Title	**SPEAR OSNMA SDK: GNSS Anti-spoofing Solution**
Document type:	Technical note
Author(s):	Luis Romero, Aleix Tejada, Xavier Banqué-Casanovas
Approved/Reviewed by:	Miquel Garcia-Fernàndez.
Document number:	SPEAR OSNMA SDK white paper v1
Version:	1.0

Table of Contents

Document Change Record

Date	Version	Author	Brief Comment
2024-02-25	0.1	XBC	1st released version
2024-03-01	1.0	MGF	1st released version (reviewed)

1. SPEAR POSITIONING ENGINE INTRODUCTION

SPEAR is a positioning engine (PE) software library able to deliver navigation solutions which are equivalent to the widely known industry standards: SPP, DGNSS, RTK, PPP, RTK-PPP. The library is delivered as a Software Development Kit (SDK) and is highly flexible and platform-agnostic.

SPEAR PE is implemented using low level programming language (C) so that it can be easily integrated into any platform whether it is at an OS level (e.g. embedded) or at application level (e.g. Android). It is designed to provide the end user with robust and accurate navigation, including the latest Galileo services: HAS (High Accuracy Service) and OS-NMA (Open Service - Navigation Message Authentication), among others. SPEAR architecture is orchestrated around different modules, each of them addressing different functionalities such as precise navigation or authentication against spoofing attacks.

The present document focuses on the specific module of the SPEAR library that provides Galileo navigation message authentication (OSNMA) as a means to identify spoofing attacks. **This module is provided and deployed as a standalone library, suitable for embedded applications and platforms.**

2. SPEAR OSNMA SDK

Given the advent of the increasing number of GNSS spoofing events being reported, applications relying on satellite navigation are in need of protection against this sophisticated kind of interference. Spoofing consists in the forgery of the satellite navigation signals so that GNSS receivers inadvertently provide a falsified location without the receiver having the means of detecting it. The European GNSS constellation, Galileo, will provide a pioneering service that will allow the detection of these spoofing events in Galileo capable receivers. This service, which is called Open Service - Navigation Message Authentication[1] (OSNMA), provides the necessary cryptographic means to authenticate the navigation message broadcasted by Galileo satellites. The navigation device could then raise a warning of a potential spoofing event in case of failure in the authentication of the OSNMA cryptographic material.

Rokubun's Galileo Open Service Navigation Message Authentication (OSNMA) Library (SDK) is the ultimate solution for decoding and processing Galileo OSNMA service for embedded navigation platforms. The OSNMA SDK enables the use of Galileo authentication in a navigation (embedded) platform in order to provide a more robust Positioning Navigation and Timing (PNT) solution. SPEAR OSNMA SDK is the fastest and most cost-effective solution to provide authentication to any positioning solution. This library will bring you up to speed, relative to similar solutions, in the provision of robust PNT solutions, able to detect spoofing episodes. SPEAR OSNMA SDK is **fully compliant with the latest versions of the Galileo OSNMA Signal-in-Space Interface Control Document**[2] and the Galileo OSNMA Receiver Guidelines[3] published by the European Union. Here is what sets it apart:

- Perfect Fit for Embedded and Automotive: Engineered in C with meticulous attention to **MISRA compliance**, Rokubun's OSNMA Library has been optimized for embedded and automotive platforms, ensuring seamless integration and reliable performance.
- Automotive SPICE qualification ongoing.
- Its **small footprint** will maximize the potential of any platform: **it is just 72 KB of code and needs only 12 KB of RAM.** It does not rely on dynamic memory.
- **Robustness**: Provides Galileo OSNMA authentication even in suboptimal satellite signal reception conditions, like deep urban scenarios.
- **Cross-Platform** and Easy Integration: SPEAR OSNMA library is designed to be architecture and OS agnostic, allowing for smooth integration across various platforms. The library can be deployed in an embedded MCU, host MCU or external processor.
- **Fits every build system**: The library can be built using the compilation toolchain of the customer's target architecture. It will be packaged and delivered as a single static library and header file, so that adding the library to any project is a matter of minutes.

