

Practical Spoofing Attacks on Galileo Open Service Navigation Message Authentication

Haiyang Wang, Yuanyu Zhang, *Member, IEEE*, Xinghui Zhu, Ji He, *Member, IEEE*, Shuangtrui Zhao, *Member, IEEE*, Yulong Shen, *Member, IEEE* and Xiaohong Jiang, *Senior Member, IEEE*

Abstract—This paper examines the Galileo Open Service Navigation Message Authentication (OSNMA) and, for the first time, discovers two critical vulnerabilities, namely artificially-manipulated time synchronization (ATS) and interruptible message authentication (IMA). ATS allows attackers falsify a receiver’s signals and/or local reference time (LRT) while still fulfilling the time synchronization (TS) requirement. IMA allows temporary interruption of the navigation data authentication process due to the reception of a broken message (probably caused by spoofing attacks) and restores the authentication later. By exploiting the ATS vulnerability, we propose a TS-comply replay (TSR) attack with two variants (real-time and non-real-time), where attackers replay signals to a victim receiver while strictly complying with the TS rule. We further propose a TS-comply forgery (TSF) attack, where attackers first use a previously-disclosed key to forge a message based on the OSNMA protocol, then tamper with the victim receiver’s LRT correspondingly to comply with the TS rule and finally transmit the forged message to the receiver. Finally, we propose a concatenating replay (CR) attack based on the IMA vulnerability, where attackers concatenate replayed signals to the victim receiver’s signals in a way that still enables correct verification of the navigation data in the replayed signals. To validate the effectiveness of the proposed attacks, we conduct real-world experiments with a commercial Galileo receiver manufactured by Septentrio, two software-defined radio (SDR) devices, open-source *Galileo-SDR-SIM* and *OSNMAlib* software. The results showed that all the attacks can successfully pass the OSNMA scheme and the TSF attack can spoof receivers to arbitrary locations.

Index Terms—OSNMA, message authentication, vulnerability analysis, spoofing attacks.

I. INTRODUCTION

GALILEO satellite navigation system is a critical component of the Global Navigation Satellite Systems (GNSS), providing positioning, navigation, and timing (PNT) services globally through satellites covering the entire Earth’s surface. Like GPS, Beidou and GLONASS, Galileo civilian signals are susceptible to spoofing attacks, due to the openness of signal formats and modulation schemes [1]. Fortunately, Galileo applies a so-called Open Service Navigation Message Authentication (OSNMA) mechanism to protect its signals/navigation data against spoofing attacks, while the other three still have no such protection, thereby making Galileo a superior choice from the perspective of security [2].

OSNMA is the only commercial GNSS navigation message authentication (NMA) scheme, which allows Galileo receivers to verify the authenticity of navigation data based on message authentication code (MAC) [3], [4], and has been integrated by numerous worldwide GNSS receiver manufacturers (e.g., Septentrio) into their products. At the core of OSNMA

is the Timed Efficient Stream Loss-Tolerant Authentication (TESLA) protocol [5], which computes a tag for each navigation data using a key obtained from a global key chain called TESLA chain. The navigation data is broadcast immediately to receivers, while the tag and key are disclosed sequentially later. Only after receiving the corresponding tag and key, can a receiver authenticate the navigation data. Note that each key on the chain is generated from its previous key by a one-way hash function and the key disclosure process is in the reverse order of the key generation. This ensures that the authenticity of a newly-disclosed key can be easily verified if it successfully generates a previously-disclosed valid key through recursive one-way hash calculation, thus preventing attackers from deducing subsequent keys based on the newly-disclosed key. To ensure the security of the TESLA protocol, each key is associated with a Galileo System Time (GST), which can be extracted from the received signals and allows receivers to obtain the order of the keys. In addition, receivers must comply with a mandatory time synchronization (TS) rule, which specifies that the difference between a receiver’s local reference time (LRT) and the GST must not exceed a given threshold. The keys that do not comply with the TS rule are viewed as outdated and will be discarded. The above mechanisms of delayed key disclosure, navigation data verification, key verification and TS rule together empower Galileo with the ability to prevent signals/navigation data from spoofing attacks [2], [6].

Although OSNMA is designed to resist spoofing attacks, it still relies on MAC, which means that as long as the navigation data, tag and key are authentic and correlated (i.e., the key is from the TESLA chain and the tag is computed based on the data and key) provided the TS requirement is guaranteed, the authentication will be successful no matter where the data, tag and key originate. This may leave a door for spoofing attacks, because once attackers force receivers to comply with the TS requirement somehow, they can transmit replayed/forged signals that contain authentic and correlated navigation data, tag and key to pass the OSNMA mechanism. This paper therefore delves into the details of Galileo OSNMA mechanism, analyzes its vulnerabilities and explores the possibility of conducting spoofing (i.e., replay and forgery) attacks against commercial Galileo receivers with OSNMA functionality.

Some initial work has been done on signal spoofing attacks against OSNMA, for example, the distance-decreasing (DD) attack in [7], the idea of which is to reduce the estimated pseudorange of a victim receiver by recording and replaying Galileo signals in real time without modifying the naviga-

tion messages; the proximity-based attack in [8], where the proximity ensures the attacker and victim receiver obtain the same navigation messages and thus it only needs to manipulate the code phase and Doppler frequency shift of the signals to achieve successful attack; and the post quantum cryptography (PQC)-based attack in [9], which directly targets the cryptographic algorithms used in OSNMA. Although the above works demonstrate the possibility of attacking OSNMA, none of them performs in-depth analysis of OSNMA and its vulnerabilities. This might be the reason why these attacks focus on replay attacks at signal level without proposing more advanced forgery attacks at message level. Besides, the DD attack is tested on a software receiver, the proximity-based attack is tested on a receiver without OSNMA functionality and the PQC attack is not even testable, which makes the feasibility of these attacks in real-world environments with OSNMA-integrated receivers questionable.

In this paper, we perform in-depth investigation of the Galileo OSNMA mechanism with an emphasis on its core security components including tag generation and verification, key signature and verification, and TS rule. Based on the investigation, we analyze the possible vulnerabilities of OSNMA, and implement several practical spoofing attacks on a commercial OSNMA-integrated receiver. This paper extends the conference version presented at ION GNSS+ 2023 [10] by combining LRT falsification and replay attacks and further implementing a more sophisticated forgery attack. Due to official updates of OSNMA, all the attacks proposed in [10] have been re-experimented in this paper. To the best of our knowledge, this is the first work that provides in-depth analysis of OSNMA security and successfully implements both replay and forgery attacks against Galileo's OSNMA in real-world environments. The main contributions of this paper are summarized as follows.

- **OSNMA vulnerabilities.** We discover two critical vulnerabilities on OSNMA, namely artificially-manipulated time synchronization (ATS) and interruptible message authentication (IMA). ATS enables attackers to manipulate the received signals and/or LRT of a victim receiver such that the GST extracted from the signals and the LRT meet the mandatory TS requirement. IMA allows the navigation data authentication process to be temporarily interrupted due to the reception of a broken message and resumed later upon the reception of a complete and correct message that contains valid data, tag and key no matter where the message originates (authentic Galileo satellites or attackers). These vulnerabilities lay the foundation for the replay and forgery attacks proposed in this paper and are anticipated to inspire more advanced attacks against Galileo's OSNMA mechanism.
- **Replay attacks on cold-start receivers.** Targeting cold-start receivers that are not tracking any Galileo authentic signals, we propose a TS-comply replay (TSR) attack with two variants (i.e., real-time and non-real-time) by exploiting only the ATS vulnerability. In the real-time TSR attack, attackers record authentic signals first and immediately replay them to a victim receiver while

strictly complying with the TS rule. However, in the non-real-time TSR attack, attackers transmit previously-recorded signals to the victim receiver, and, to ensure successful attacks, the attackers also falsify the LRT of the victim receiver to comply with the TS rule with respect to the GST extracted from the replayed signals.

- **Forgery attacks on cold-start receivers.** We further propose a TS-comply forgery (TSF) attack by exploiting the ATS vulnerability, where attackers 1) forge navigation data based on the location to which they want to deceive the victim receiver; 2) compute valid tags for the forged navigation data by using a previously-disclosed key; 3) assemble the forged navigation data, tags and key into three successive navigation messages, respectively, based on the OSNMA format and Galileo I/NAV message format; 4) falsify the victim receiver's LRT as done in the non-real-time TSR attack; and 5) transmit the forged messages to the victim receiver. The TSF attack can spoof a victim receiver to arbitrary locations in theory, expanding the spoofing range of the replay attacks.
- **Replay attacks on already-tracking receivers.** Targeting receivers that are already tracking authentic signals, we propose a concatenating replay (CR) attack by exploiting both the ATS and IMA vulnerabilities. In this attack, attackers replay signals in real time (also complying with the TS rule) in a way that concatenates replayed messages to the currently-receiving message of the victim receiver. The concatenation will destroy the currently-receiving message first, thereby suspending the navigation data authentication process, and the later-replayed signals will successfully restore the authentication process because they contain valid data, tags and key.
- **Real-world attack implementation.** We conduct real-world experiments with a commercial OSNMA-integrated receiver manufactured by Septentrio, two software-defined radio (SDR) devices to test the TSR and CR attacks. In addition, we use the Galileo-SDR-SIM [11] software combined with a SDR as the generator of forged signals and use the GNSS-SDR [12] (as a position solver) combined with the OSNMAlib [13] (an official implementation of OSNMA protocol) as a software receiver to test the TSF attack¹. Please refer to our open-source project in [14] for the implementation details of all the attacks. Experimental results showed that all the attacks can successfully pass the OSNMA scheme and manipulate the position solution of the victim's receiver, and the TSF attack can spoof the receiver to arbitrary locations.

The rest of the paper is organized as follows. In Section II, we review the related work on GNSS signal spoofing attacks. In Section III, we analyze the OSNMA mechanism in detail and discover the ATS and IMA vulnerabilities. Section IV introduces the proposed TSR, TSF and CR attack schemes respectively. Section V demonstrates the implementation of these attacks in real-world environments and analyzes the

¹The reason of creating a software receiver is that the signals generated by Galileo-SDR-SIM cannot be recognized by the Septentrio receiver.

experimental results. Finally, Section VI concludes the paper.

II. RELATED WORK

We divide the work related to spoofing attacks on OSNMA into two categories: forgery attacks and replay attacks, with a primary focus on work conducted in real-world environments.

1) Forgery Attacks: Forgery attacks on OSNMA are currently unexplored, while those on GNSS signals without integrated encryption mechanisms are widely studied. Due to the openness of the civilian GNSS signal modulation format, researchers developed open-source GNSS signal simulators, such as *GPS-SDR-SIM* [15] and *Galileo-SDR-SIM* [11], coupled with the low cost of software-defined radios (SDR), which makes forgery attacks against GNSS signals extremely easy to implement. For example, researchers at the Mobile Security of Alibaba Group successfully spoofed the time and location of smartphones and smartwatches using open-source software and SDR [16]. Similarly, in [17], the locations of cell phones and a DJI Matrice 100 Quadcopter were successfully spoofed by generating GPS signals via SDR.

The forged signals generated by the above schemes may be relatively easier to detect, because they differ significantly from the authentic signals actually received by the receiver. To make the forgery attacks more stealthy, Humphreys *et al.* proposed a scheme to place a portable receiver-spoofing near the antenna of the intended victim receiver. The spoofing can accurately estimate the knowledge of the target receiver antenna's position and velocity. Based on these estimates, the spoofing aligns the forged signal correlation peaks with those corresponding to the authentic signal and gradually increases the power of the forged signal, eventually causing the receiver to lock onto the forged signal. [18]. Similarly, researchers at Carnegie Mellon University generate forged signals that are synchronized with the code phase of legitimate GPS satellites allowing for more stealthy attacks [19].

In summary, due to the public modulation of GNSS signals and the absence of authentication mechanisms, attackers usually use predefined (static or dynamic) positions to generate the corresponding signals, granting them significant flexibility in performing spoofing attacks. Further, as researchers delve deeper into stealthier attacks, they become more difficult to detect. However, since the forged signals lack correct OSNMA data, receivers integrated with the OSNMA mechanism can readily detect such signals. In this paper, we propose a novel attack scheme to spoof receivers by utilizing the OSNMA mechanism to forge navigation messages containing OSNMA data, thereby spoofing the receiver's position and time, even if the receiver is integrated with the OSNMA mechanism.

