Analysis and Recommendations for MAC and Key Lengths in Delayed Disclosure GNSS Authentication Protocols

IGNACIO FERNÁNDEZ-HERNÁNDEZ
European Commission, Bruxelles, Belgium

TOMER ASHUR
imec-COSIC, KU Leuven, Leuven, Belgium
TU Eindhoven, Eindhoven, The Netherlands

VINCENT RIJMEN
imec-COSIC, KU Leuven, Leuven, Belgium

Data and signal authentication schemes are being proposed to address Global Navigation Satellite Systems' (GNSS) vulnerability to spoofing. Due to the low power of their signals, the bandwidth available for authentication in GNSS is scarce. Since delayed-disclosure protocols, e.g., TESLA (timed-efficient stream loss-tolerant authentication), are efficient in terms of bandwidth and robust to signal impairments, they have been proposed and implemented by GNSS. The length of message authentication codes (MACs) and cryptographic keys are two crucial aspects of the protocol design as they have an impact on the utilized bandwidth, and therefore on the protocol performance. We analyze both aspects in detail for GNSS-TESLA and present recommendations for efficient yet safe MAC and key lengths. We further complement this analysis by proposing possible authentication success and failure policies and quantify the reduction of the attack surface resulting from employing them. The analysis shows that in some cases it is safe to use MAC and key sizes that are smaller than those proposed in best-practice guidelines. While some of our considerations are general to delayed-disclosure lightweight protocols for data and signal authentication, we particularize them for GNSS-TESLA protocols.

Manuscript received May 26, 2020; revised October 14, 2020; released for publication December 21, 2020. Date of publication January 26, 2021; date of current version June 9, 2021.

DOI. No. 10.1109/TAES.2021.3053129

Refereeing of this contribution was handled by J. Blanch.

The work of Tomer Ashur was supported by an FWO post-doctoral fellowship under Grant Number 12ZH420N.

Authors' addresses: Ignacio Fernández-Hernández is with European Commission, Bruxelles 1040, Belgium, E-mail: (Ignacio.fernandez-hernandez@ec.europa.eu); Tomer Ashur is with TU Eindhvoen, Eindhoven 5612, The Netherlands, E-mail: (t.ashur@tue.nl); Vincent Rijmen is with KU Leuven, Leuven 3000, Belgium, E-mail: (vincent.rijmen@kuleuven.be). *(Corresponding author: Ignacio Fernández-Hernández.*)

I. INTRODUCTION

The threat of spoofing GNSS (Global Navigation Satellite Systems) is more than a decade old, and cases of GNSS spoofing have been observed with increasing frequency. Possible spoofing countermeasures implemented on the receiver's end include signal power verification [1], [2], consistency checks, and/or combination of GNSS with other navigation sources (e.g., inertial navigation systems [3]). An alternative to this approach is to add GNSS authentication already at the source by adding cryptographic features to GNSS signals. Galileo, the European GNSS, is the first such system to include cryptographic authentication [4], and other systems have already shared studies or plans to follow suit [5], [6].

However, adding cryptographic authentication to GNSS is not a trivial task as such systems are subject to technical limitations. They are one-way broadcast systems, thus they cannot configure their signal to specific users, which makes it hard for the receiver to determine that the signal was received from an authoritative source. Further compounding the problem is that their signals are received at a very weak power of around -155 to -160 dBW on Earth, which restricts the available data bandwidth, as receivers must accumulate enough energy to decode each bit. For example, the popular GPS L1 C/A transmits every bit over a period of 20 ms, resulting in a throughput of 50-bit/s; for Galileo E1 these numbers are respectively 4 ms and 250-symbols/s, encoding bits with a ½ coding rate.

The academic literature contains a plenitude of proposals for radionavigation authentication protocols [7]–[12]. The standard approach is to adopt and adapt cryptographic protocols from other domains, e.g., network communications. However, this process may require specific tailoring before it can efficiently meet the special constraints of GNSS.

This article focuses on delayed-disclosure protocols. These are protocols based on the dissemination of an authentication tag [sometimes also called "code"; or more specifically MAC (Message Authentication Code) when the authentication tag is the output of a MAC function], stored by the receiver, and authenticated at a later time following a disclosure of a certain cryptographic datum such as a yet-unknown symmetric-key. The motivation for this is twofold. First, such protocols can be optimized for bandwidth, motivating numerous proposals such as Galileo Open Service Navigation Message Authentication (OSNMA) [13]; GPS's experimental signal authentication scheme, CHIMERA [5], and SBAS authentication [14]–[16]. Second, while the delayed disclosure requirement is only one option for data authentication (the other option being digital signatures), it may be a necessary component for spreading code authentication [17]. This makes our analysis applicable to code-level authentication schemes for GNSS in which the authentication code watermarks the satellite spreading code. Among delayed-disclosure protocols, we focus on lightweight protocols, and in particular the TESLA (timed-efficient stream loss-tolerant authentication) protocol, standardized in [18].

After this introduction, we provide a short overview of TESLA adapted for GNSS data authentication, hereinafter referred to as GNSS-TESLA. We then propose a framework for analyzing the probability of MAC forgery attacks. Two user profiles are considered: safety-critical and standard; and we derive the probability of a successful forgery attack against each of them respective to our proposals for successful and failed authentication policies, and respective to different MAC sizes. We then particularize the discussion to delayed-disclosure schemes using hash chains for key dissemination. This key dissemination mechanism is the engine underlying the TESLA protocol. We describe the adversarial model and possible attack vectors and derive attack complexities, which we use to estimate attack costs and recommend safe key sizes and cryptoperiods. We conclude the article with a discussion on how to interpret our recommendations.