[1] https://www.gsc-europa.eu/galileo/services/galileo-open-service-navigation-message-authentication-osnma

[2] OSNMA SIS ICD, Issue 1.1, October 2023

[3] OSNMA Receiver Guidelines, Issue 1.3, January 2024

SPEAR OSNMA SDK is suitable for:

- GNSS Chipset (SoC, SoM, SoP…) Manufacturer: the library can be embedded into the MCU of a GNSS SoC so that your solution provides authentication of the Galileo navigation message as a spoofing detection measure.
- GNSS Receiver Manufacturer: the library can be embedded into the host processor of the GNSS receiver so that this provides authentication of the Galileo navigation message as a spoofing detection measure. Get the most of Galileo OSNMA such as providing a PNT solution using only authenticated (navigation message) satellites.
- Navigation solution integrator (i.e. Automotive TCU or OBU, robotics/drone navigation OBU, IoT tracker….): the library can be embedded into the automotive or robotics GNSS-based navigation On-board Unit or Telematics Control Unit so that the navigation solution features Galileo navigation message authentication as a means of anti-spoofing protection. Get the most of Galileo OSNMA such as providing a PNT solution using only authenticated (navigation message) satellites.
- GNSS Simulators and Scientific receivers: SPEAR OSNMA SDK can be used to validate the implementation of the Galileo OSNMA in the I/NAV (E1-B) message in GNSS simulation solutions. The library can thus be used for the fetching, decoding and processing of the Galileo OSNMA pages in the I/NAV message to verify correct OSNMA implementation on the simulated SiS.

SPEAR OSNMA SDK has undergone extensive testing using the official OSNMA test vectors provided by EUSPA, and additionally, it has been validated in real conditions at the European Commission's Galileo testing facilities at the Joint Research Center (JRC) in Ispra, Italy.

The OSNMA SDK processing flow can the summarized in two stages (Figure 1): (1) reception of the necessary navigation and authentication data and (2) verification of this data in order to check authenticity[4]. During the reception stage, the SDK retrieves the necessary data structures to trigger the authentication algorithms. Once the reception stage is completed, the authentication stage generates the authenticated navigation data for the Galileo satellites whose data was received in the reception stage. Complementary, it can provide a list of satellites that have been authenticated. SPEAR OSNMA SDK is robust in harsh GNSS reception conditions (such as urban canyons, dense foliage, or obstructions), where up to 60% of the navigation message pages may be lost. This allows navigation devices embedding our SDK to provide navigation authentication even in GNSS challenging environments such as dynamic urban use cases.

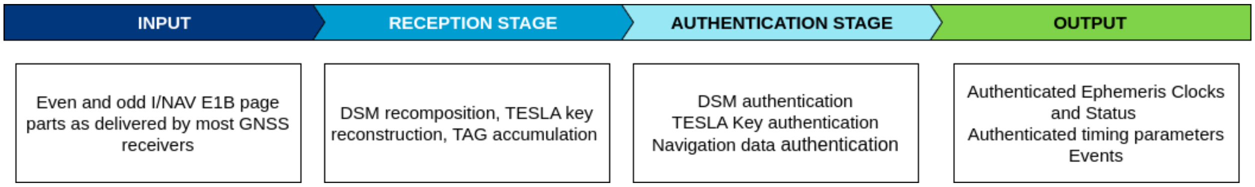
INPUT
RECEPTION STAGE
AUTHENTICATION STAGE
OUTPUT
Even and odd I/NAV E1B page parts as delivered by most GNSS receivers
DSM recomposition, TESLA key reconstruction, TAG accumulation
DSM authentication
TESLA Key authentication
Navigation data authentication
Authenticated Ephemeris Clocks and Status
Authenticated timing parameters
Events

Figure 1 Diagram of the SPEAR OSNMA SDK processing flow

[4] For detailed OSNMA SDK algorithm architecture, see diagram in Annex 1 of this document.

Input data for the OSNMA SDK is:

- Raw Galileo I/NAV E1b navigation message pages, as provided by most GNSS receivers

Output data for the OSNMA SDK is:

- Authenticated Navigation data: Ephemeris, Clocks and satellite Status (ADKD 0) upon OSNMA successful authentication for a given satellite and epoch.
- Authenticated Navigation data: Timing Parameters (ADKD 4) upon OSNMA successful authentication for a given satellite and epoch.
- OSNMA authentication failure events upon failed authentication for a given satellite and epoch.
- Events related to authentication status, constellation status and other relevant parameters.