2) Replay Attacks: Many researchers have conducted experiments on replay attacks in real environments. Note that OSNMA begins testing in 2021, and the work before this mainly focuses on the extension of replay attacks. For example, researchers from the KTH Royal Institute of Technology conducted experiments based on two colluding adversaries [20]: one for recording authentic GNSS signals which are a mixture of all GNSS signals at the same frequency including GPS, Galileo, etc., and the other for replaying these signals at a different location via networks.

Replaying GNSS signals over long distances requires a large network bandwidth, incurring a non-negligible overhead. To mitigate this overhead, the researchers in [21] opted for a message-level replay attack, where GNSS messages (decoded from authentic GNSS signals) instead of signals are transmitted via networks. GNSS signals are regenerated and replayed at the remote location based on the replayed messages. Experimental results show that the message-level replay attack requires a network speed of only 15 KB/s for successful replay.

While the replay attacks in [20] and [21] require two colluding adversaries, the researchers in [22] proposed a replay attack that requires only one adversary. The idea is that the adversary manipulates the arrival time of different satellite's signals by introducing an appropriate delay, calculated based on a target position, and then replays the delayed signals to a local victim receiver. However, this scheme requires separating the signals of each navigation satellite and accurately calculating the target position to introduce corresponding delays, which is an extremely complex task in a real-world environment. Therefore, they simulated the experiments using GPS-SDR-SIM. The same delay-and-replay idea was employed in [8], where authentic signals from different directions were delayed by varying amounts. Again, it is still unable to physically separate the satellite signals and instead implement spoofing attacks by manipulating the code phase and Doppler frequency shift, where the navigation data from the legitimate satellite signals are not modified. The attack was tested on both a commercial receiver (ublox M8N) and GNSS-SDR [12], which is an open-source software-defined GNSS receiver, whereas either the commercial receiver or the GNSS-SDR did not implement the OSNMA mechanism.

Zhang *et al.* [7] proposed a variant of replay attack, called distance-decreasing (DD) attacks which estimates the early fraction of the bit/symbol period from Galileo signal and in parallel initiate bit/symbol transmission to reduce the computed pseudorange of the victim receiver by an amount that corresponds to a fraction of a bit/symbol. However, DD attacks introduce bit errors to the spoofing signals and any bit error will cause OSNMA to fail. The researchers assumed that the adversary configures parameters to minimize DD attacks induced errors, and used GNSS-SDR to conduct attack experiments.

Although the aforementioned researchers have advanced the replay attack to some extent, they did not conduct attacks on the receiver integrated with OSNMA, thus the effectiveness of the attacks cannot be determined.

III. OSNMA VULNERABILITY ANALYSIS

This section analyzes the OSNMA mechanism in depth and introduces the two vulnerabilities, i.e., ATS and IMA.

A. OSNMA Basics

OSNMA is based on the TESLA protocol, which authenticates navigation data through two mechanisms, i.e., MAC and delayed disclosure of secret keys. A MAC is a tag that is computed over the navigation data using a secret key and

used to verify the authenticity of the data. The key is obtained from the global one-way TESLA chain and disclosed to ground receivers a pre-defined time delay later after it is used. The delayed key disclosure prevents attackers from obtaining the key immediately, thereby ensuring the authenticity of navigation data [2], [23], [24].

1) *Galileo E1-B I/NAV message format:* The navigation data, tags and keys are transmitted in the Galileo E1-B I/NAV messages, which are grouped into frames. Each frame consists of 24 subframes and each subframe contains a complete navigation data, a key and several tags. To implement the delayed key disclosure, the tags of the navigation data transmitted in the i -th subframe are transmitted in the $i+1$ -th subframe while the key is transmitted in the $i+2$ -th subframe. Each subframe has a duration of 30 seconds, which means that a navigation data is authenticated about 60 seconds later after it is received by receivers.

A subframe is further divided into 15 pages. Page is the minimum reception unit of Galileo messages and contains a portion of the navigation data, key and tags. More concretely, each page consists of two parts, i.e., an even and odd part, as illustrated in Fig. 1. The navigation data portion is stored in the *Data* fields of both parts while the tag and key portions are stored in the *OSNMA* field of the odd part. The *Data* field contain the parameters (e.g., Ephemeris parameters, time and clock correction parameters, ionospheric parameters) required for receivers to solve a position. The *OSNMA* field contains a 8-bit *HKROOT* data, which stores a portion of the root key of the TESLA chain, and a 32-bit *MACK* data, which stores a key portion and several tag portions. After receiving a complete subframe, a receiver can combine the *Data* (resp. *MACK*) fields of all the 15 pages to generate a complete navigation data (resp. key and tags).

2) *GST and subframe organization:* GST is the time reference used by the Galileo satellites and ground stations. It is the basis for the timing signals transmitted by the Galileo satellites, which are then used for positioning and synchronization purposes. GST is transmitted in certain pages periodically and used by a receiver to organize subframes. Subframe organization is performed in a round-by-round manner. A receiver keeps receiving pages from the satellites and stamps each page with a time based on the GST. After receiving 15 pages, the receiver organizes them into a subframe. We call this a round, which lasts for 30 seconds. If some pages are destroyed in a certain round, the receiver still tries to parse this incomplete subframe while the OSNMA mechanism will discard all the pages received in this round. From this aspect, each subframe is associated with a GST and thus the navigation data, tags and key inside are all stamped by this GST. GST plays a critical role in the OSNMA mechanism, for example, checking the validity of the keys and checking the TS rule, as introduced in the following subsections.

3) *TESLA chain:* The TESLA chain stores all the keys used for generating tags. This chain starts with a random seed key K_N , known only to the OSNMA provider, and ends with the root key K_0 . Each key (say K_n , $n \in 0, 1, 2, \dots, N$) on the chain is computed from its previous key (i.e., K_{n+1}) through a hash function, i.e., $K_n = \text{Hash}(K_{n+1})$. This

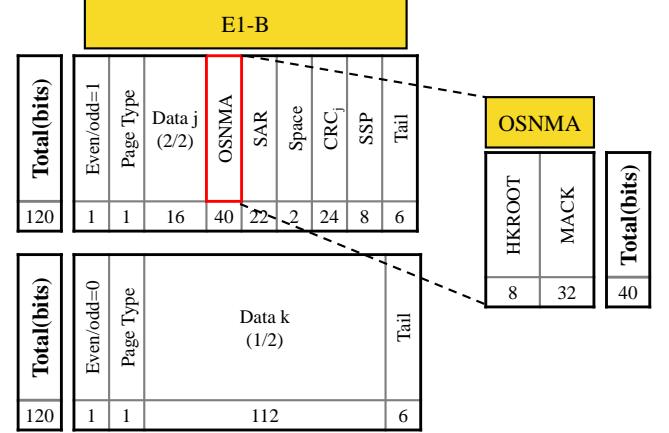


Fig. 1: Structure of a E1-B I/NAV Page.

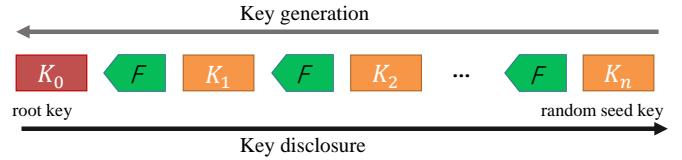


Fig. 2: TESLA chain.

concept is further illustrated in Fig. 2. The root key is used and disclosed first, followed by K_1, K_2, \dots, K_N . This allows easy verification of every received key by checking if it can successfully hashes back to the root key. The only problem here is to ensure the authenticity of the root key. The *HKROOT* field contains a portion of the root key *KROOT* and the portions of other parameters required for verifying the root key, such as *KROOT Week Number (WNK)*, *Time of Week (TOWK)*, *MF*, *NMAHeader*, etc. After receiving several successive subframes, a receiver can concatenate the *HKROOT* fields of all the pages of these subframes to obtain a complete root key and other parameters. By concatenating the root key and the parameters, a message M is generated as follows:

$$\begin{aligned} M = & NMAHeader || \dots || MF || \dots || WNK || TOWK || \dots \\ & || KROOT, \end{aligned} \quad (1)$$

where *NMAHeader* specifies the status of the OSNMA service, such as the availability of OSNMA. *WNK* and *TOWK* provide the time associated with a root key referred to GST, *MF* specifies the MAC function for generating tags. For simplicity, a few fields have been omitted. Please see the OSNMA documentation [23] for the detailed parameters. OSNMA uses the Elliptic Curve Digital Signature Algorithm (ECDSA) to sign the message M , and the generated signature data is also stored in *HKROOT*. Receivers can use the public key obtained from the OSNMA server to verify M , thereby ensuring the authenticity of the root key and its start time.

As the number of disclosed keys increases, hashing back to the root key incurs a non-negligible overhead, and thus the latest verified key can be used as the reference instead, which depends on the performance of the receiver. Note that each key is associated with a GST. Thus, the number of hash

calculations n required for verifying the i -th disclosed key K_i against the latest verified key K_{lv} ($lv < i$) depends on their GST difference, that is,

$$n = \frac{GST_i - GST_{lv}}{\Delta T}, \quad (2)$$

where GST_i (resp. GST_{lv}) represents the GST of the subframe containing K_i (resp. K_{lv}), and ΔT represents the subframe duration (i.e. 30 seconds). We can see that GST achieves some kind of synchronization between a subframe and the key inside, which is critical for the verification of secret keys. If the GST of a subframe does not match the key inside (possibly due to an adversary using a previously-disclosed key), the key verification will fail. In addition, the key disclosure process is in the reverse order of the key generation, which ensures that adversaries cannot derive a key that has not been disclosed yet from the newly-disclosed key.

4) *Tag generation and verification:* OSNMA divides a navigation data into multiple segments², and computes a tag for each segment using the same key. We use S_i to denote the i -th segment of a certain navigation data. During the computation, OSNMA adds some additional information to S_i to generate a message m_i as follows:

$$m_i = (PRN_D || PRN_A || GST_{SF} || \dots || S_i || \dots), \quad (3)$$

where PRN_D represents the satellite transmitting the navigation data, PRN_A represents the satellite transmitting the authentication information, GST_{SF} is the GST of the subframe in which the tag is transmitted. The distinction between PRN_A and PRN_D signifies OSNMA's capability to authenticate navigation data from both the Galileo and other systems.

Next, OSNMA takes the first unused key (say K_n) from the TESLA chain and computes a tag tag_i for m_i using the MAC function MF specified by the *HKROOT* field as follows:

$$tag_i = MF(K_n, m_i). \quad (4)$$

The tags of all segments will be concatenated as a whole and further split into 15 portions. Each portion is stored in the *MACK* field of one page, as illustrated in Fig. 1.

Once a receiver receives the navigation data, tags and key K_n , it can initiate the tag verification process by computing tags following the same procedures as described above. If the locally-computed tags match the received ones, the navigation data transmitted by the satellite PRN_D is authentic.

5) *The overall OSNMA authentication process:* Fig. 3 depicts the overall OSNMA authentication process.

- *Navigation data, key and tag reception:* The receiver retrieves the navigation data and the corresponding OSNMA data (tags, key and root key, etc.) from the received Galileo signals.
- *Root key verification:* The root key is authenticated by means of its digital signature using the public key.
- *Key verification:* The receiver authenticates the received key with the root key or with a previously authenticated key by using the specified hash function.

- *Tag verification:* The receiver locally re-generates the tags with the verified key and the navigation data, and checks whether they match the received tags.

More details about the key and tag verification can be found in [2], [24]. This paper focuses more on the security vulnerabilities in the OSNMA authentication process.

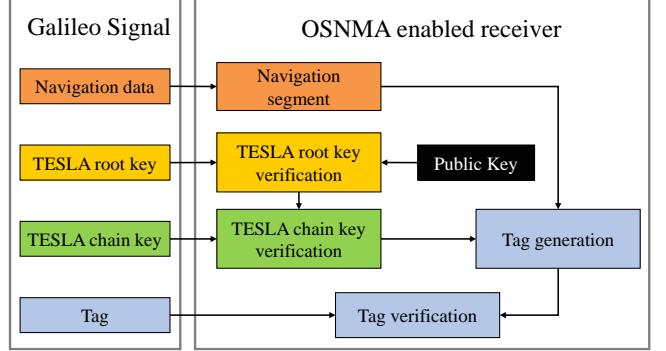


Fig. 3: OSNMA processing logic.