II. PRELIMINARIES

A. Overview of TESLA Delayed-Disclosure Authentication Protocols for GNSS

The TESLA scheme is a delayed-disclosure authentication protocol for continuous data broadcasting presented in [19]. Several references detail TESLA protocol implementations applied to GNSS, such as [20] and [21]. The idea behind this protocol is to use a one-way function (e.g., a hash function) to create a chain of authentication keys; this is called a *keychain*. The last key of this chain is then released and authenticated off-channel. The last bootstrapping step, i.e., how to authenticate the last key in the chain, is outside the scope of this article. In each step of the protocol, a unit of data is authenticated by means of a cryptographic MAC function using a previously disclosed key in the chain. The key is disclosed after a delay period and its validity is verified by locally re-evaluating the one-way function with respect to this key and comparing its output to the last known valid key. Once the key is accepted as valid, previously received data authenticated using this key, can be verified.

The receiver stores the received data stream *D* and its corresponding MAC *M*, and stores them awaiting the arrival of the delayed-disclosed key *K*. Upon the arrival of this key, the receiver computes *M'* using *D* and *K*, and compares this with the broadcast *M*. More formally

$$\begin{aligned} M' &= mac(K, D) \\ M = M' &\rightarrow \text{Authentication passed} \\ M \neq M' &\rightarrow \text{Authentication failed} \end{aligned} \tag{1}$$

where *mac* is a MAC function taking key and data as first and second inputs, respectively, and retuning their MAC. Implicit in this definition is that the output of the MAC function is *S*-bit long. This is easily achieved by directly employing a MAC function outputting MACs of the appropriate length, or by truncating a longer output. The cryptographic security of the authentication process therefore depends on the choice of the MAC function, the MAC length, and the use of a suitable key.

B. Primitives and Generic Security

MAC functions are a family of functions, parameterized by a symmetric key, such that each function in the family performs a unique mapping from an arbitrary-length input to a fixed-length output, of *S* bits in our case. In the sequel, we refer to this output as *the MAC*, and to its generating function as *the MAC function.* An attack against the MAC function (resp., the MAC) is any method to distinguish it from a random mapping (resp., from a truly *S*-bit random bitstring). MAC functions are a well-understood object and some designs have been shown to withstand all forms of known cryptanalysis despite many years of scrutiny. In symmetric-key cryptography this is the highest form of confidence as symmetric-key algorithms are normally not reducible to hard mathematical problems as in the case of public-key cryptography. Specifically, HMAC-SHA-256 and CMAC-AES-256, proposed for Galileo OSNMA [22], have been determined to satisfy the confidence threshold required by standardization bodies and were subsequently included in their portfolios [23]–[25].

The same discussion, *mutatis mutandis*, applies to hash functions, which can be viewed as a degenerate MAC function whose key was fixed to a publicly known value.[1] A more detailed framework for the security objectives and basic attacks of one-way functions, including keyed- and unkeyed-hash functions, can be found in [26].

We note that GNSS does not introduce any particularities in that respect and proceed under the assumption that choosing a secure primitive (MAC or hash function) is a closed problem: a function included in the portfolio of a respectable standardization body would suffice, security-wise, and efficiency can now be the primary driver for the actual choice within the portfolio. We can therefore proceed to analyze the two remaining factors: the MAC function's output size and the key length.

C. Adversarial Model and Attack Objectives

In the context of GNSS data authentication, the goal of an adversary is to trick the user into accepting spoofed data, i.e., data that was not originated from an authoritative source. This can be done by attacking the output directly or by means of a key recovery; each of these is said to be an *attack vector*. The output and the key, both being of finite length, can be attacked generically irrespective to their generating function.

The *attack surface* of an attack vector is the number of targets within this attack vector. For example, if instead of providing spoofed data respective to a specific MAC it is sufficient for an adversary to provide spoofed data respective to either of two MACs we say that the attack surface is doubled; conversely, if the spoofed data must

[1] Hash functions often have this abstracted by eliminating the key interface altogether or by defining a different hierarchy of primitives (i.e., instantiating a MAC function from a hash function, e.g., HMAC). Nevertheless, these are conceptual abstractions made for convenience and they are in no way more "correct" than the one we offer.

correspond to both MACs, the attack surface is reduced by a factor 2.

The cost of an attack is captured by the *attack complexity*, usually the number of calls to the generating function. For example, given an S-bit MAC, providing a matching input consistent with the key that was used to generate this MAC has a generic attack complexity of 2^S. Similarly, finding a preimage for an n-bit output of a hash function has generic attack complexity of 2^n. Note that due to the nature of such random mappings their analyses are probabilistic and the attack complexities are *expected* values of random variables.

We analyze in the sequel the attack surface of the two dominant cryptographic attack vectors for GNSS-TESLA. In the next section we consider MAC forgery. Then, in the subsequent section we analyze the attack surface of TESLA-based key recovery.

III. MAC FORGERY

This section analyzes the attack surface of MAC forgery attacks. These are attacks supplementing the user with spoofed data matching an existing MAC independently of the key (and the authentication/signature mechanism). Two approaches to MAC forgeries are possible: deterministic and probabilistic. In the deterministic approach, the adversary first recovers the secret key, and subsequently continues to authenticate spoofed data with a success probability of 1 (thus "deterministic"). It is therefore covered in the next section on finding a key preimage. In the probabilistic approach, the adversary attacks the random mapping directly by "guessing" an input (resp., an input–output pair) and hope it matches (resp., they match) the right output (resp., each other). The generic attack complexity of the probabilistic approach is a function of the MAC length. We assume that the receiver is loosely synchronized with the transmitter (the GNSS) and therefore the adversary cannot delay the signal enough to receive the key and forge the MAC [27]. We also assume that the MAC function is secure and only generic attacks are possible. Observing that in GNSS-TESLA the MAC length is generally smaller than the key size, it is not rewarding for the adversary to prefer the deterministic approach when executing a nonpersistent attack. Therefore, in this section, we analyze the success probability of an adversary taking the probabilistic approach to attack the MAC function.