The OSNMA SDK has been designed with flexibility in mind. Navigation devices embedding it can use the resulting OSNMA output data to perform several higher level actions leveraging authentication information, such as:

- Assessment of the likeliness of a spoofing event by: (a) monitoring authentication success satellites vs authentication failed satellites (spoofed), (b) using authenticated navigation data and/or (c) reception of OSNMA authentication failure events.
- Exclusive usage of the authenticated navigation data (ADKD 0, 4) to compute an authenticated PVT solution.

A set of comprehensive examples on how to use the SDK for several use cases are provided alongside the SDK.

The time-to-first-authenticated-fix (TTFAF) of the navigation device will depend mainly on the initialization mode and the use case (static open sky conditions, dynamic rural road conditions, dynamic urban road conditions…). The OSNMA SDK supports three different initialization modes:

- **Cold-start**: The receiver neither possesses the Public Key nor the TESLA Root Key and they need to be retrieved from the Galileo I/NAV message.
- **Warm-start**: The receiver stores the Public key in non-volatile memory and therefore can retrieve and verify the TESLA Root Key and then proceed with the navigation data authentication.
- **Hot-start**: The receiver stores both the verified Public Key and a TESLA Root or Chain key in non-volatile memory so that it can immediately proceed with the navigation data authentication.

To accommodate systems with constrained resources, the memory-dependent methods are implemented via a Hardware Abstraction Layer (HAL). Examples of HAL integration in Linux and bare metal implementations are provided with the SDK.

3. Performance Figures

The SPEAR OSNMA SDK has been integrated and tested in embedded MCUs (ARM Cortex M4F, M7, Xtensa LX6), MPUs (Arm Cortex A8, A72) as well as X86 systems. This demonstrates the platform agnosticism of the library. All the figures presented in this section are obtained running the library in a standard X86 laptop. Detailed metrics (i.e. number of processor cycles) for embedded architectures can be provided on request.

The SPEAR OSNMA SDK performance has been assessed using one of most common key performance indicators found in the literature and technical reports on authenticated positioning: Time to First Authenticated Fix (TTFAF). TTFAF is the time the navigation device takes to generate the initial authenticated PVT fix, i.e. a PVT generated using the navigation information from the first 4 authenticated satellites. This parameter strongly depends on the OSNMA initialization mode, which in turn depends on the availability of the Public Key and/or the TESLA Root Key in the navigation device, and on the operating scenario (i.e. static or dynamic, open-sky or urban). These scenarios determine the rate of I/NAV message pages properly received, which can be therefore processed. The TTFAF performance of the SPEAR OSNMA SDK has been assessed under different combinations of initialization modes and operating scenarios. The number of satellites in view transmitting OSNMA data is also an important parameter to be taken into account. As of today, only 20 Galileo satellites are transmitting OSNMA bits. In our performance assessment, a maximum of 9 satellites transmitting OSNMA in view are considered since this is the maximum typical number of Galileo satellites in view in open-sky conditions in mid latitudes (typical elevation mask 5 degrees).

The official OSNMA test vectors, provided by EUSPA along with the Galileo OSNMA Receiver Guidelines document[5], have been used to test the performance of the OSNMA SDK. These provide 1 hour duration navigation datasets covering different scenarios concerning Galileo OSNMA service provision. The main parameters used to obtain the performance figures hereafter are:

- **I/NAV Page drop rate**: that sets the percentage of navigation pages properly received. This parameter is used to degrade the acquisition conditions in order to emulate the loss of I/NAV pages due to reduced visibility and receiver motion in nominal operating environments (i.e. urban scenarios, where the drop rate can be 50% to 60%, as per internal assessments in real conditions experiments).
- **Number of satellites in view**: The number of satellites transmitting OSNMA that the receiver is tracking at a time, greatly depends on the scenario. In open sky conditions, the receiver will be tracking more satellites than in suburban or urban environments. In our simulations, we have considered a range of 4 to 9 satellites transmitting OSNMA in view to assess the availability of OSNMA data
- **OSNMA initialization mode**: The starting mode of the receiver will inherently impact the time it takes to authenticate the navigation message of Galileo Satellites.

Figures below show the average TTFAF as a function of the I/NAV page loss rate and the number of visible satellites transmitting OSNMA. Note that under ideal conditions, the TTFAF is lower-bounded by the time taken by the Galileo system to transmit the complete set of OSNMA data which includes the public key, the Tesla root key, the navigation message and the authentication tags (in cold start mode).