B. ATS Vulnerability

To ensure the security of TESLA protocol and the authenticity of navigation data, OSNMA stipulates that receivers must meet the mandatory TS requirement, which specifies that each receiver must be loosely time-synchronized to the GST before receiving and processing OSNMA information [2], i.e., the difference between the true time obtained at the receiver and the GST obtained from satellites must not exceed a threshold T_L , which is usually less than the interval of a subframe (i.e., 30 seconds). Only when the TS requirement is satisfied can the OSNMA mechanism initialize and authenticate navigation data. Formally, the TS requirement is given by

$$t_{true} - t_{GST} < T_L, \quad (5)$$

where t_{true} represents the true time and t_{GST} represents the GST.

Note that the true time is a virtual concept and is not available in practice. What is actually available to a receiver is the LRT, denoted as t_{LRT} , which can be obtained from an internal real-time clock or a network time protocol (NTP) server. In general, error exists between the LRT and the true time, which may be accumulated after the long run of an internal crystal oscillator clock or caused by network latency. OSNMA assumes that the error is bounded and the true time should be in the following confidence interval:

$$t_{true} \in [t_{LRT} - B, t_{LRT} + B], \quad (6)$$

where B denotes the error bound.

Introducing the error bound B makes the TS requirement in Eq. (5) difficult to apply and thus OSNMA adopts a more practical requirement based on the LRT, which is given by

$$t_{GST} \in [t_{LRT} - B, t_{LRT} + B], \quad (7)$$

or equivalently

$$|t_{GST} - t_{LRT}| < B. \quad (8)$$

²A segment is not equivalent to a portion.

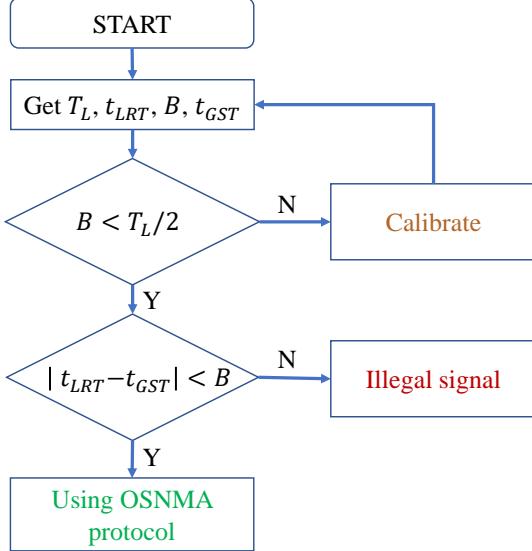


Fig. 4: TS startup flow diagram.

To comply with the requirement in Eq. (5), B should be less than or equal to $T_L/2$. Every time the receiver performs TS, it first checks whether the error bound B exceeds $T_L/2$. If yes, the receiver reckons that the LRT is untrustworthy and will not perform the TS. Instead, it will consider calibrating the LRT. If B is less than $T_L/2$, the receiver further checks whether Eq. (8) is satisfied. If yes, the OSNMA starts up. Otherwise, the received signal is considered illegal. For receivers that obtain the LRT from NTP servers, the error bound check can be omitted because the time provide by NTP servers are considered as extremely close to the true time. For receivers that obtain the LRT from an internal crystal oscillator clock, the error bound is checked against some previous LRT that has been calibrated. The TS process is depicted in Fig. 4.

Provided the LRT satisfies Eq. (6), the modified TS rule in Eq. (7) achieves certain level of signal replay/forgery resistance, because any replayed signals are previously-transmitted signals whose GSTs are far behind the LRT very likely and thus violates the TS rule, and any forged signals use previously-disclosed keys whose GSTs are also very likely far behind the LRT. However, the TS rule cannot fully resist the replay and forgery attacks.

We can see from Eq. (7) that the capability of the TS rule to resist replay and forgery attacks depends heavily on the value of the bound B . The smaller the bound is, the more powerful the TS rule is in resisting replay and forgery attacks. As previously mentioned, error between the LRT and the true time cannot be eliminated and the bound B can never be zero. In addition, the value of B cannot be obtained accurately when a receiver attempts to perform OSNMA [25]. For example, a receiver that obtains its LRT from an internal crystal oscillator clock is offline for a long period, B should be given a large value when it is back online, taking into account the crystal oscillator error and environmental factors. The method to obtain and characterize B at startup is out of the scope of this paper, but it should be noted that accurate

determination the value of B remains a challenging issue for GNSS receivers [25]. If a receiver manufacturer encounters difficulties in setting B and instead opts for a more lightweight scheme, for example, using an alternate TS requirement of $t_{LRT} - t_{GST} < T_L$, the possibility of successful replay attacks will increase significantly.

We can also see from Eq. (7) that the LRT is another decisive parameter. If attackers can falsify the LRT and GST concurrently while still complying with the TS rule, then attacks can also be successful. For receivers that obtain the LRT from NTP servers, tampering with the LRT is particularly easy, because NTP servers are vulnerable to Man-in-the-Middle (MitM) attacks [26]–[28], in which an attacker places itself in between the client and server, selectively delaying packets or directly forging packets to tamper with the victim receiver's LRT. After falsifying the LRT, attackers can forge navigation data, use a previously-disclosed key to generate associated tags, encapsulate these data according to the Galileo navigation message format and transmit them out. As long as the GST of the forged signals and the falsified LRT satisfy Eq. (7), the forged signals can pass the authentication. Following the similar idea, we can also easily perform replay attacks.

In summary, OSNMA uses TS to ensure the security of the TESLA protocol and guarantee the authenticity of the navigation data. However, attackers can achieve TS by falsifying a receiver's signals and/or LRT. Such TS is not real TS, and we call it artificially-manipulated TS (i.e., ATS), as the name of the vulnerability stands.

C. IMA Vulnerability

Section III-A introduces the key and tag verification processes under normal circumstances where a receiver is located in an environment with extremely good visibility and keeps receiving complete subframes, i.e., the *Data* and *OSNMA* fields are all complete. However, in some situations, a receiver may not be able to receive complete messages, probably due to the reception of Galileo signals with poor quality. For instance, in situations like tunnels or areas with high-rise buildings, a receiver may not be able to track Galileo satellite signals, the subframe being received is destroyed, and the tag and key inside cannot be parsed. In these situations, the destroyed subframe will be discarded by the receiver [13], [29] and the authentication process will be suspended.

OSNMA currently does not apply any scheme to handle such authentication interruption but solely relies on key and tag verification as the main security mechanism. When an authentication interruption occurs, OSNMA does not stop running but continues to wait for the arrival of new complete subframes (e.g., when the receiver returns to an environment with good signal quality). Upon receiving at least three successive complete subframes, OSNMA can restore the key and tag verification processes. This is because the tags of the navigation data in the first complete subframe are in the second complete subframe and the key used to authenticate the navigation data is in the third complete subframe. Since the receiver has verified and stored the root key and previously authenticated key, verifying the root key again is unnecessary.

Instead, the receiver verifies the keys from the new subframes using the previously authenticated key, as outlined in Eq. (2), which reduces the calculation consumption of the receiver while ensuring a fast recovery of OSNMA.

The above handling of interrupted message authentication is reasonable, because in real environment the receiver will inevitably receive incomplete subframes. However, such a mechanism leads to a serious vulnerability of OSNMA, which we call IMA. IMA means that OSNMA allows temporary suspension of the authentication process probably due to the reception of broken navigation data, tag or key, and restores the authentication after receiving complete and valid subframes later. This opens a door for attackers as they can deliberately disrupt the receiver's subframe reception, suspending the authentication process and then append replayed signals that contain complete subframes to bypass the OSNMA.

Ideally, destroying the subframe being received is not necessary. Assuming that an attacker can replay signals with zero latency and seamlessly take over the receiver, the receiver does not lose subframes, and the replayed signal can directly pass OSNMA without causing a pause. However, implementing such an attack in a real environment is extremely difficult, because the replayed signals will inevitably introduce a certain delay, causing the receiver to lose lock and fail to receive the complete subframe.

IV. PROPOSED ATTACKS

The goal of attackers is to manipulate the position and time estimated by a victim receiver. Since authentic satellites have relatively low power, attackers have the capability to transmit strong spoofing signals that overpower the authentic signals, effectively burying them beneath the noise floor. Consequently, the victim receiver will lock onto and track the spoofed signals instead of the authentic ones. More specifically, when the victim receiver is undergoing a cold start, it will inherently track the spoofing signals. If the victim receiver is already tracking authentic signals, the spoofing signals will cause the receiver to lose lock and restart acquiring signals. During this process, the victim receiver acquires and tracks the spoofing signals. If the spoofing attacks are carefully designed by exploiting the ATS and IMA vulnerabilities introduced in Section III, the spoofing signals will bypass the OSNMA mechanism and lead to falsified position and time at the victim receiver. In what follows, we will introduce such spoofing attacks that we proposed under different scenarios.

A. Real-time TS-comply Replay (TSR) Attack

Exploiting the ATS vulnerability, we propose a real-time TSR attack that targets cold-start receivers. The basic principle of the attack is to continuously record and replay authentic Galileo signals in real time in a way that ensures the GST extracted from the replayed signals can satisfy the TS requirement in Eq. (7) with respect to the victim receiver's LRT.

As illustrated in Fig. 5, the real-timeside TSR attack involves two colluding adversaries, i.e., a local adversary that resides near the victim receiver and replays Galileo signals, and a remote adversary that resides at the location to which



Fig. 5: An overview of the real-time TSR attack scenario.

they want to deceive the victim receiver and receives authentic Galileo signals. These two adversaries are connected via a long-range cable, a local area network (LAN) or the Internet for transmitting signals. The real-time TSR attack is performed according to the following steps:

- 1) The remote adversary keeps receiving authentic signals from Galileo satellites.
- 2) The remote adversary immediately forwards the signals to the local adversary. The signals experience some delay before reaching the local adversary, typically caused by propagation delay in the cable or network latency.
- 3) The local adversary replays the signals immediately with a power higher than that of the authentic Galileo signals.
- 4) The victim receiver is powered on some time later. Since the receiver is not tracking any satellites, it can easily locks onto the replayed signals.

We assume that the victim receiver obtains the LRT immediately after startup. As long as the TS requirement in Eq. (7) is satisfied, the OSNMA will be successfully initiated at the receiver, as illustrated in Fig. 4. After the OSNMA starts up, the replayed signals can successfully pass the OSNMA authentication process as follows (see Fig. 6).

- 1) The receiver obtains the root key and auxiliary parameters from several (for example, 3 in Fig. 6) replayed subframes to finish the root key verification. Since the root key and parameters are all authentic and comply the TS requirement, the root key will successfully pass the verification process.
- 2) Using the verified root key, the receiver successfully authenticates the key in each replayed subframe through recursive hash calculations.
- 3) The navigation data in the first replayed subframe and the key in the third replayed subframe are extracted and used to compute several tags following the procedure in Section III-A4. The computed tags are successfully verified against the tags in the second replayed subframe. So far, the first replayed subframe has been successfully authenticated.
- 4) Following the same steps, the subsequently-replayed subframes can all be successfully authenticated.

Fig. 7 illustrates the real-time TSR attack in time scale, where t_0 represents the time when the signals are transmitted

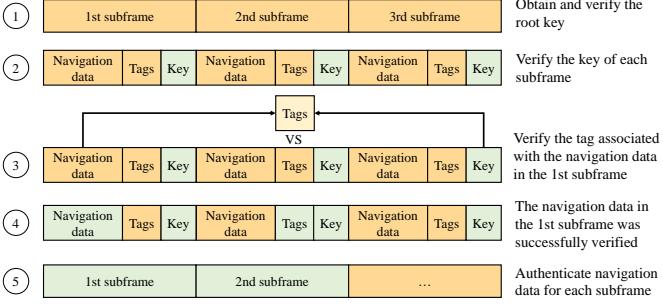


Fig. 6: OSNMA authentication of replayed signals.