A. Related Work

The U.S. National Institute of Standards and Technology (NIST) recommends that MAC lengths be no smaller than 32 bits, but states explicitly that a low-bandwidth channel might still use 32-bit MACs [28]. The Joint Technical Committee of the International Organisation for Standardization and the International Electrotechnical Commission (ISO/IEC JTC 1) recommends in [29] that the tag length shall be a multiple of 8 bits, but makes no statement on minimum tag length. Neither do they in [18] for the specific case of broadcast authentication. ENISA's report on Algorithms, Key Sizes, and Parameters [30] discusses the subject briefly noting that the "*choice of the MAC output size can be very much scheme, protocol, or even system, dependent*" without making a recommendation on the output size. The 2020 version of this report (to be published) will ratify this statement.

Within the GNSS domain, current references range between 15 and 40 bits. Some early references considered 15 bits as sufficiently low to discourage attacks [31]. Others suggest that 24 bits or more are secure [32]. For GNSS integrity systems such as SBAS, a minimum of 29 bits is proposed in [16]. 30 bits are proposed in [15] although MACs as short as 10 are also discussed. Galileo OSNMA currently published test specification mentions the MAC size as a configurable parameter with a value ranging between 10 and 40 bits [22]. Some of these references mention that the MAC length should be defined according to the random guessing probability of the attacker, but they do not define user authentication logic and operation period, which we now consider in detail.

B. MAC Forgery Attack Model

A successful forgery is a pair (D', M') such that

$$M' = mac\left(K, D'\right) \tag{2}$$

where D' is the spoofed data; and M' is an S-bit MAC successfully authenticating D' with respect to a valid key K, unknown to the attacker. Observing that the constraint that K is a valid key frustrates the birthday paradox on (D, M'), it is sufficient for our needs, without loss of generality, to restrict the discussion to a spoofed D' respective to a fixed M. Committing to a spoofed value (or respectively, to a pair (D', M')) is called a *forgery attempt* and it is the attack vector we consider. The probability that a single forgery attempt is successful is $P = 2^{-S}$, and the attack complexity is $\frac{1}{P} = 2^S$.

C. MAC Forgery Success Probability for GNSS-TESLA

In GNSS-TESLA, we define the attack surface as all forgery attempts within a predetermined time frame. The probability that the attack surface is penetrated, i.e., that at least one forgery attempt succeeds, is given by

$$P_{N_T} = 1 - \left(1 - 2^{-s}\right)^{N_T} \tag{3}$$

where N_T is the number of forgery attempts (the attack surface). A good approximation to this probability is $N_T \cdot 2^{-S}$ as long as $N_T << 2^S$ (see also [28]).

In the original TESLA scheme, every step of the protocol involves one dataset, one MAC, and one delayed key. In GNSS-TESLA, we generalize this definition: we define a *block* as the information unit that contains *at least* one satellite, and one key and one MAC per satellite, but noting that a block will usually include data from several satellites; a satellite may transmit more than one dataset, and there may be several MACs per key transmission; and even multiple key transmissions. Since every block may contain several MACs and datasets, verification attempts are tried

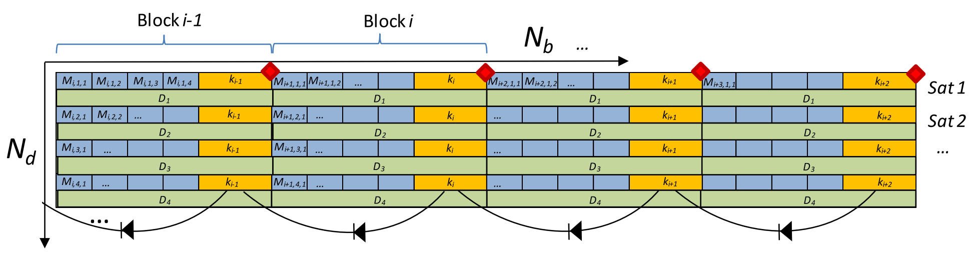


Fig. 1. GNSS-TESLA data structure for a MAC and key block from multiple satellites. Each satellite transmits its data (D), several MACs (M) and the key (k). $\boldsymbol{N_d}$ is the number of datasets authenticated in each block and $\boldsymbol{N_b}$ the number of blocks within the attack period. Each diamond represents that the key has been fully received and verifications of MACs from the preceding block can commence.

for multiple pairs of (D, M), which allows multiple forgery attempts. This is depicted in Fig. 1, where a block contains a single TESLA key, e.g., $k_{\text{i-1}}$; multiple datasets D_j transmitted from the different satellites, e.g., D_1 is transmitted from the first satellite; and multiple MACs transmitted from each satellite, e.g., $M_{i,1,1}$, is the first MAC from satellite 1; $\mathrm{M}_{i,1,2}$, is the second MAC from satellite 1, etc. The MACs are authenticated using the key included in the next block, e.g., k_i in our example. The term *dataset* is used here in place of *message* in order to avoid confusion with the GNSS navigation message. The latter is understood as a several-minute data stream that may contain several datasets, including e.g., satellite almanacs, time conversion parameters, or satellite orbits and clocks. For more details we refer the reader to GNSS Signal-In-Space Interface Control Documents, or SIS ICDs [33], [34]. We require that at least one of the MACs transmitted by each satellite self-authenticates its own satellite's dataset to ensure that each dataset can be authenticated, and make no further assumptions about the other MACs which may cross-authenticate other satellites or other datasets.

The attack surface $N_T = N_d \cdot N_b$ depends on both the number of datasets ($D_1, D_2 \ldots$) authenticated per block (N_d) and the number of blocks (N_b) received within the attack period.

We define the probability of finding a successful forgery within a predefined timeframe τ as $P_{f,\tau}$, and bound it by $P_{\max,\tau}$:

$$P_{f,\tau} = \frac{N_d \cdot N_b}{2^S} < P_{\max,\tau} . \tag{4}$$

In (4), S is the MAC length, N_d is the number of different datasets authenticated per block, and N_b is the number of blocks transmitted during the attack period τ. We still assume that the number of forgery attempts is sufficiently low to satisfy $N_d N_b \ll 2^S$. Specifically, we consider $\tau = 1$ h and motivate this decision later. In (4) we also consider $P_{\max,\tau}$ as the threshold above which the system is no longer considered secure. This value should be supplied by systems' designers according to their own risk analysis and compared to $P_{f,\tau}$ which we now proceed to estimate.

