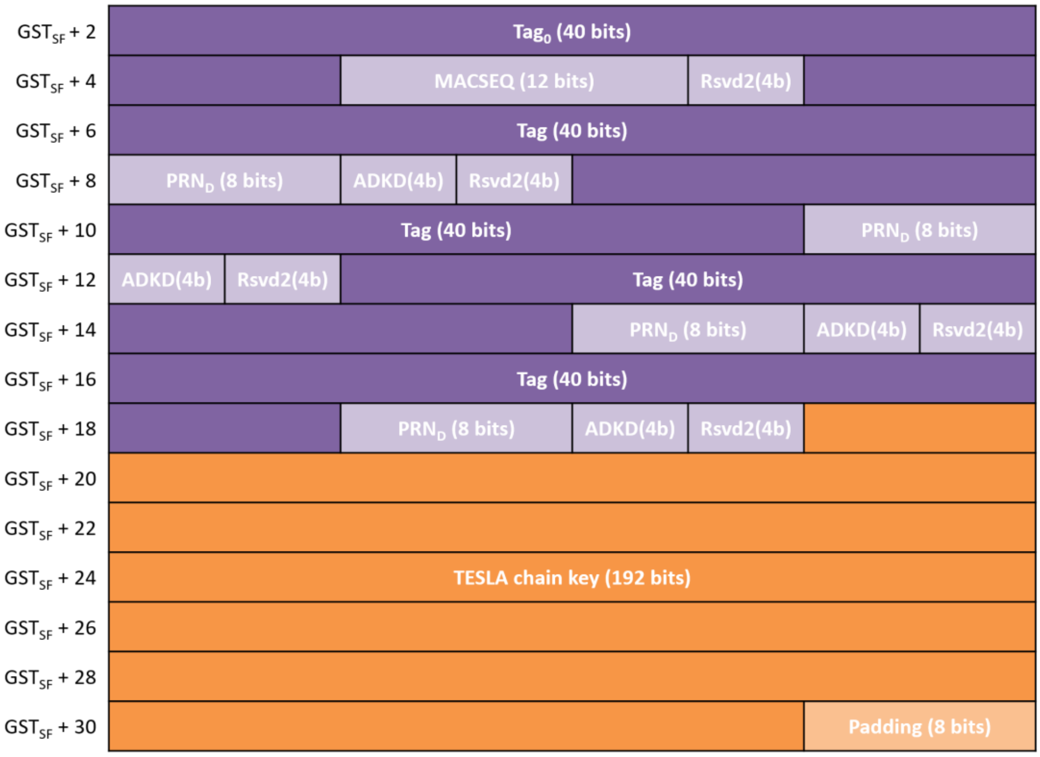


Figure 9. MACK message structure for Configuration 5

Annex 3. Receiver Initial Conditions and Fulfilment of the Time Synchronisation Requirement

As explained in section 2.1, the security of the OSNMA protocol depends on the fulfilment by the receiver of the time synchronisation requirement. This requirement can be expressed as function of the receiver time synchronisation uncertainty *B*, with respect to Galileo System Time, as follows:

$$B < \frac{\mathrm{T_L}}{2} \qquad \text{Eq. 5}$$

B is defined such that:

$$t_i^{\mathrm{GST}} \in [t_i^{Rx} - B; t_i^{Rx} + B] \qquad \text{Eq. 6}$$

Where t_i^{Rx} is the time of event *i* as reported in the receiver local time and t_i^{GST} is the time of event *i* as reported in Galileo System Time.

A receiver with a synchronisation uncertainty with respect to GST,B, such that $B < \frac{T_L}{2} + 150\ sec$, can process slow MAC with a 10 sub-frame delay (ADKD12 from [AD.2]).

If none of the above conditions on the receiver time synchronisation uncertainty *B* is verified, the OSNMA protocol shall not be used.

At start up, the receiver may be in one of two states:

- It has a knowledge of its time synchronisation uncertainty *B*,
- It has no time information.

These two cases are discussed in the following sections.