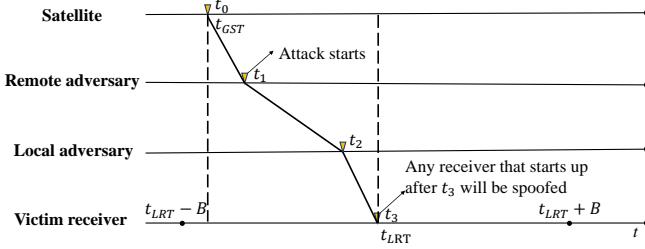


Fig. 7: Real-time TSR attack in time scale.

from the satellites, t_1 represents the time when the signals arrive at the remote adversary, t_2 represents the time when the signal arrives at the local adversary, and t_3 represents the time when the replayed signals arrive at the victim receiver. The difference $t_1 - t_0$ (resp. $t_2 - t_1$) represents the propagation delay of the signals from the satellites (resp. the remote adversary) to the remote adversary (resp. local adversary), and the difference $t_3 - t_2$ represents the processing time at the local adversary plus the propagation delay of the signals from local adversary to the victim receiver. We can see that t_0 is extremely close to the GST. For simplicity of analysis, we assume t_0 is equivalent to the GST. Note that t_3 is equivalent to the LRT at the receiver. This means that as long as $|t_0 - t_3| < B$ is satisfied, the attack will be successful. Suppose the real-time TSR attack starts at t_1 , any receiver in the vicinity of the local adversary that starts up after t_3 will be successfully spoofed.

B. Non-real-time TSR Attack

Non-real-time TSR attack is another type of TSR attacks, following the same principle of complying the TS rule. The key difference is that, in the non-real-time TSR attack, adversaries replay authentic Galileo signals that are recorded previously (e.g., one day ago) instead of real-time signals. To satisfy the TS requirement, adversaries must falsify the LRT of the victim receiver with respect to the GST of the replayed signals. This means that the non-real-time targets one specific receiver in most cases. Since the attack is not real time, one adversary is enough to implement the attack. In this case, the adversary can record the signals first and then move to the vicinity of the victim receiver to replay the recorded signals. The non-real-time TSR attack is performed as follows:

- An adversary records authentic signals from Galileo satellites at any moment t_{record} .
- The adversary forwards the signals to another local adversary or moves directly to the vicinity of the victim receiver. Undoubtedly, the GST of the signals is far behind the LRT of the victim receiver.
- The adversary (or the local adversary) falsifies the victim receiver's LRT to t_{record} in order to satisfy the TS rule and then replays the record signals.

The way of falsifying the receiver's LRT depends on how the receiver obtains the LRT. We take the most widely used NTP protocol as an example. NTP is designed to synchronize the time of machines in a network [30], [31]. GNSS receivers can use this protocol to obtain an accurate LRT from NTP servers. However, this protocol is vulnerable to the MitM attack [27], [28]. For example, the adversary can delay the request packets and deliver the response packets promptly thereby arbitrarily delaying the time obtained by the victim receiver. Using this method, the attacker can accurately falsify the LRT of the receiver based on the difference between t_{record} and the authentic NTP server time to fulfill the TS requirement.

C. TS-comply Forgery (TSF) Attack

The non-real-time TSR attack demonstrates the feasibility of using previously-recorded signals to conduct replay attacks. However, its capability is limited, because receivers can be spoofed to only the locations and time determined by the navigation data of the previously-recorded subframes. One may ask: *can adversaries just replace the original navigation data and tags inside the previously-recorded subframes with forged ones while keeping the keys, so that they can spoof the receivers to any place and any previous time?* The answer is yes and this is exactly what the TSF attack does.

In general, a receiver determines its position (x_r, y_r, z_r) and clock offset t_r based on the coordinates of more than four satellites and its pseudoranges to these satellites [32], [33]. This process can be described mathematically by the following system of equations:

$$\begin{cases} \sqrt{(x_r - x_1)^2 + (y_r - y_1)^2 + (z_r - z_1)^2} + c \cdot t_r = \rho_1 \\ \sqrt{(x_r - x_2)^2 + (y_r - y_2)^2 + (z_r - z_2)^2} + c \cdot t_r = \rho_2 \\ \sqrt{(x_r - x_3)^2 + (y_r - y_3)^2 + (z_r - z_3)^2} + c \cdot t_r = \rho_3 \\ \sqrt{(x_r - x_4)^2 + (y_r - y_4)^2 + (z_r - z_4)^2} + c \cdot t_r = \rho_4 \end{cases} \quad (9)$$

where (x_i, y_i, z_i) denotes the i -th ($i = 1, 2, 3, 4$) satellite's coordinate and ρ_i denotes the pseudorange from the receiver to this satellite. Since the satellite clock and the receiver clock are not synchronized, the parameter t_r is introduced (this value is fixed for the receiver) here to represent the offset between the receiver clock and the GNSS system time. The coordinate (x_i, y_i, z_i) is obtained from the Ephemeris field in the navigation data. The pseudorange is measured based on the PRN code phase, which cannot be obtained from the navigation data but is calculated directly in the signal acquisition and track stage instead. The clock correction and ionospheric parameters in the navigation data are also used to correct some errors in the

pseudorange measurement. More on the principles of GNSS positioning can be found in [32].

According to Eq. (9), given a target position (x_r, y_r, z_r) to which the adversaries want to spoof the victim receiver, to achieve the TSF attack, the adversaries need to select at least four satellites (say target satellites) and then forge the navigation data (Ephemeris, clock correction and ionospheric parameters) and generate signals for each satellite in a way that can produce the coordinates (x_i, y_i, z_i) and pseudoranges ρ_i as specified by Eq. (9). Of course, the Doppler effect caused by the satellite movement also needs to be considered [11]. Forging navigation data, mimicking Doppler effect and generating signals can be done by using GNSS signal simulators, such as the commercial GNSS signal simulator Spectracom GSG-64. However, the simulated signals do not contain OSNMA data and thus fail to pass the OSNMA authentication. The TSF attack addresses this issue by adding valid OSNMA data (tags) to the forged navigation data.

Next, we introduce the TSF attack in greater details. To spoof the victim receiver to a target position (x_r, y_r, z_r) and a target time t , the adversaries need an old key that is used to generate tags for the forged navigation data transmitted at t . Note that this key was disclosed at $t + 2\Delta T$ to comply with the TS rule. We assume that adversaries have obtained a valid subframe that contains this key before conducting the TSF attack. We call this subframe the auxiliary *key* subframe. We also assume that two subframes proceeding the auxiliary key subframe are also available. We call the first one the auxiliary *data* subframe and the second one the auxiliary *tag* subframe, where the former (resp. the latter) is used to store the forged navigation data (resp. tags). With the forged Ephemeris, clock correction parameters, ionospheric parameters and the auxiliary subframes, the adversaries perform the following steps for each target satellite to realize the TSF attack. For simplicity of explanation, we take one target satellite as an example.

1) *Forging navigation data of auxiliary data subframe:* The adversaries use the forged Ephemeris, clock correction parameters, ionospheric parameters of the satellite to replace the corresponding fields in the navigation data of the auxiliary data subframe. Note that the GST-related parameters in the auxiliary data subframe will be kept to ensure successful key verification.

2) *Forging tags of auxiliary tag subframe:* The adversaries generate tags for the forged navigation data using the key extracted from the auxiliary key subframe. As introduced in Section III-A1, the key can be obtained by concatenating the *MACK* fields of the 15 pages of the key auxiliary subframe. In addition, the MAC function *MF* is also needed, which we assume has been obtained by the adversaries ahead of the attack. Given the key and MAC function, the adversaries divide the forged navigation data into multiple segments and computes a tag for each segment. The adversaries replace the original tags in the auxiliary tag subframe with the forged tags.

3) *Forging subframe:* The adversaries perform a cyclic redundancy check (CRC) for each page of the auxiliary data subframe that contains the forged navigation data and fill the CRC data into the corresponding *CRC* fields to prevent the

receiver from detecting bit errors. So far, the adversaries have forged a subframe with forged navigation data and placed the corresponding forged tags in the next subframe (auxiliary tag subframe).

4) *Forging signals:* Given the target position and the forged subframe (more exactly, the forged Ephemeris), the adversaries forge the pseudoranges based on Eq. (9). Taking the forged ionospheric parameters and the speed of light into consideration, the adversaries then estimate the PRN code phases and apply the code phases to generate baseband signals. More concretely, the adversaries modulate the forged navigation subframe into a baseband signal, mix it with the PRN code, upconvert the mixed signal onto the Galileo E1 frequency band and send the upconverted signals out. As mentioned above, this can be done with the help of a GNSS signal simulator.

The above steps show the process of forging signals of 30 seconds (i.e., the duration of a subframe) for one target satellite. Executing the above steps for all target satellites will result in a 30-second TSF attack. If the adversaries want to conduct the TSF attack continuously, they can use successive subframes and repeat the above steps. Algorithm 1 shows the detailed process of continuous TSF attack based on a set of successive auxiliary subframes. Also, we focus on one target satellite. Note that since the TSF attack is TS-comply, falsification of the LRT of the victim receiver is also required as in the non-real-time TSR attack. Also, the keys of the auxiliary subframes must not be changed. Otherwise, when OSNMA uses Eq. (2) to verify the key, an incorrect number of hashes will lead to the failure of key verification. A TSF attacker does not need to record the signal itself, but only extracts the OSNMA data from the navigation message, thereby avoiding the resource overhead of signal forwarding.

D. CR Attack

The TSR and TSF attacks exploit the ATS vulnerability to implement attacks, targeting cold-start receivers. In this subsection, we introduce the CR attack, which exploits both the IMA and ATS vulnerabilities. The CR attack targets receivers that are currently tracking and locked onto authentic Galileo signals.

Different from cold-start receivers, GST is available at already-tracking receivers. This means that the CR attack must be conducted in real time due to the following two reasons. First, LRT falsification fails to work, because any significant modification of the time source will violates the TS rule. Second, non-real-time attacks will cause the timestamp of the replayed keys to fall behind the GST, leading to failed key verification. More exactly, the term “real-time” here means that if the victim is receiving the signals containing the i -th subframe, the adversaries must also replay the signals containing the i -th subframe. Note that these signals are different because they are received at different locations, containing different basic observables.

From the perspective of implementation, the CR attack follows the same procedure as those of the real-time TSR attack, while the basic principle is significantly different. The CR attack follows the idea of “*destroy-and-concatenate*”, as

Algorithm 1: Continuous TSF attack of one satellite

```

1 Prerequisite: MAC function  $MF$ , auxiliary subframe set  $SF_{aux} = \{sf_{aux,1}, sf_{aux,2}, \dots, sf_{aux,N}\}$ .
2 Input: Target position  $(x_r, y_r, z_r)$ .
3 Output: Forged subframe set  $SF_{forged}$ .
4 for  $n = 0$  to  $N - 2$  do
5   Forge a navigation data  $Data_n$  based on  $x_r, y_r, z_r$ ;
6   Divide  $Data_n$  into  $M$  segments  $S_1, S_2, \dots, S_M$ ;
7   Obtain the TESLA key  $K_{n+2}$  from  $sf_{aux,n+2}$ ;
8   for  $i = 1$  to  $M$  do
9     Generate an auxiliary message
10     $m_i = PRN_D \parallel \dots \parallel S_i \parallel \dots$ ;
11    Forge a tag  $tag_i = MF(K_{n+2}, m_i)$ ;
12   Replace the navigation data field of  $sf_{aux,n}$  with  $Data_n$ ;
13    $sf_{forged,n} = sf_{aux,n}$ ;
14   Replace the tags in the OSNMA field of  $sf_{aux,n+1}$  with  $tag_1, tag_2, \dots, tag_M$ ;
15   for  $j = 1$  to  $15$  do
16     Calculate and update the CRC data for the  $j$ -th page of  $sf_{forged,n}$ ;
17   Append  $sf_{forged,n}$  to  $SF_{forged}$ ;
18 return  $SF_{forged}$ ;

```

illustrated in Fig. 9. In the *destroy* phase, the replayed signals will jam the authentic signals that the victim receiver is concurrently receiving, destroying the subframe being received and suspending the OSNMA authentication process. As the replayed signals continuously arrive at the victim receiver, in the *concatenate* phase, the replayed signals will concatenate subframes that contain valid keys and tags to the previous undestroyed subframes, restoring the OSNMA authentication process due to the IMA vulnerability.