[5] Galileo OSNMA Receiver Guidelines (v1.3)

The following figure shows the TTFAF for the following scenarios of interest based on the page loss rate[6]:

- Base station (static open sky): 8 satellites in view, 0% to 15% I/NAV page loss rate.
- Open sky dynamic: 6 satellites in view, 15% to 45% I/NAV page loss rate.
- Dynamic urban: 5 satellites in view, more than 45% I/NAV page loss rate.

Results obtained prove that our OSNMA SDK is robust to provide authentication even in harsh GNSS acquisition environments that involve a significant loss of navigation pages. The figure shows that the fastest TTFAF that can be achieved, with 0% drop rate, with SPEAR OSNMA SDK is ca. 90s for hot (Figure 4), 150s for warm (Figure 3) and 600s cold start (Figure 2) respectively. This means that the SPEAR OSNMA SDK can be used in urban canyons and authenticated navigation messages will still be provided by the library for robust navigation.

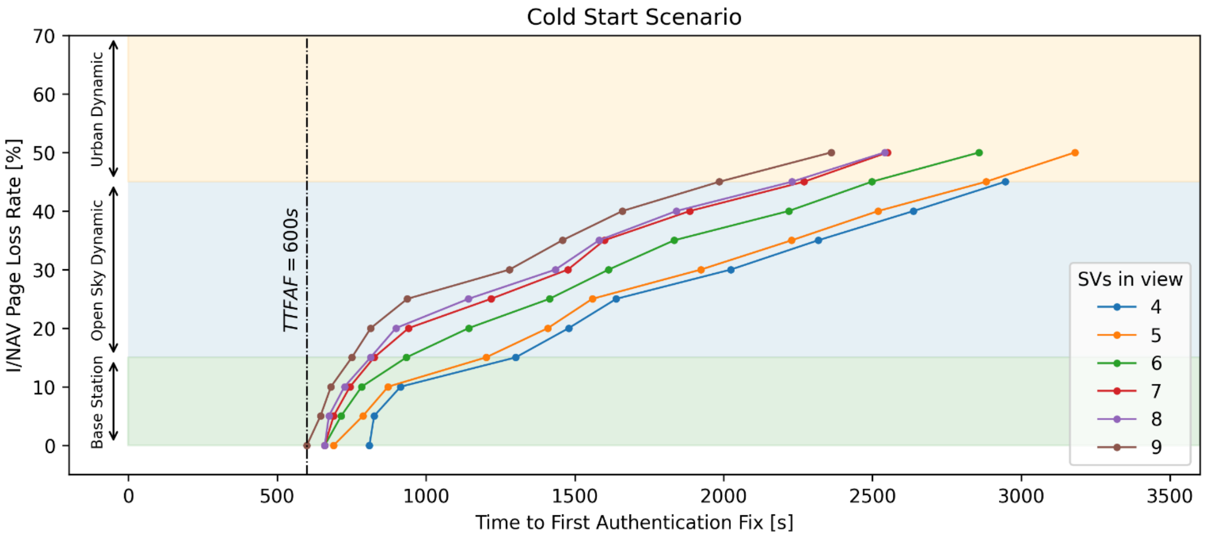

Figure 2 Statistics of the TTFAF for different drop rates in cold start