D. Definition of $\boldsymbol{N_d}$ and Assumptions

The number of forgery attempts per block depends on the number of datasets that are authenticated in the block, and the policy of accepting successful authentications in effect on the receiver.

Case Nd-A—Lenient authentication policy: Under this policy the receiver treats every dataset independently from the others and accepts a dataset if it is successfully authenticated by its corresponding MAC, discarding it otherwise. Assuming N_d (possibly spoofed) datasets are included in a block, the probability that *at least one* forgery attempt succeeds, is well approximated by $P_f = N_d \cdot 2^{-S}$. We stress that if several MACs authenticate a dataset, the receiver considers the dataset authenticated only if *all* of the received ones are successfully verified. Thus, having more than one MAC per dataset reduces the attack surface. Regarding the number of datasets, one could assume four satellites in view, which is the reference for GNSS constellation design, and therefore four datasets, but in usual geometries there are more satellites in view and we must consider at least one additional dataset not related to a specific satellite, for example, for UTC-GNSS time conversion [33]. We will assume for the lenient authentication policy $N_d = 8$.

Case Nd-B—Strict authentication policy: Under this policy the receiver accepts a block as authenticated only if all datasets in the block are successfully verified. If at least one dataset fails, the entire block is discarded and all authentication verifications in this block are considered as failed. Assuming j datasets, since all must now authenticate, the probability for a successful forgery decreases exponentially to

$$P_{f,\text{all}} = \frac{1}{2^{S \cdot j}} \tag{5}$$

where $P_{f,\text{all}}$ is the probability that the adversary successfully spoofs j datasets. Note that under this policy the adversary must avoid detection in addition to successfully spoof a dataset. The most rewarding strategy in this case is to leave $j - 1$ datasets untouched and spoof just one dataset, giving a success probability of $P_f = 2^{-S}$. This corresponds to setting $N_d = 1$ for the strict authentication policy.

E. Definition of N_b and Assumptions

The above N_d quantified the number of forgery attempts that can be made within a single block. The number of blocks that can be attacked within τ, namely N_b, depends on certain assumptions. First of all, the duration of a block needs to be determined. According to our block definition, at least one dataset needs to be transmitted in a block. Therefore, the block duration must be the longest of the authentication and dataset transmission intervals. For example, if three keys and one dataset (e.g., a set of ephemeris) are transmitted per a 30-s subframe, the block duration is 30 s. Even if there are multiple trials from various MAC-key combinations, since all authentications must pass, the attack surface is reduced, as mentioned before. Therefore, N_b is given by

$$N_b = \frac{\tau}{\max\left(\tau_{\text{auth}}, \tau_{\text{data}}\right)} \tag{6}$$

where τ_{auth} is the time between authentications (also known as TBA in other references [9], [31]), and τ_{data} is the dataset transmission period, e.g., 30 s for Galileo I/NAV or GPS L1C/A ephemeris datasets. Note that we assume that the adversary has no time advantage over the victim and therefore both learn if a verification succeeded or failed concurrently, following the key disclosure. This assumption prevents the adversary to react to a failed verification by preventing the victim to receive that key and begin a verification known to fail. The situation can be incorporated into the attack surface but this would unnecessarily complicate the model: First, it would require a continuous service denial by the adversary for potentially long durations, which can be detected by the receiver through other means. Second, this might only increase N_b, so it is anyway bounded by case Nb-A below. Therefore, we do not consider it.

Case Nb-A—No treatment of failed authentications: Under this policy, a failed authentication has no impact on future verification attempts. For example, applying (6) to Galileo OSNMA, $N_b = 3600$ s / max(30 s, 30 s) = 3600 s / 30 s = 120, where 30 s is the maximum τ_{auth} for OSNMA, and τ_{data} is also 30 s (see [33]). To simplify the analysis we set $N_b = 2^7 = 128 > 120$.[2] Other GNSS have similar or lower N_b values making it representative for GNSS in general.

Case Nb-B—Treatment of failed authentications—satellite exclusion: Under this policy, the attack surface is reduced by excluding the satellite from which the offending dataset originated for a certain time period. For example, if, following a failed authentication, the originating satellite is excluded from subsequent blocks for the rest of the τ period, each satellite can be attacked at most once during this period. The consequence is that, by disabling the satellites for the rest of τ, N_b would be reduced to the number of authenticated satellites. Thus, we assume in our analysis $N_b = 16$, i.e., 16 trials from a maximum of 16 different satellites authenticated in an hour.

Case Nb-C—Treatment of failed authentications—service exclusion or increased vigilance: Under this treatment policy, the authentication function is discontinued after a failed authentication for the rest of τ. If this is too aggressive, the receiver may move to applying a stricter verification policy. For example, it may require that two MACs authenticate each dataset, effectively doubling the number of authenticating bits (see Fig. 2) or apply other measures. The actual measure used for this policy is a matter of dedicated risk analysis, and we abstract it here it by assuming that the measure ensures that a successful forgery following a failed forgery attempt is impossible.

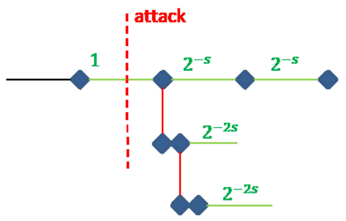


Fig. 2. Request of an additional MAC after failed authentication. The attack starts after the first verification. The verification success is represented by the green line, and the failure by the red line. The success probability for each verification is shown in green.

The new measures can be implemented for just the originating satellite, in which case $N_b = 16$, as justified for Nb-B, or to all visible satellites, in which case $N_b = 1$. We will proceed assuming the latter.