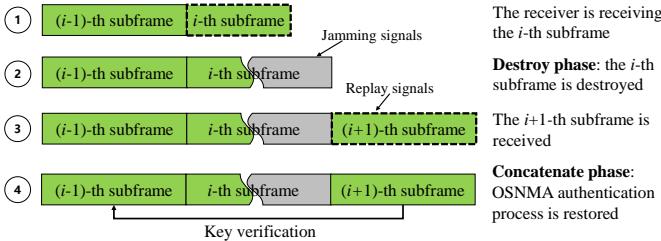


Fig. 8: Illustration of signal reception under CR attack.

Next, we introduce the CR attack in greater details. Suppose the victim receiver is currently receiving the i -th subframe and has obtained a GST t_{GST} in the previous subframe. We use P_i^j ($j = 1, 2, \dots, 15$) (resp. \hat{P}_i^j ($j = 1, 2, \dots, 15$)) to denote the j -th page in the i -th (resp. replayed) subframe. Suppose the victim receiver is receiving P_i^j when the replayed signals arrive. In this case, the pages $P_i^j, P_i^{j+1}, P_i^{j+2}, \dots, P_i^{15}$ will be destroyed by $\hat{P}_i^1, \hat{P}_i^2, \dots, \hat{P}_i^{16-j}$. We use \tilde{P}_i^j ($j = 1, 2, \dots, 15$) to denote the destroyed page. As introduced in Section III-A2, subframes are parsed in a round-by-round manner. This means that, the receiver will

organize the pages $P_i^1, P_i^2, \dots, P_i^{j-1}, \tilde{P}_i^j, \tilde{P}_i^{j+1}, \dots, \tilde{P}_i^{15}$ as the i -th subframe in this round and the pages $\hat{P}_i^{17-j}, \dots, \hat{P}_i^{15}, \hat{P}_{i+1}^1, \hat{P}_{i+1}^2, \dots, \hat{P}_{i+1}^{16-j}$ as the $i+1$ -th subframe in the next round. We can see that as long as j is larger than 1, the replayed signals will cause misaligned pages in all the subsequent subframes. In this case, the replayed subframes are all destroyed as well and the replay attack is meaningless but simplify jams the subframes of the victim receiver. Thus, the critical condition for a successful CR attack is that the replayed signals must arrive during the victim receiver is receiving the first page of a subframe.

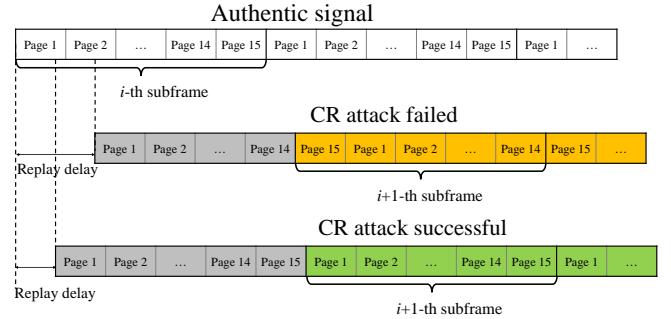


Fig. 9: Illustration of successful and failed CR attacks

Fig. 9 illustrates the cases of successful and failed CR attacks. Assume that the adversaries replay the signals immediately when they start receiving the first page. The time difference between the start of the first authentic page and that of the first replayed page represents the replay delay, which is dominated by the propagation latency between the remote and local adversaries. Note that the duration of a page is 2 seconds. This means that as long as the replay delay is less than 2 seconds, the CR attack will be successful. Replay via Internet represents the worst case with the longest latency, but 2 seconds are still enough for sending signals from the remote to the local adversary in most cases³.

V. ATTACK IMPLEMENTATION AND EVALUATION

In this section, we implement our proposed attacks. The devices used in this experiment are shown in Fig. 10, where two GNSS active antennas is used to receive and replay Galileo satellite signals and the USRP B210 device is used for signal processing. The Septentrio receiver is used as the victim receiver to evaluate the performance of the proposed attacks. This receiver implements all the mechanisms of OSNMA and relies on an NTP server to obtain its LRT. The GNSS time derived by the receiver is equivalent to GST and is used for TS. We observe the ONSMA status and position solution of this receiver to evaluate the feasibility of the proposed attack, and explore the conditions that need to be met for successful attack.

A. Real-time TSR Attack Implementation And Evaluation

To implement the real-time TSR attack, we use two SDR devices (USRP B210) as two adversaries: one (say remote

³Internet latency is usually less than a few milliseconds.



Fig. 10: Devices used to perform the attacks.

SDR) is connected to an antenna (say receiving antenna) to receive authentic Galileo signals in the E1 band, and the other (say local SDR) is connected to another antenna (say replaying antenna) to replay the signals. The configuration is illustrated in Fig. 11. Both SDR devices are connected to a common PC (Lenovo Y9000P), which is responsible for adding a certain delay to the replayed signals. The delay mimics the signal propagation delay between the two SDR devices and is used to study the effect of delay on the effectiveness of attacks. The delay is introduced based on GNU Radio [34].

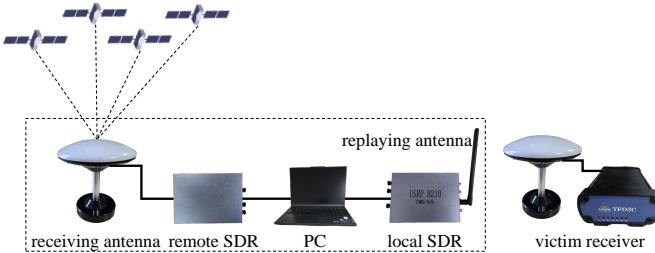


Fig. 11: Illustration of real-time TSR attack.

Fig. 12 illustrates the experiment results, which is part of the main page of the victim receiver (IP address: 192.168.3.1) accessed from a PC. The OSNMA status part shows some important information such as the current status (e.g., authenticating and init failed) of the OSNMA process, the time difference between the GST and LRT (i.e., Trusted Time Delta) and the time source (i.e., Trusted Time Source). We can see from Fig. 12 that when the delay exceeds 30 seconds, the receiver fails to achieve TS and does not initiate OSNMA. However, with a delay set to 29.5 seconds, the receiver successfully completes TS and proceeds with OSNMA. We infer from these observations that as long as the replay delay remains below 30 seconds, the real-time TSR attack satisfies the TS requirement and thus can be successfully conducted to spoof the receiver. The value of 30 coincides with the T_L of OSNMA, i.e., the duration of a subframe. Thus, we deduce that the victim receiver does not adopt the TS requirement in Eq. (7), and use an alternate instead, i.e. $t_{LRT} - t_{GST} < T_L$. A possible countermeasure to resist real-time TSR attack is to reduce the value of T_L or B , which, on the other hand, may increase the rate of false alarm, i.e., detecting authentic signals as replayed ones. For example, network congestion may cause large LRT errors, resulting in failure of TS between the authentic signal and the LRT.

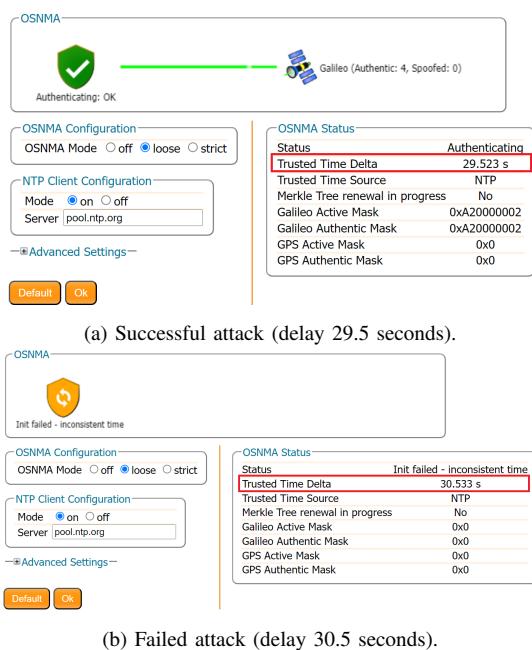


Fig. 12: Real-time TSR attack on the victim receiver.

B. Non-real-time TSR Attack Implementation And Evaluation

As illustrated in Fig. 13, the non-real-time TSR attack implementation involves a SDR device with two antennas. The SDR device is connected to a PC to record authentic signals at t_{record} . The recorded signals are then transmitted after a specified delay. The open-source tool Delorean [35] is used to tamper with the LRT of the victim receiver based on the idea of MITM attack. We delay the LRT of the victim receiver by $t_{LRT}-t_{record}$ seconds, while replaying the recorded authentic signal to ensure the TS requirement. Note that both the GST in the signal and the LRT of the receiver in this attack experiment are later than the true time, so we use a trusted and relatively accurate third-party time as a reference for the true time, which is the Universal Time Coordinated (UTC) obtained from the National Institute of Standard and Technology (NIST)⁴.

Fig. 14 illustrates the results of the attacks when the authentic signals are delayed by 32 seconds. We can know from the real-time TSR attack, a delay of 32 seconds will lead

⁴The website is <https://www.time.gov>

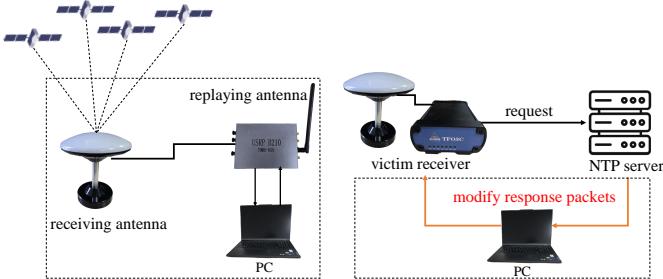


Fig. 13: Illustration of non-real-time TSR attack.

to failed attack without spoofing the LRT of the receiver, as shown in Fig. 14a. We can see from Fig. 14b that, when we use an NTP spoofing tool to delay the victim receiver's LRT by 32 seconds, the time difference between GST and LRT is 0.771 seconds (satisfying the TS requirement), the replayed signals successfully passed OSNMA authentication. Please note that, at the time of conducting the experiments, the GST was 18 seconds faster than UTC time. Thus, we can see that the GST extracted by the receiver from the replayed signals (2024-04-24 14:55:44) actually lags behind the third-party reference time ((2024-04-24 14:56:00) by near 32 seconds. This also shows the success of the attack.

C. TSF Attack Implementation And Evaluation

To implement the TSF attack, we use a PC to forge subframes and a SDR device to transmit the forged signals. Based on the steps introduced in Section IV-C, we conducted the TSF attack as follows.

- 1) We chose a target position (4° N, 50° E, 100 m) to which we want to spoof the victim receiver.
- 2) We used subframes from the ESA public test vector (16_AUG_2023_GST_05_00_01.csv) released by the EUSPA as the auxiliary subframe set [36].
- 3) We forged subframes for each satellite according to Algorithm 1 (Line 4 to Line 15). Note that the auxiliary subframes contain the Ephemeris data of 26 satellites from 05:00:00 to 06:00:00 on August 16, 2023, which can be treated as forged Ephemeris data (navigation data forging) at the time of conducting the experiments. In addition to Ephemeris data forging, we also modified the ionospheric parameter data of all the auxiliary subframes of each satellite. Appendix VI-A gives an example of forging the ionospheric parameter of the first auxiliary subframe for the satellite with PRN 2. Appendix VI-B and VI-C show the corresponding tag and subframe forging, respectively.
- 4) We used Galileo-SDR-SIM as the tool to forge signals, which takes the target position and the forged subframes (the auxiliary subframe set with modified ionospheric parameters) as input and uses the SDR device as the baseband signal processing module and radio frequency frontend to transmit the forged signals. In our experiment, Galileo-SDR-SIM uses the satellites of PRN 02, 03, 05, 09, 15, 18, 25 and 30 as the target satellites.