[6] Note that the page loss rate define the type of scenario being considered

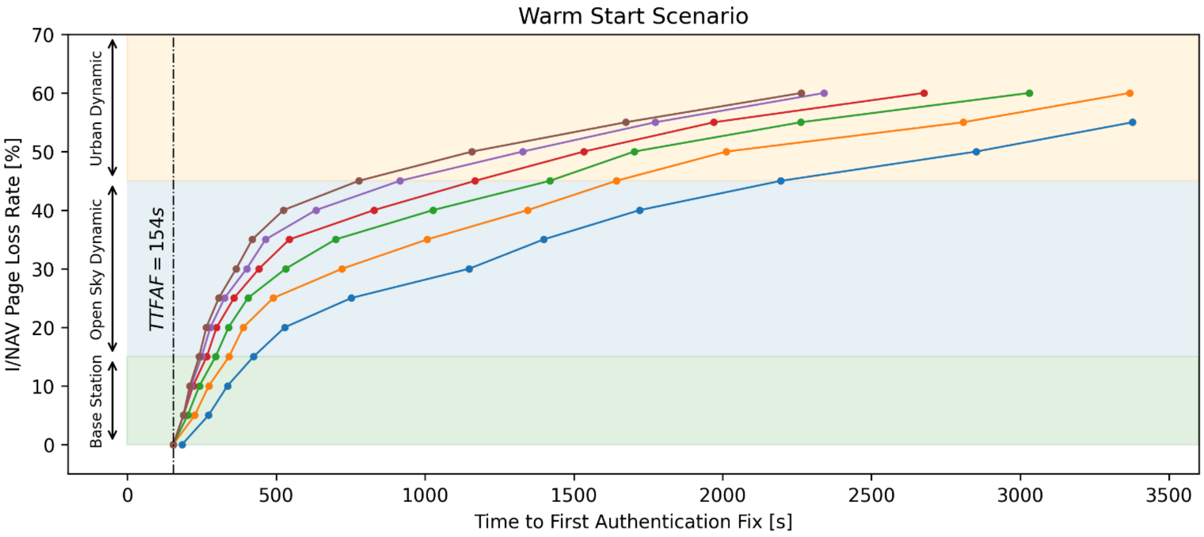

Figure 3 Statistics of the TTFAF for different drop rates in warm start

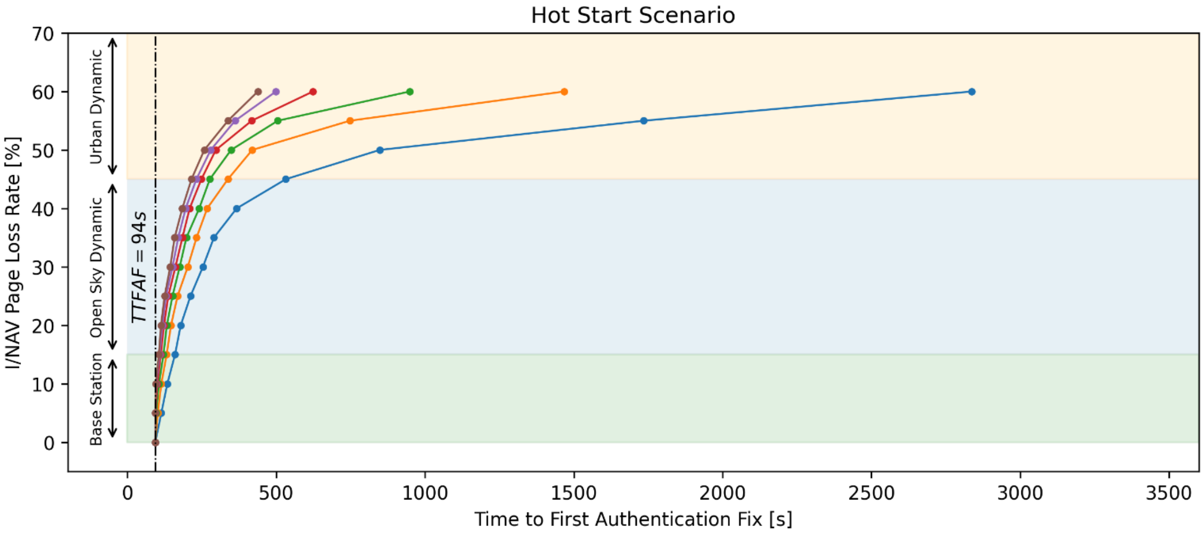

Figure 4 Statistics of the TTFAF for different drop rates in hot start

ANNEX I: SPEAR OSNMA SDK ARCHITECTURE

A simplified architecture of the most important elements that constitute the SPEAR OSNMA SDK is shown in Figure 5. In particular, it shows the required **inputs** needed by the library as well as the **outputs** generated by it (in the form of events that can be handled by the client using the library by means of e.g. callback methods). Besides the I/O elements, the library is split into the module that parses (extracts) the OSNMA bits from the I/NAV message and then forwards it to the “MACK and DSM RECEIVER'' module that reconstructs the MACK and Digital Signature Message (DSM). The output events are then generated by the “TAG and KEY VERIFICATION ENGINE '' module, which is the ultimate responsible to verify and authenticate the navigation data.