F. User Profiles and Security Levels

In order to determine a safe MAC length S, we consider two types of users: safety-critical and standard, each associated with different risk profiles; and define a different tolerance level $P_{\max,\tau}$ for each.

Safety-critical user: Currently, the most stringent GNSS requirements come from safety-critical aviation operations. These include an integrity requirement of maximum failure probability of 10^{-7} per 150 s for precision approach (the most stringent phase flight), and 10^{-7} per hour for the other navigation phases after take-off (*en-route*, terminal approach, non-precision approach) (see [35]). As we cannot allocate the whole failure probability to a MAC forgery attack, we conservatively assume that 1% of the

[2]Note that any system shown to be secure when $N_b = 128$ remains secure with the more restricted $N_b = 120$, but not any system that is insecure when $N_b = 128$ is also insecure when $N_b = 120$. Effectively, we will now proceed to demonstrate the security of a weaker system than the one we are interested in.

TABLE I
Summary of Possible Parameters

N_d	8 (lenient policy)	1 (strict policy)
N_b	$N_T = N_b N_d$; Number of trials	
128 (no treatment)	1024 (do nothing)	128 (discard full block after a failed authentication)
16 (satellite exclusion)	128 (satellite exclusion during τ)	16 (discard block & satellite exclusion during τ)
1 (service exclusion, or increased vigilance)	8 (service exclusion during τ)	1 (discard full block if one failed authentication & service exclusion during τ)

TABLE II
Possible Values for N_b, N_d, and Resulting Attack Surfaces

Parameter	Value	Justification
$P_{max,\tau}$	10^{-9}	Safety-Critical User. Acceptable impact on aviation safety (International Civil Aviation Organisation, 2016).
	10^{-5}	Standard User. Based on current satellite failure probabilities (U.S. DoD, 2008).
τ	1h	Based on aviation integrity requirement (International Civil Aviation Organisation, 2016).
N_d	8	Case Nd-A: Lenient authentication policy. At most 8 datasets per block.
	1	Case Nd-B: Strict authentication policy. If any of the datasets in the block failed to authenticate, the entire block is discarded and all datasets are considered to have failed their authentication.
N_b	128	Case Nb-A: No treatment of failed authentications. Value for Galileo INAV/GPS L1C/A (120 blocks in one hour)
	16	Case Nb-B: Treatment of failed authentications – originating satellite exclusion until the end of the attack period τ. At most 16 satellites in view during the attack period τ.
	1	Case Nb-C: Treatment of failed authentications – service exclusion or additional MAC required until the end of the attack period τ, for long-enough MACs.

TABLE III
Success Probability $P_{f,\tau}$ of a MAC Forgery Attack With $\tau = 1$ h, for Different MAC Lengths and Attack Surfaces

MAC length	Attack surface				
	1024	128	16	8	1
16	1.56E-02	1.95E-03	2.44E-04	1.22E-04	1.53E-05
17	7.81E-03	9.77E-04	1.22E-04	6.10E-05	7.63E-06
18	3.91E-03	4.88E-04	6.10E-05	3.05E-05	3.81E-06
19	1.95E-03	2.44E-04	3.05E-05	1.53E-05	1.91E-06
20	9.77E-04	1.22E-04	1.53E-05	7.63E-06	9.54E-07
22	2.44E-04	3.05E-05	3.81E-06	1.91E-06	2.38E-07
24	6.10E-05	7.63E-06	9.54E-07	4.77E-07	5.96E-08
26	1.53E-05	1.91E-06	2.38E-07	1.19E-07	1.49E-08
28	3.81E-06	4.77E-07	5.96E-08	2.98E-08	3.73E-09
30	9.54E-07	1.19E-07	1.49E-08	7.45E-09	9.31E-10
32	2.38E-07	2.98E-08	3.73E-09	1.86E-09	2.33E-10
34	5.96E-08	7.45E-09	9.31E-10	4.66E-10	5.82E-11
36	1.49E-08	1.86E-09	2.33E-10	1.16E-10	1.46E-11
38	3.73E-09	4.66E-10	5.82E-11	2.91E-11	3.64E-12
40	9.31E-10	1.16E-10	1.46E-11	7.28E-12	9.09E-13
60	8.88E-16	1.11E-16	1.39E-17	6.94E-18	8.67E-19
80	8.47E-22	1.06E-22	1.32E-23	6.62E-24	8.27E-25

Note: When the success probability is lower than $\boldsymbol{P}_{\mathbf{max},\tau} = 10^{-9}$ (safety-critical user) the cell is highlighted in green; when lower than $\boldsymbol{P}_{\mathbf{max},\tau} = 10^{-5}$ (standard user) it is highlighted in orange.

overall probability is negligible and will have no impact on a safety-critical user. We therefore let $P_{\max,\tau} = 10^{-9}$ and $\tau = 1$ h. This means that we can tolerate a successful MAC forgery attack of one in a billion over a 1 h operation period.[3]

Standard user: In this case, we base our reliability assumptions on available GNSS documentation. In particular, the GPS Standard Positioning Service—Performance Standard [36] defines the following figure for GPS: *"1 × 10^{-5} Probability Over Any Hour of the SPS SIS Instantaneous URE Exceeding the NTE Tolerance Without a Timely Alert during Normal Operations (URE : User Range Error; NTE: Not-to-exceed)."*

As we do not consider a safety case for the standard user, we can take $1 \cdot 10^{-5}$ as a general reference error rate. We set $P_{\max,\tau} = 10^{-5}$ and maintain $\tau = 1$ h.

G. Determining MAC Size

Table I presents a summary of the parameters for determining the MAC output length S. Table II presents the resulting attack surfaces.

Table III presents, for safety-critical and standard user profiles, the success probability $P_{f,\tau}$ of a MAC forgery attack respective to the full set of parameters and MAC lengths between 16 and 80, as per [13].