- 5) We used the open-source software GNSS-SDR as a software receiver to receive the forged signals and solve the positions, and used the open-source software OSNMAlib [13], [29], as the OSNMA module to report the authentication results. The GNSS-SDR and OSNMAlib together create a software Galileo receiver with OSNMA functionality. The reason of using a software receiver is that the signals generated by Galileo-SDR-SIM and USRP B210 cannot be tracked by the Septentrio receiver. GNSS-SDR first tracks the forged signal, parses the pages in the forged subframes from each satellite, and sends the pages to OSNMAlib for OSNMA authentication. Using the forged subframes, GNSS-SDR also solves the positions. Fig. 15 shows the verification process of the TSF attack.

All the scripts we used to forge navigation data, tags and subframes, and verify the attack can be found in our github project [14]. Note that OSNMAlib reserves an interface to obtain LRT using the NTP protocol, but this interface function is not implemented [13], so the time tampering attack used in TSR is omitted here.

To verify the effect of the tag forging, we divide the auxiliary subframe set into two groups. For one group, we forged only the navigation data (i.e., ionospheric parameters), and for the other group, we forged both the navigation data and the tags. Fig. 16 shows the OSNMAlib outputs for the two groups. We can see from Fig. 16a that OSNMA authentication fails when we only forged navigation data. Fig. 16b shows that OSNMA authentication succeeds when we forged both the navigation data and tags. For the case of successful OSNMA authentication, we also show the output of the GNSS-SDR in Fig. 17, including the satellite PRN number locked by GNSS-SDR, the parsed navigation data and the solved positions. We can see that the satellite PRNs are the same as those selected by the Galileo-SDR-SIM and the navigation data (ionospheric parameter data) we forged are correctly parsed. Also, the solved position is basically consistent with the position we forged, i.e., (4° N, 50° E, 100 m), which shows that our proposed attack can successfully spoof the receiver.

D. CR Attack Implementation And Evaluation

Before the CR attack, we started the Septentrio receiver up for a period of time to ensure it successfully acquires and tracks satellites to solve correct positions. After the receiver successfully acquires and tracks satellites, we used another SDR device (USRP B210) to transmit jamming signals in the E1 band to make the receiver lose lock on the authentic satellites, and then started replaying the Galileo signals. We replayed the signals using the same equipment as the real-time TSR attack and followed the same steps.

Fig. 18 illustrates the carrier-to-noise ratio of the victim receiver in the normal, jamming, and replay stages, where E15, E27, E30 and E34 represent the satellite PRNs. Note that, in our experiment, the replay attack is conducted locally, i.e., the adversaries are in the proximity of the victim receiver. Thus, the satellites seen by the receiver are the same in the normal and replay stages. We can observe changes of the carrier-to-

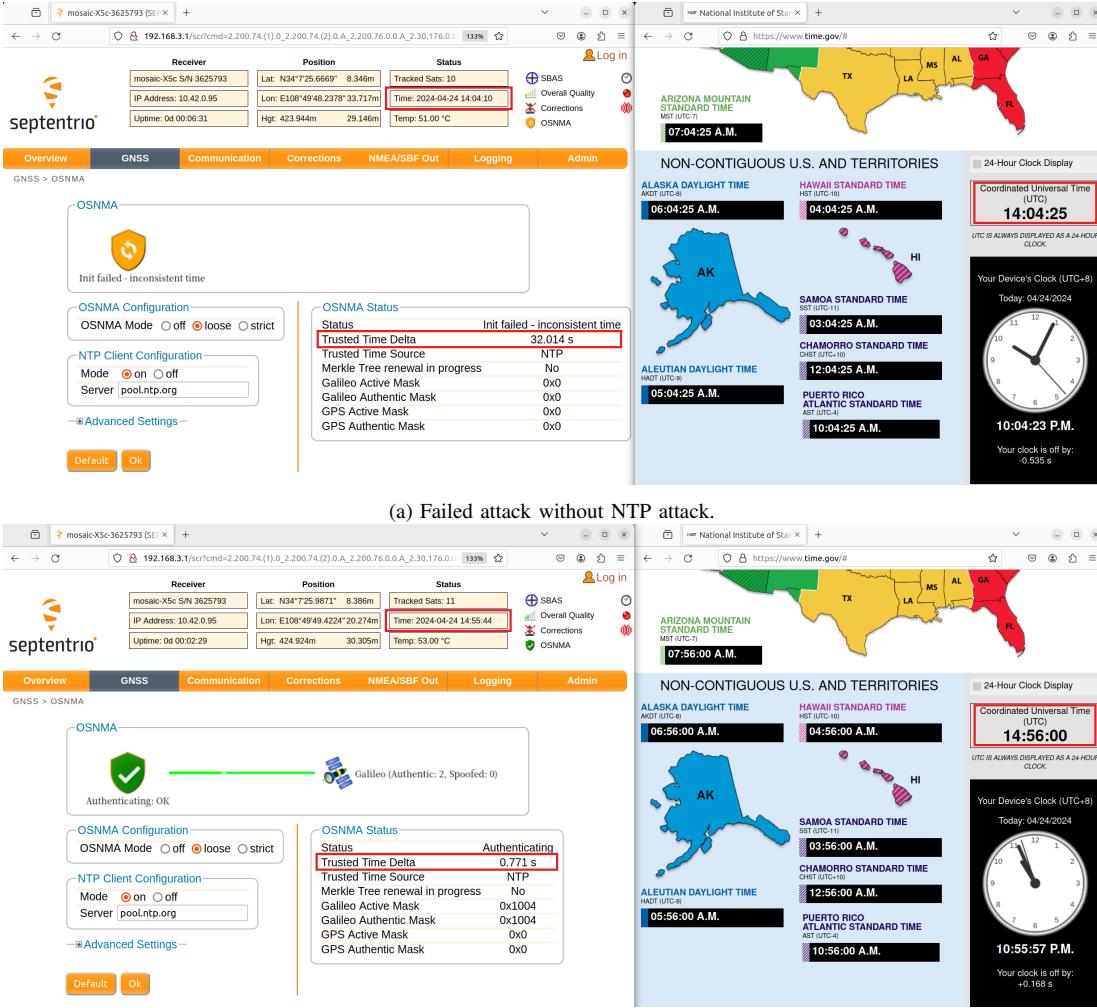


Fig. 14: Non-real-time TSR attack on the victim receiver (delay 32 seconds).

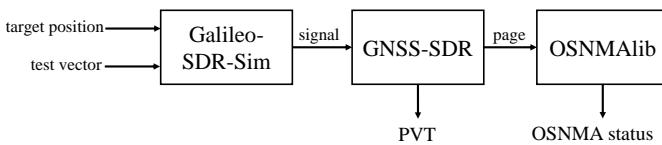


Fig. 15: The verification process of TSF attack.

noise ratio across different stages, implying that the victim receiver was successfully locked onto the replayed signals.

As analyzed in Section IV-D, the signals must be replayed within a delay of less than 2 seconds for the CR attack to be successful. For the Septentrio receiver, we conducted extensive experiments under different replay delays (around 2 seconds) to explore the impact of replay delays on attack effectiveness. We summarized the results in Fig. 19, which indicates that if the delay of the replayed signal is less than or equal to 1.4 seconds, the CR attack is successful and the OSNMA is passed (see Fig. 19a). However, once the delay exceeds 1.5 seconds, OSNMA identifies the replayed signals as spoofed signals, leading to the failure of the CR attack (see Fig. 19b).

```

INFO - Tesla Key Authenticated 1251 277260
INFO - MACSEQ AUTHENTICATED
PRN_A: 02 GST_SF: 1251 277230 FLX Tags: 0
INFO - Tag verification:
ERROR - Tag FAILED
(02, 00) PRN_A: 02 GST_SF: 1251 277230 COP: 15 TAG0
INFO - Data authenticated:
  
```

(a) Forging navigation data only.

```

INFO - Tesla Key Authenticated 1251 277260
INFO - MACSEQ AUTHENTICATED
PRN_A: 02 GST_SF: 1251 277230 FLX Tags: 0
INFO - Tag verification:
INFO - Tag AUTHENTICATED
(02, 00) PRN_A: 02 GST_SF: 1251 277230 COP: 15 TAG0
INFO - Data authenticated:
  
```

(b) Forging both navigation data and tags.