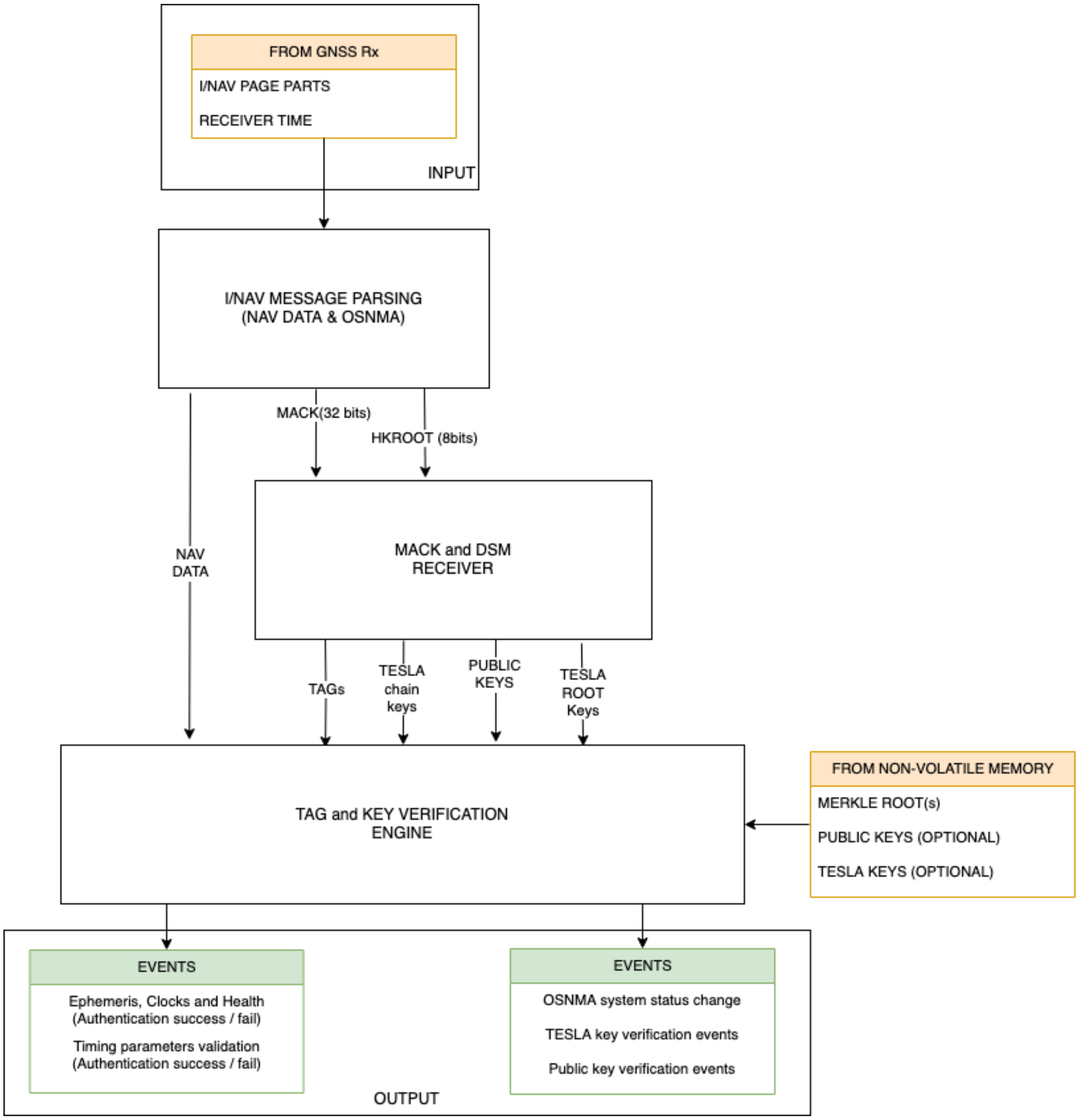
FROM GNSS Rx
I/NAV PAGE PARTS
RECEIVER TIME
INPUT
I/NAV MESSAGE PARSING (NAV DATA & OSNMA)
MACK(32 bits)
HKROOT (8bits)
NAV DATA
MACK and DSM RECEIVER
TAGs
TESLA chain keys
PUBLIC KEYS
TESLA ROOT Keys
TAG and KEY VERIFICATION ENGINE
FROM NON-VOLATILE MEMORY
MERKLE ROOT(s)
PUBLIC KEYS (OPTIONAL)
TESLA KEYS (OPTIONAL)
EVENTS
Ephemeris, Clocks and Health (Authentication success / fail)
Timing parameters validation (Authentication success / fail)
EVENTS
OSNMA system status change
TESLA key verification events
Public key verification events
OUTPUT

Figure 5 Architecture of the SPEAR OSNMA SDK library