Following our assumptions, Table III shows that for a safety-critical user, 30-bit MACs can be used as long as policy Nb-C is in effect. For a standard user, 17-bit MACs can achieve the desired risk level, if Nb-C is in effect, and 28-bit MACs otherwise. Table III therefore shows that MAC lengths shorter than the usual 32-bit minimum are sometimes sufficient, following a proper risk assessment. A GNSS or an authentication application can use this table indicatively and derive a MAC size complaint to their desired risk tolerance, namely $P_{\max,\tau}$.

Possibly a more intuitive way to view the robustness of the proposed MAC lengths is to calculate the expected time until a successful forgery attack i.e., the accumulated probability over a period longer than 1 h. For example, considering a 40-bit MAC with no failure treatment ($P_{f,\tau=1h} = 9.31 \cdot 10^{-10}$), an attack is expected to have one successful forgery every $\frac{\tau}{P_{f,\tau}} = \frac{1}{9.31\text{E}-10} \cong 1\,073\,741\,824$ h, which are approximately 122 489 years, even assuming a continuous 24/7 operation of the receiver and the attacker. If we take as a reference the 80-bit MAC with no failure treatment, this leads to one forgery attack every $1.35 \cdot 10^{17}$ years, or around ten million times the estimated age of the universe.

In conclusion, Table III shows that MAC lengths in the 30 to 40-bit range provide a good tradeoff for different users and authentication policies. Designers should also take into account the performance impact of failed authentications

[3]Note that considering here a 1 h operation period is more restrictive than considering a 150 s operation period. This is because there are 24 periods of 150 s in one hour thus a failure at the beginning of every 150 s period would amount to 24 failures in a full hour.

in the absence of attacks, and the complexity increase in the receiver logic resulting from specific failure treatment policies. We also recommend that GNSS providers adding authentication publish implementation guidelines explaining the desired authentication treatment in the receiver.

IV. FINDING A PREIMAGE IN A GNSS-TESLA KEYCHAIN

A limitation of the probabilistic approach considered above is that it can only attack one MAC at a time. For a persistent attack, the adversary must use the deterministic approach and recover the secret key before a successful spoof. However, for persistent spoofing it is not enough to recover the key corresponding to a single MAC, as was the case in the previous section, since this key is ephemeral and cannot be used to spoof datasets in the next block. Instead, the adversary must recover a key from which the ephemeral keys are generated. In the case of GNSS-TESLA, this is a preimage in the keychain. Therefore, the security of the key is reduced to that of the GNSS-TESLA keychain and, as the keychain is a chain of hashes, the generic attack complexity is determined by the hash output length, which corresponds to the key size.

A. Related Work

The TESLA key is a symmetric key used to generate the MAC. NIST recommends symmetric key sizes of at least 112 bits until 2030 and at least 128 bits from 2030 and beyond [37], and ENISA considers 128-bit security levels to be the minimum requirement for new systems being deployed [30]. ENISA recommends 80-bit values only for legacy systems or "*transactional data,*" i.e., data that has a very short validity period, and NIST directly mentions in [37] that a security strength of 80 bits "*is no longer considered adequate*". However, these recommendations are only partly applicable to TESLA keys: a brute-force attack over an ephemeral TESLA key can only be performed during the duration of a block (30s in our model), as opposed to the long-lived keys which drive the key sizes recommended in the standards. The place where the key size does play a dominating role is in the adversary's ability to attack the TESLA keychain, i.e., when it is considered as an output of a hash function, and this is the focus of the rest of this section.

Typical key lengths for GNSS-TESLA as per the previous literature range from 80 to 128 bits [31], [32]. This 80–128-bit range also appears in the first proposals for the OSNMA protocol [38], [39], later extended to 256 bits to deal with quantum attacks. Two references focus on the parameter selection of GNSS-TESLA. The model in [40] recommends values of more than 80 bits and provides a formal analysis claiming a loss of security which grows with the cryptoperiod. The model of [40] is also used in [41], which proposes a key length range of 115–125 bits as sufficient for safety applications such as SBAS, for a success probability in the order of 10^{-9}. In the sequel we propose a model with some different assumptions. For example, we do not assume that a hash function is a one-to-one mapping (i.e., a permutation) whereby collisions are possible only via truncation, and leading to the conclusion that there is some entropy loss which depends on the chain's cryptoperiod. Rather, our model considers this loss as an expected behavior of any random mapping whose range is smaller than its domain, in particular hash functions. We also consider that a secure hash function must be so for any input distribution. Abstracting both the input distribution and the output truncation into the hash function, which we consider secure, allows us to propose a simple model which crystalizes the relevant factors when determining a safe key length. Our model and resulting recommendations are presented in this section.

B. Attacks Against the TESLA Keychain

We model each TESLA key as part of a keychain generated using a one-way function as follows:

$$k_{i-1} = f(k_i) = h(k_i\,|t_{\mathrm{GNSS}}|\,\alpha) \tag{7}$$

where k_{i-1} is the key generated from and disclosed prior to k_i; $h(\cdot)$ is an n-bit hash function (i.e., its output size is n-bit long); t_{GNSS} is the GNSS time, α is a salt unique to every chain and disclosed just before the chain enters into force; and "$a|b$" is the concatenation of a and b. As in the previous section, we will assume that the synchronization process between the transmitters and the receiver is performed securely as part of the bootstrapping process.

A common method to obtain a one-way function is through a cryptographically secure hash function. A hash function is a function taking inputs of arbitrary length, producing an output of a fixed length n. Intuitively speaking, a secure hash function behaves like a random mapping. For any one-way function $h(\cdot)$, a preimage attack is an attack that, given Y, finds an X such that $Y = h(X)$. A cryptographically secure hash function has to satisfy, among other security requirements, that the work effort required for finding a preimage for a given output of length n is 2^n. Standardized hash functions (e.g., SHA-256 [42] and SHA-3 [43]) are the subject of long ongoing research, and are believed to offer this level of preimage-resistance.