Fig. 16: Output of OSNMAlib.

```

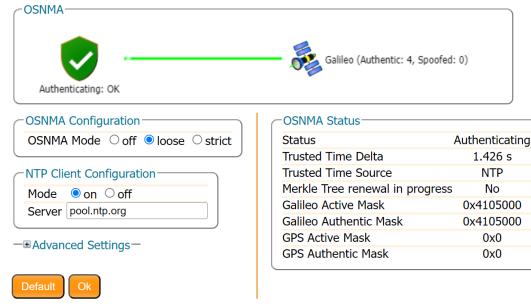
Position at 2023-Aug-10 01:00:49.499985 UTC using 6 observations. Lat = 1.599994074 [deg], Long = 49.99993717 [deg], Height = 85.099 [m]
Velocity: East: -0.003 [m/s], North: -0.003 [m/s], Up = 0.009 [m/s]
New Galileo E11 J/NAV message received to channel 1: UTC model parameters from satellite Galileo PRN E02 (Block FOC-FW11)
New Galileo E11 J/NAV message received to channel 2: Lono/GST model parameters from satellite Galileo PRN E03 (Block FOC-FW12)
New Galileo E11 J/NAV message received to channel 3: Lono/GST model parameters from satellite Galileo PRN E04 (Block FOC-FW13)
New Galileo E11 J/NAV message received to channel 4: Lono/GST model parameters from satellite Galileo PRN E05 (Block FOC-FW14)
New Galileo E11 J/NAV message received to channel 5: UTC model parameters from satellite Galileo PRN E06 (Block FOC-FW15)
New Galileo E11 J/NAV message received to channel 6: UTC model parameters from satellite Galileo PRN E07 (Block FOC-FW16)
New Galileo E11 J/NAV message received to channel 7: UTC model parameters from satellite Galileo PRN E08 (Block FOC-FW17)
New Galileo E11 J/NAV message received to channel 8: UTC model parameters from satellite Galileo PRN E09 (Block FOC-FW18)
New Galileo E11 J/NAV message received to channel 9: UTC model parameters from satellite Galileo PRN E10 (Block FOC-FW19)
New Galileo E11 J/NAV message received to channel 10: UTC model parameters from satellite Galileo PRN E11 (Block FOC-FW11)
New Galileo E11 J/NAV message received to channel 11: UTC model parameters from satellite Galileo PRN E12 (Block FOC-FW12)
New Galileo E11 J/NAV message received to channel 12: UTC model parameters from satellite Galileo PRN E13 (Block FOC-FW13)
New Galileo E11 J/NAV message received to channel 13: UTC model parameters from satellite Galileo PRN E14 (Block FOC-FW14)
New Galileo E11 J/NAV message received to channel 14: UTC model parameters from satellite Galileo PRN E15 (Block FOC-FW15)
New Galileo E11 J/NAV message received to channel 15: UTC model parameters from satellite Galileo PRN E16 (Block FOC-FW16)
New Galileo E11 J/NAV message received to channel 16: UTC model parameters from satellite Galileo PRN E17 (Block FOC-FW17)
New Galileo E11 J/NAV message received to channel 17: UTC model parameters from satellite Galileo PRN E18 (Block FOC-FW18)
New Galileo E11 J/NAV message received to channel 18: UTC model parameters from satellite Galileo PRN E19 (Block FOC-FW19)
New Galileo E11 J/NAV message received to channel 19: UTC model parameters from satellite Galileo PRN E20 (Block FOC-FW11)
New Galileo E11 J/NAV message received to channel 20: UTC model parameters from satellite Galileo PRN E21 (Block FOC-FW12)
New Galileo E11 J/NAV message received to channel 21: UTC model parameters from satellite Galileo PRN E22 (Block FOC-FW13)
New Galileo E11 J/NAV message received to channel 22: UTC model parameters from satellite Galileo PRN E23 (Block FOC-FW14)
New Galileo E11 J/NAV message received to channel 23: UTC model parameters from satellite Galileo PRN E24 (Block FOC-FW15)
New Galileo E11 J/NAV message received to channel 24: UTC model parameters from satellite Galileo PRN E25 (Block FOC-FW16)
New Galileo E11 J/NAV message received to channel 25: UTC model parameters from satellite Galileo PRN E26 (Block FOC-FW17)
New Galileo E11 J/NAV message received to channel 26: UTC model parameters from satellite Galileo PRN E27 (Block FOC-FW18)
New Galileo E11 J/NAV message received to channel 27: UTC model parameters from satellite Galileo PRN E28 (Block FOC-FW19)
New Galileo E11 J/NAV message received to channel 28: UTC model parameters from satellite Galileo PRN E29 (Block FOC-FW11)
New Galileo E11 J/NAV message received to channel 29: UTC model parameters from satellite Galileo PRN E30 (Block FOC-FW12)
New Galileo E11 J/NAV message received to channel 30: UTC model parameters from satellite Galileo PRN E31 (Block FOC-FW13)
New Galileo E11 J/NAV message received to channel 31: UTC model parameters from satellite Galileo PRN E32 (Block FOC-FW14)
New Galileo E11 J/NAV message received to channel 32: UTC model parameters from satellite Galileo PRN E33 (Block FOC-FW15)
New Galileo E11 J/NAV message received to channel 33: UTC model parameters from satellite Galileo PRN E34 (Block FOC-FW16)
New Galileo E11 J/NAV message received to channel 34: UTC model parameters from satellite Galileo PRN E35 (Block FOC-FW17)
New Galileo E11 J/NAV message received to channel 35: UTC model parameters from satellite Galileo PRN E36 (Block FOC-FW18)
New Galileo E11 J/NAV message received to channel 36: UTC model parameters from satellite Galileo PRN E37 (Block FOC-FW19)
New Galileo E11 J/NAV message received to channel 37: UTC model parameters from satellite Galileo PRN E38 (Block FOC-FW11)
New Galileo E11 J/NAV message received to channel 38: UTC model parameters from satellite Galileo PRN E39 (Block FOC-FW12)
New Galileo E11 J/NAV message received to channel 39: UTC model parameters from satellite Galileo PRN E40 (Block FOC-FW13)
New Galileo E11 J/NAV message received to channel 40: UTC model parameters from satellite Galileo PRN E41 (Block FOC-FW14)
New Galileo E11 J/NAV message received to channel 41: UTC model parameters from satellite Galileo PRN E42 (Block FOC-FW15)
New Galileo E11 J/NAV message received to channel 42: UTC model parameters from satellite Galileo PRN E43 (Block FOC-FW16)
New Galileo E11 J/NAV message received to channel 43: UTC model parameters from satellite Galileo PRN E44 (Block FOC-FW17)
New Galileo E11 J/NAV message received to channel 44: UTC model parameters from satellite Galileo PRN E45 (Block FOC-FW18)
New Galileo E11 J/NAV message received to channel 45: UTC model parameters from satellite Galileo PRN E46 (Block FOC-FW19)
New Galileo E11 J/NAV message received to channel 46: UTC model parameters from satellite Galileo PRN E47 (Block FOC-FW11)
New Galileo E11 J/NAV message received to channel 47: UTC model parameters from satellite Galileo PRN E48 (Block FOC-FW12)
New Galileo E11 J/NAV message received to channel 48: UTC model parameters from satellite Galileo PRN E49 (Block FOC-FW13)
New Galileo E11 J/NAV message received to channel 49: UTC model parameters from satellite Galileo PRN E50 (Block FOC-FW14)
New Galileo E11 J/NAV message received to channel 50: UTC model parameters from satellite Galileo PRN E51 (Block FOC-FW15)
New Galileo E11 J/NAV message received to channel 51: UTC model parameters from satellite Galileo PRN E52 (Block FOC-FW16)
New Galileo E11 J/NAV message received to channel 52: UTC model parameters from satellite Galileo PRN E53 (Block FOC-FW17)
New Galileo E11 J/NAV message received to channel 53: UTC model parameters from satellite Galileo PRN E54 (Block FOC-FW18)
New Galileo E11 J/NAV message received to channel 54: UTC model parameters from satellite Galileo PRN E55 (Block FOC-FW19)
New Galileo E11 J/NAV message received to channel 55: UTC model parameters from satellite Galileo PRN E56 (Block FOC-FW11)
New Galileo E11 J/NAV message received to channel 56: UTC model parameters from satellite Galileo PRN E57 (Block FOC-FW12)
New Galileo E11 J/NAV message received to channel 57: UTC model parameters from satellite Galileo PRN E58 (Block FOC-FW13)
New Galileo E11 J/NAV message received to channel 58: UTC model parameters from satellite Galileo PRN E59 (Block FOC-FW14)
New Galileo E11 J/NAV message received to channel 59: UTC model parameters from satellite Galileo PRN E60 (Block FOC-FW15)
New Galileo E11 J/NAV message received to channel 60: UTC model parameters from satellite Galileo PRN E61 (Block FOC-FW16)
New Galileo E11 J/NAV message received to channel 61: UTC model parameters from satellite Galileo PRN E62 (Block FOC-FW17)
New Galileo E11 J/NAV message received to channel 62: UTC model parameters from satellite Galileo PRN E63 (Block FOC-FW18)
New Galileo E11 J/NAV message received to channel 63: UTC model parameters from satellite Galileo PRN E64 (Block FOC-FW19)
New Galileo E11 J/NAV message received to channel 64: UTC model parameters from satellite Galileo PRN E65 (Block FOC-FW11)
New Galileo E11 J/NAV message received to channel 65: UTC model parameters from satellite Galileo PRN E66 (Block FOC-FW12)
New Galileo E11 J/NAV message received to channel 66: UTC model parameters from satellite Galileo PRN E67 (Block FOC-FW13)
New Galileo E11 J/NAV message received to channel 67: UTC model parameters from satellite Galileo PRN E68 (Block FOC-FW14)
New Galileo E11 J/NAV message received to channel 68: UTC model parameters from satellite Galileo PRN E69 (Block FOC-FW15)
New Galileo E11 J/NAV message received to channel 69: UTC model parameters from satellite Galileo PRN E70 (Block FOC-FW16)
New Galileo E11 J/NAV message received to channel 70: UTC model parameters from satellite Galileo PRN E71 (Block FOC-FW17)
New Galileo E11 J/NAV message received to channel 71: UTC model parameters from satellite Galileo PRN E72 (Block FOC-FW18)
New Galileo E11 J/NAV message received to channel 72: UTC model parameters from satellite Galileo PRN E73 (Block FOC-FW19)
New Galileo E11 J/NAV message received to channel 73: UTC model parameters from satellite Galileo PRN E74 (Block FOC-FW11)
New Galileo E11 J/NAV message received to channel 74: UTC model parameters from satellite Galileo PRN E75 (Block FOC-FW12)
New Galileo E11 J/NAV message received to channel 75: UTC model parameters from satellite Galileo PRN E76 (Block FOC-FW13)
New Galileo E11 J/NAV message received to channel 76: UTC model parameters from satellite Galileo PRN E77 (Block FOC-FW14)
New Galileo E11 J/NAV message received to channel 77: UTC model parameters from satellite Galileo PRN E78 (Block FOC-FW15)
New Galileo E11 J/NAV message received to channel 78: UTC model parameters from satellite Galileo PRN E79 (Block FOC-FW16)
New Galileo E11 J/NAV message received to channel 79: UTC model parameters from satellite Galileo PRN E80 (Block FOC-FW17)
New Galileo E11 J/NAV message received to channel 80: UTC model parameters from satellite Galileo PRN E81 (Block FOC-FW18)
New Galileo E11 J/NAV message received to channel 81: UTC model parameters from satellite Galileo PRN E82 (Block FOC-FW19)
New Galileo E11 J/NAV message received to channel 82: UTC model parameters from satellite Galileo PRN E83 (Block FOC-FW11)
New Galileo E11 J/NAV message received to channel 83: UTC model parameters from satellite Galileo PRN E84 (Block FOC-FW12)
New Galileo E11 J/NAV message received to channel 84: UTC model parameters from satellite Galileo PRN E85 (Block FOC-FW13)
New Galileo E11 J/NAV message received to channel 85: UTC model parameters from satellite Galileo PRN E86 (Block FOC-FW14)
New Galileo E11 J/NAV message received to channel 86: UTC model parameters from satellite Galileo PRN E87 (Block FOC-FW15)
New Galileo E11 J/NAV message received to channel 87: UTC model parameters from satellite Galileo PRN E88 (Block FOC-FW16)
New Galileo E11 J/NAV message received to channel 88: UTC model parameters from satellite Galileo PRN E89 (Block FOC-FW17)
New Galileo E11 J/NAV message received to channel 89: UTC model parameters from satellite Galileo PRN E90 (Block FOC-FW18)
New Galileo E11 J/NAV message received to channel 90: UTC model parameters from satellite Galileo PRN E91 (Block FOC-FW19)
New Galileo E11 J/NAV message received to channel 91: UTC model parameters from satellite Galileo PRN E92 (Block FOC-FW11)
New Galileo E11 J/NAV message received to channel 92: UTC model parameters from satellite Galileo PRN E93 (Block FOC-FW12)
New Galileo E11 J/NAV message received to channel 93: UTC model parameters from satellite Galileo PRN E94 (Block FOC-FW13)
New Galileo E11 J/NAV message received to channel 94: UTC model parameters from satellite Galileo PRN E95 (Block FOC-FW14)
New Galileo E11 J/NAV message received to channel 95: UTC model parameters from satellite Galileo PRN E96 (Block FOC-FW15)
New Galileo E11 J/NAV message received to channel 96: UTC model parameters from satellite Galileo PRN E97 (Block FOC-FW16)
New Galileo E11 J/NAV message received to channel 97: UTC model parameters from satellite Galileo PRN E98 (Block FOC-FW17)
New Galileo E11 J/NAV message received to channel 98: UTC model parameters from satellite Galileo PRN E99 (Block FOC-FW18)
New Galileo E11 J/NAV message received to channel 99: UTC model parameters from satellite Galileo PRN E100 (Block FOC-FW19)

```

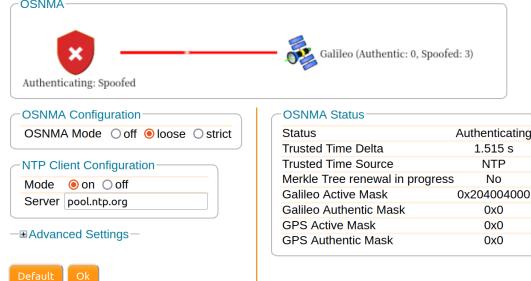
Fig. 17: Output of GNSS-SDR.



Fig. 18: Changes in carrier-to-noise ratio.



(a) Signals delayed by 1.4 seconds.



(b) Signals delayed by 1.5 seconds.

Fig. 19: CR attack on the victim receiver.

In order to study the cause behind the above phenomenon, we saved the Spetentrio binary format (SBF) files generated by the receiver during OSNMA processing in normal, jamming, and replay stages, and imported them into OSNMAlib for data parsing and analysis of authentication operations. The results show that when the delay is less than or equal to 1.4 seconds, the replayed subframes passed all OSNMA processes in the OSNMAlib successfully. However, when the delay exceeds 1.5 seconds, one-page misalignment (shift) occurred for the pages in the replayed subframes, as illustrated in Fig. 9. Therefore, we indicate that as long as the replay delay does not exceed 1.4 seconds, the CR attack can be successfully implemented.

To demonstrate the one-page misalignment, we show in Fig. 20 the 120-bit *HKROOT* (combination of the *HKROOT* of

all 15 pages) of two replayed subframes (one from satellite 34 and one from satellite 27) as examples. As shown in Fig. 20a, in general, normal *HKROOT* starts with 8 fixed bits (0x52 in Fig. 20) over a period of time, while the attack led to 8-bit (i.e., one *HKROOT* portion) shift, as shown in Fig. 20b. Since one page contains one *HKROOT* portion, this corresponds to the shift of one page. Consequently, the keys parsed from the replay subframes inevitably failed the verification, leading to the failure of tag verification.

```

INFO - KROOT with CID: 1 - PKID: 1 - GST0: WN 1290 TOW 460800
AUTHENTICATED

```

```

INFO - NMAS is TEST.
SVID: 34 HKROOT: 0x52418ec67093432268ec5294f078a0
SVID: 27 HKROOT: 0x5242b5366530affdf664b479c2d2f5
INFO - Chain in force is 1.
INFO - CPKS is NOMINAL.