A3.1. Known Time Synchronisation Uncertainty

The first case corresponds, for example, to a receiver with an internal Real Time Clock (RTC) running on battery when the RF section and main GNSS core are powered down. At the switch on, the receiver must get information on its time synchronisation uncertainty *B*, which may be estimated based on the stability of the clock used. The receiver can then proceed as described below:

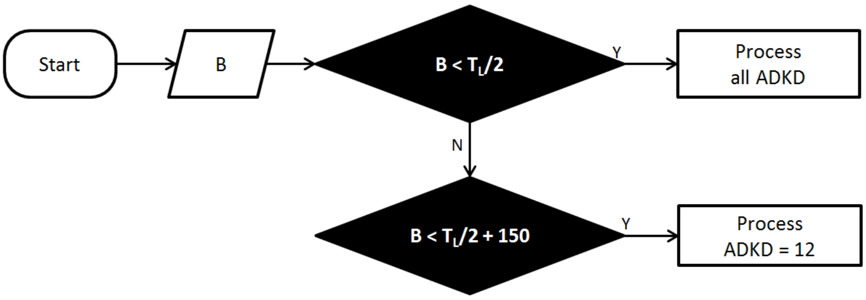


Figure 10. ADKD processing as a function of the time synchronisation uncertainty

If the time synchronisation requirement is fulfilled ($B<T_L/2$), the receiver can process all tags, regardless of their ADKD types. If the receiver does not fulfil the requirement, it may still be able to exploit the slow MACs, corresponding to ADKD = 12, as illustrated in Figure 10.

To be noted that the timing solution computed with the data authenticated is not itself authenticated. Therefore, if this time solution is used to re-synchronise the clock (i.e. to reduce *B*), an associated risk shall be taken into account.

A3.2. Unknown Time Synchronisation Uncertainty

If the receiver has no time information, an estimation of the GST and its associated uncertainty shall be retrieved. Several strategies are possible, for example:

- An external clock can be used;
- For connected users, a secure source can be exploited, e.g., a secure network connection with a time transfer capability smaller than the synchronization requirement. This condition is achievable by conventional approaches, e.g., a secured implementation of the Network Time Protocol (NTP) can usually maintain time to within tens of milliseconds over the public internet, achieving better than one millisecond accuracy in local area networks under ideal conditions.

Care should be taken to estimate the Galileo System Time and therefore to perform any conversion, if required, taking into account any associated uncertainty.

As stated in section 2.1, the security of the protocol relies on the correct synchronisation of the receiver with the system time. Thus, considerations about the security and associated risks of accessing the timing information shall be taken into account.

Annex 4. Increasing Resilience of Receivers to Spoofing Attacks and the Role of OSNMA

OSNMA provides means to authenticate navigation data, which can then be used to compute a PVT. It should be noted though, that as this PVT is computed using non-verified ranging information, it cannot be considered authenticated. Thus, users can obtain a more robust PVT solution by combining OSNMA with additional checks in the receiver. It is to be noted that the insertion of OSNMA data within I/NAV increases the unpredictability of I/NAV data stream itself. In particular, all tags are unpredictable by definition and, for each sub-frame, the earliest-received key is also unpredictable. The receiver can use the unpredictability of the I/NAV symbols encoding OSNMA to make the signals more robust against replay attacks. In addition, some examples of receiver consistency checks are provided below.

At the signal processing level:

- Search and detection of vestigial signals around possibly false tracked signal can be carried out;
- Tracking control loop parametrisation and correlation function can be monitored.

At the measurement level:

- Consistency checks between the Automatic Gain Control (ACG) and Carrier-to-Noise (C/N0) values can be performed to detect abnormal power emissions;
- Measurements can be monitored over time to detect abrupt changes;
- Receiver autonomous integrity monitoring (RAIM) techniques, including the detection of measurements inconsistent with the estimated position, can be implemented.

At the PVT level:

- The position and velocity solutions can be monitored over time to detect abnormal values, sudden changes and trajectories and dynamics that are not consistent with the vehicle's dynamics;
- The receiver can monitor the time solution over time and cross-check it against the performance of the internal clock.

Beyond the GNSS receiver itself:

- Anti-tampering measures can be put in place to prevent the manipulation of the GNSS receiver and antenna;
- Additional sensors (e.g. odometers, inertial sensors) can be used to perform consistency checks;
- Multiple antennas enable the receiver to detect the presence of counterfeit signal from the direction of arrival of the signals.