In the context of GNSS, short keys are necessary in order to save bandwidth and the key size is determined by the output length of the hash function. The main driver for choosing the key length is to prevent preimage attacks, which are natural attacks on the TESLA chain. If after releasing the key an adversary can find a preimage for that key, she can use this value to authenticate spoofed messages, as per (2), but with the difference that knowing the key allows a successful forgery with probability 1 rather than 2^{-S}. It is important to note that a preimage can be found for any one-way function by means of exhaustive search, i.e., given an output value Y, by trying out all possible inputs until the right output is obtained.

C. Terminology

Let *f* be the key generation function as in (7), and *z* the chain length (i.e., the cryptoperiod). In order to generate a chain, we first choose uniformly a *seed-key* k_z in the interval [0,2n-1]. The previous key is generated as $k_{z-1} = f(k_z)$, and successive keys k_{i-1} are generated as $k_{i-1} = f(k_i)$. The last key in the chain used for authentication is k_1. When the chain is fully generated, $k_0 = f(k_1)$ is released and authenticated off-channel. We refer to k_0 as the *root key*.

For simplicity, we assume that a newly disseminated key is compared against the previous one in a single step, and refer to the key against which the new key is compared as the *effective key*, and to any key that is not part of the original chain as an *ineffective key*. Furthermore, we refer to a key k_i, leading to an effective key after *m* hashing steps as a *preimage of degree m*, i.e., k_0 is a preimage of degree 0 of k_0, k_1 is a preimage of degree 1 of k_0, etc.

D. Preimage Analysis

Given an effective key k_i, we can analyze its number of preimages as a binomial variable. The first observation is that, as k_i is part of a chain, it must have at least one preimage *ipso facto*. The rest of the 2^n-1 values are left to be mapped at random by the hash function which, by definition, maps each of them to a fixed value with probability $1/(2^n-1) \approx 2^{-n}$. For an effective key k_i, we get that the number of *additional* preimages can be modeled as a binomial random variable *X* having

$$X \sim B\left(2^n - 1, 2^{-n}\right) \tag{8}$$

with expected value exactly $\mu_X = (2^n-1) \cdot 2^{-n} \approx 1$. For an ineffective key, the expected number of preimages is μ_X. For an effective key, taking into account that in addition to any randomly mapped ineffective preimages, there is always an effective preimage in the chain, the expected number of preimages in $\mu_X + 1 \approx 2$, i.e., the previous key in the chain and possibly other ineffective keys.

By induction, we assume that an effective key k_i has *m* preimages of degree $m - 1$ on average. Then, the number of preimages of degree *m* for k_i is $m + 1$. As k_i is an effective key, one of the *m* preimages of degree $m-1$ is the effective key in the chain leading to k_i. This key has two preimages on average. Each of the other $m - 1$ preimages of degree $m - 1$ has a single preimage on average. We conclude that the number of preimages of degree *m* is $1 \cdot (m-1) + 2 \cdot 1 = m+1$. An example for this behavior is depicted in Fig. 3.

A preimage attack of one block requires a single hash (i.e., $k_i = f(k_{i+1})$). However, executing a forgery on one block requires two hashes. To see this, consider the state of the system and the public information in block *i*. Previously, in block $i-1$ the system disseminated D_{i-1}, $mac(k_i, D_{i-1})$, and k_{i-1} and all of these are now known to the adversary. Now, in block *i*, the system authenticates the data using k_{i+1} and will no longer accept data authenticated with k_i. The adversary guesses k_{i+1} and since she does not know k_i at this point, she must execute two hashing steps to compare

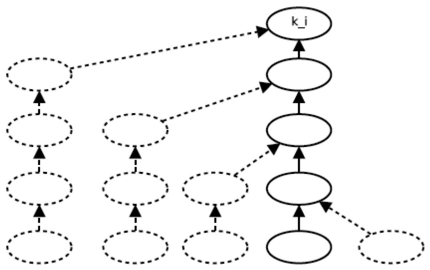


Fig. 3. Tree of chains ending in k_i. Keys along the original chain are connected via a solid line, while ineffective keys are drawn with a dotted line. An effective key (solid circle) has two incoming arrows: one from an effective key (solid circle), and one from an ineffective key (dotted circle) which was mapped at random according to (7); ineffective keys have only a single incoming arrow from a randomly mapped preimage according to (7).

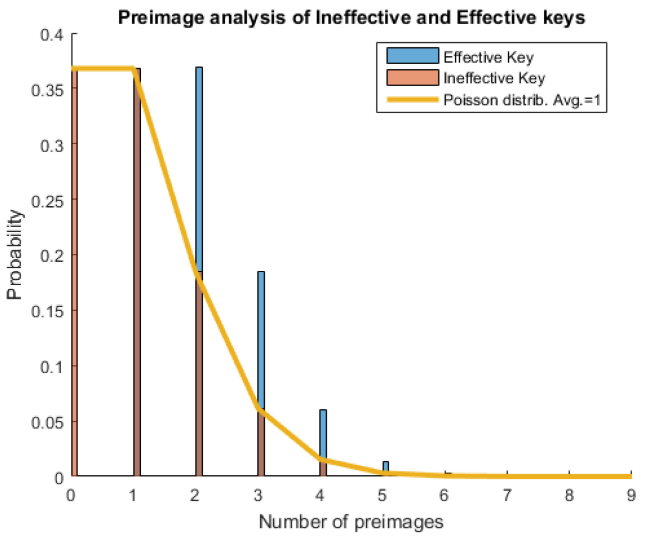


Fig. 4. Histogram of number of preimages for effective and ineffective keys over one SHA-256 iteration truncated to 16 bits, compared with a Poisson distribution.

$f(f(k_{i+1}))$ against the already known k_{i-1}. Therefore, a forgery attack over $\ell - 1$ blocks requires ℓ hashes.