```

(a) *HKROOT* of normal subframes.

```

SVID: 34 HKROOT: 0xa05242b5366530affdf664b479c2d2
SVID: 27 HKROOT: 0xf5243ddfcffa1b4a2ea871ca3e78
SVID: 27 HKROOT: False
INFO - --- SUBFRAME --- WN 1290 TOW 462510 SVID 34 ---
INFO - Received block 2 from DSM ID 5.
INFO - DSM ID 5 blocks: dict_keys([2])

```

(b) *HKROOT* of replayed subframes

Fig. 20: Illustration of page misalignment caused by CR attack.

VI. CONCLUSION

In this paper, we analyze the Galileo Open Service Navigation Message Authentication (OSNMA) mechanism and identified two vulnerabilities, namely artificially-manipulated time synchronization (ATS) and interruptible message authentication (IMA). Exploiting these vulnerabilities, we propose the real-time and non-real-time TS-comply replay (TSR) attacks, the TS-comply forgery (TSF) attack and the concatenating replay (CR) attack. We verify the feasibility of these attacks through extensive real-world experiments. The results showed when the difference between Galileo System Time (GST) and the LRT of the receiver is within 30 seconds, the TSR and TSF attacks will be successful. In addition, as long as the signal replay delay does not exceed 1.4 seconds, the CR attack can be successfully implemented.

REFERENCES

- [1] D. Schmidt, K. Radke, S. Camtepe, E. Foo, and M. Ren, "A survey and analysis of the gnss spoofing threat and countermeasures," *ACM Computing Surveys (CSUR)*, vol. 48, no. 4, pp. 1–31, 2016.
- [2] European Union, "GALILEO OPEN SERVICE NAVIGATION MESSAGE AUTHENTICATION (OSNMA) RECEIVER GUIDELINES, Issue 1.2," January 2024.
- [3] M. Götzelmann, E. Köller, I. Viciano-Semper, D. Oskam, E. Gkougkas, and J. Simon, "Galileo open service navigation message authentication: Preparation phase and drivers for future service provision," *NAVIGATION: Journal of the Institute of Navigation*, vol. 70, no. 3, 2023.
- [4] I. F. Hernández, T. Ashur, V. Rijmen, C. Sarto, S. Cancela, and D. Calle, "Toward an operational navigation message authentication service: Proposal and justification of additional osnma protocol features," in *2019 European Navigation Conference (ENC)*, pp. 1–6, IEEE, 2019.

- [5] A. Perrig, J. Tygar, A. Perrig, and J. Tygar, "Tesla broadcast authentication," *Secure Broadcast Communication: In Wired and Wireless Networks*, pp. 29–53, 2003.
- [6] I. Fernández-Hernández and G. Seco-Granados, "Galileo nma signal unpredictability and anti-replay protection," in *2016 International Conference on Localization and GNSS (ICL-GNSS)*, pp. 1–5, 2016.
- [7] K. Zhang, E. G. Larsson, and P. Papadimitratos, "Protecting gnss open service navigation message authentication against distance-decreasing attacks," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 58, no. 2, pp. 1224–1240, 2021.
- [8] M. Motallebihomi, H. Sathaye, M. Singh, and A. Ranganathan, "Location-independent gnss relay attacks: A lazy attacker's guide to bypassing navigation message authentication," *ACM WiSec 2023*, 2023.
- [9] J. Junquera-Sánchez, C. Hernando-Ramiro, O. Gamallo-Palomares, and J.-A. Gómez-Sánchez, "Assessment of cryptographic approaches for quantum-resistant galileo osnma," *NAVIGATION: Journal of the Institute of Navigation*, vol. 71, no. 2, 2024.
- [10] H. Wang, Y. Zhang, Y. Shen, J. Zhu, Y. Chen, and X. Jiang, "Novel replay attacks against galileo open service navigation message authentication," in *Proceedings of the 36th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2023)*, pp. 3897–3907, 2023.
- [11] H. Sathaye, M. Motallebihomi, and A. Ranganathan, "Galileo-sdrsim: An open-source tool for generating galileo satellite signals," in *Proceedings of the 36th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2023)*, pp. 3470–3480, 2023.
- [12] C. Fernandez-Prades, J. Arribas, P. Closas, C. Aviles, and L. Esteve, "Gnss-sdr: An open source tool for researchers and developers," in *Proceedings of the 24th International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS 2011)*, pp. 780–794, 2011.
- [13] A. Galan, I. Fernandez-Hernandez, L. Cucchi, and G. Seco-Granados, "Osmalib: An open python library for galileo osnma," in *2022 10th Workshop on Satellite Navigation Technology (NAVITEC)*, pp. 1–12, 2022.
- [14] [Online]. Available: <https://github.com/Haiyang-Wong/Galileo-OSNMA-Generator>. Accessed Oct., 2025.
- [15] "osqzss/gps-sdr-sim: Software-defined gps signal simulator." [Online]. Available: <https://github.com/osqzss/gps-sdr-sim>, 2023. Accessed Dec., 2023.
- [16] K. Wang, S. Chen, and A. Pan, "Time and position spoofing with open source projects," *black hat Europe*, vol. 148, pp. 1–8, 2015.
- [17] E. Horton and P. Ranganathan, "Development of a gps spoofing apparatus to attack a dji matrice 100 quadcopter," *The Journal of Global Positioning Systems*, vol. 16, no. 1, pp. 1–11, 2018.
- [18] T. E. Humphreys, B. M. Ledvina, M. L. Psiaki, B. W. O'Hanlon, P. M. Kintner, et al., "Assessing the spoofing threat: Development of a portable gps civilian spoofer," in *Proceedings of the 21st International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS 2008)*, pp. 2314–2325, 2008.
- [19] T. Nighswander, B. Ledvina, J. Diamond, R. Brumley, and D. Brumley, "Gps software attacks," in *Proceedings of the 2012 ACM conference on Computer and communications security*, pp. 450–461, 2012.
- [20] M. Lenhart, M. Spanghero, and P. Papadimitratos, "Relay/replay attacks on gnss signals," in *Proceedings of the 14th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, pp. 380–382, 2021.
- [21] M. Lenhart, M. Spanghero, and P. Papadimitratos, "Distributed and mobile message level relaying/replaying of gnss signals," in *Proceedings of the 2022 International Technical Meeting of The Institute of Navigation*, pp. 56–67, 2022.
- [22] S. Shang, H. Li, Y. Wei, and M. Lu, "A flexible replay delay control method for gnss direct meaconing signal," in *Proceedings of the 2020 International Technical Meeting of The Institute of Navigation*, pp. 992–1000, 2020.
- [23] European Union, "GALILEO OPEN SERVICE NAVIGATION MESSAGE AUTHENTICATION (OSNMA) SIGNAL-IN-SPACE INTERFACE CONTROL DOCUMENT, Issue 1.1," October 2023.
- [24] European Union, "GALILEO OPEN SERVICE NAVIGATION MESSAGE AUTHENTICATION (OSNMA) USER ICD FOR THE TEST PHASE, Issue 1.0," November 2021.
- [25] I. Fernandez-Hernandez, T. Walter, A. Neish, and C. O'Driscoll, "Independent time synchronization for resilient gnss receivers," in *Proceedings of the 2020 International Technical Meeting of The Institute of Navigation*, pp. 964–978, 2020.
- [26] M. Conti, N. Dragoni, and V. Lesyk, "A survey of man in the middle attacks," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 3, pp. 2027–2051, 2016.
- [27] L. Rudman and B. Irwin, "Characterization and analysis of ntp amplification based ddos attacks," in *2015 Information Security for South Africa (ISSA)*, pp. 1–5, IEEE, 2015.
- [28] A. Malhotra and S. Goldberg, "Attacking ntp's authenticated broadcast mode," *ACM SIGCOMM Computer Communication Review*, vol. 46, no. 2, pp. 12–17, 2016.
- [29] A. Galan, C. Ifiñez, I. Fernandez-Hernandez, S. Pollin, and G. Seco-Granados, "Osmalib improvements and real-time monitoring of galileo osnma," in *2024 International Conference on Localization and GNSS (ICL-GNSS)*, pp. 1–7, IEEE, 2024.
- [30] D. L. Mills, *Computer network time synchronization: the network time protocol on earth and in space*. CRC press, 2017.
- [31] D. Franke, D. Sibold, K. Teichel, M. Dansarie, and R. Sundblad, "Network time security for the network time protocol," *RFC 8915*, 2020.
- [32] E. D. Kaplan and C. Hegarty, *Understanding GPS/GNSS: principles and applications*. Artech house, 2017.
- [33] K. Borre, *A software-defined GPS and Galileo receiver: a single-frequency approach*. Springer Science & Business Media, 2007.
- [34] [Online]. Available: <https://www.gnuradio.org/>. Accessed Dec., 2023.
- [35] [Online]. Available: <https://github.com/jselvi/Delorean>. Accessed Mar., 2024.
- [36] [Online]. Available: <https://www.gsc-europa.eu/electronic-library/programme-reference-documents>. Accessed Dec., 2023.

APPENDIX

This appendix provides an example of implementing the TSF attack, including forging navigation data forgery, tags and subframes. This example is based on the subframes of the satellite with PRN 02 extracted from the test vectors (16_AUG_2023_GST_05_00_01.csv) provided by EUSPA.

A. Forging navigation data

We select the first subframe (WN = 1251, TOW = 277200) as an example and modify its ionospheric parameters, which are located on the 13-th page of the subframe. Below are the original and modified bits of the 13-th page:

0x054BC11429A07F9FC009C6875D2A80AAAAB21D
69F9A18E29635CF8EC0100 (original)
0x054BC11429A17F9FC009C6875D2A80AAAAB21D
69F9A18E29635CF8EC0100 (modified)

B. Forging tag

The ionospheric parameter is located in the first navigation segment, which can be retrieved from the subframe (TOW = 277200) as follows:

0x1311F898EE1868001F06E7AA04D76D1333662A42
49DD4A6EBB4CAE193D2A133FF06889EB3F5F823B
87F37F405AC4C0BFFFBBFFB11F8001D4E9A0009901
2F0450A685FE7F000 (552 bits)

We generate a message as follows by concatenating different fields according to Eq. (3):

0x024E343AEE0144C47E263B861A0007C1B9EA813
5DB44CCD98A909277529BAED32B864F4A84CFFC1
A227ACFD7E08EE1FCDFD016B1302FFEFFFEC4
7E000753A680026404BC11429A17F9FC00 (600 bits)

The key can be retrieved from the MACK field in the subframe with TOW = 277260 as follow:

0xACA75FBC1C6E40A397CA7EE7EE908870 (128 bits)

The MAC function indicated by the *HKROOT* field is HMAC-SHA-256. Taking the message and the key as input, the output from the MAC function based on Eq. (4) is as follows.

*0x3C2585C882811FD8B740A5C04CE82C1FC8CA4F
722A018A5B32C031F9025F749C* (256 bits)

This output is finally truncated to a 40-bit tag length indicated by the TS field, 40 bits, which is as follows:

0x3C2585C882 (40 bits)

The generated tag is the first tag in the subframe with TOW = 277230. Specifically, the 40-bit tag is divided into two parts, 32 bits and 8 bits. The 32-bit part is filled into the *MACK* field of the first OSNMA portion, and the 8-bit part is filled into the *MACK* field of the second OSNMA portion. The tag field in the first page of the subframe (TOW = 277230) is replaced by the 32-bit part as follows:

*0x021333662A4249DD4A6EBB4CAE1900BD2A5C8F
0961722AAAAA73F18B0100* (replaced tag)

C. Forging subframe

The CRC value for the modified page data needs to be re-calculated and replaced to the corresponding field. Therefore, the page associated with the ionospheric parameter in the first subframe (TOW = 2772300) needs CRC calculation, and the page associated with the first tag in the second subframe (TOW = 2772330) also needs CRC calculation. Below are the data of the first page in the subframe (TOW = 2772300) after replacing tag and CRC:

*0x021333662A4249DD4A6EBB4CAE1900BD2A5C9E
8497BA6AAAAA6A9778C100* (replaced CRC)