In order to validate this model, we performed two experiments on a downscaled case. In the first experiment, we analyzed the preimage probability distribution over one iteration, and in the second, we measured the average number of preimages for a chain with several iterations. For the first experiment, we define our one-way function as SHA-256 truncated to 16 bits ($n = 16$). The probability of having zero to nine preimages for both effective and ineffective keys is shown in Fig. 4. The Figure also shows that the probability for an ineffective key to have *k* preimages, $P_{inef}(k)$, with a preimage→image random mapping function, follows a Poisson distribution:

$$P_{inef}\ (k) = \frac{e^{-\lambda}\lambda^k}{k!} \tag{9}$$

where the average is $\lambda = 1$, as expected. This result corresponds with the model proposed in [40, Sect III], with the difference that here it is used to model ineffective keys only. On the other hand, the probability distribution of the number of preimages for an effective key is equivalent to the

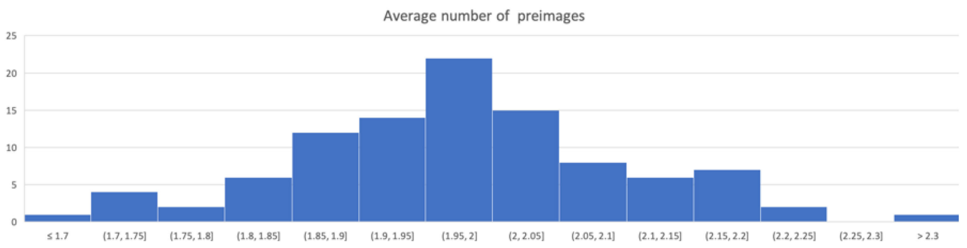


Fig. 5. Histogram of the average number of preimages for 128 keys in a TESLA-like chain, in 0.05-width intervals, for 100 experiments.

distribution of (9) but offset by one, the known preimage, and therefore with an average of 2.

In the second test, we measured the average number of preimages in a real one-way chain. We set $n = 20$ and $z = 128$. We defined the one-way function f as follows:

$$f\ (k_i) = k_i \oplus AES_Q^{\text{trunc} = 20}(k_i\,||0000|\,|c||\,00\ldots0) \quad (10)$$

where $k_i \parallel 0000 \parallel c \parallel 00\ldots0$ denotes the string consisting of k_i followed by four zero bits, an 8-bit counter c, and 96 zero bits. The counter c is incremented after each experiment. $AES_Q^{\text{trunc} = 20}(\cdot)$ denotes the truncation to 20 bits of an AES evaluation under the all-zero key. We performed 100 experiments with a new value of c for each new chain. We computed in each step the average number of (degree 1) preimages for all values on the chain. The results are shown in Fig. 5.

Note that the results from Figs. 4 and 5 are consistent with not only with the model in Fig. 3, but also with the downscaled numerical examples of [40], suggesting that it is sufficient to consider one random mapping and discern between effective and ineffective keys.

In the following, we will model an attack on the chain based on the success rates derived from the preimage analysis. Given an effective key k_i an attack requires creating a chain of length ℓ such that k_i is the last key of the chain. We consider two scenarios: an online attack, where the attacker waits until k_i is disclosed, and an offline attack, where the attacker tries different values of k_i before it is disclosed, and stores them in a table.

E. Online Attack

As we saw in the preimage analysis there are $\ell + 1$ starting points converging after ℓ steps to an effective key (see Fig. 3). In order to fully construct such a chain, ℓ hashes are required. Using a random starting point $\bar{k}_{i+\ell}$, the probability that such a candidate chain ends in a specific value k_i (i.e., an effective key), already disclosed, is

$$p = \frac{\ell + 1}{2^n}. \quad (11)$$

The expected number of starting points to be checked before finding a good key is inversely proportional to p, i.e., $1/p$. Given $\hat{t}$ the hashing time in seconds by the adversary, it takes $\ell \cdot \hat{t}$ seconds to generate a chain of ℓ keys, and it is expected to take

$$\frac{2^n}{\ell + 1}\ell \cdot \hat{t} = 2^n \cdot \hat{t}\frac{\ell}{\ell + 1} \quad (12)$$

seconds to test enough starting points to find a spoofed chain of length ℓ. Note that when $\ell = 1$, then $\ell/(\ell+1) = 0.5$, and when ℓ is large enough $\ell/(\ell + 1) \approx 1$. This means that the expected time for a successful attack would always be in the interval of $2^{n-1}\hat{t} \leq \tau \leq 2^n\hat{t}$ seconds, independently of the length of the attacked chain. This was already obtained in [40, Section V], where it was attributed to entropy loss. We attribute it to the fact that both models do not account for the true cost of verifying that a spoofed chain is mapped to an effective key. Accounting for this true cost, i.e., $\ell+1$, (12) becomes $2^n \cdot \hat{t}\,\frac{\ell+1}{\ell+1} = 2^n \cdot \hat{t}$ seconds, regardless of the length of the attacked chain. Moreover, both models assume that the key to verify a block is disseminated in the following out. Supposing a higher latency GNSS-TESLA where the key is disseminated 1000 blocks later, (12) would become $2^n \cdot \hat{t}\,\frac{\ell+1001}{\ell+1}$ which would suggest that there is an "entropy gain" for short chains. In fact, what we see is that the fixed overhead for verifying that a chain is valid is more significant in short chains than in long ones.

F. Offline Attack

In an offline attack, the adversary tries to attack a future effective key k_i by generating chains of length ℓ with endpoints $k_i^0, \ldots, k_i^{m-1}$ and storing them in a large database. A clever way to build such a database and a generic attack were proposed by Hellman in [44] and also analyzed in [41] for GNSS-TESLA. We will describe a variant of [44] adapted to our setting: As soon as the chain salt α and t_{GNSS} (7) are disclosed, the adversary chooses a block i, and a chain length ℓ she wishes to attack. She chooses a starting point $\bar{k}_{i+\ell}^0$ and computes its respective endpoint $\bar{k}_i^0 = f^{\ell}\,(\bar{k}_{i+\ell}^0)$. She stores the pair $(\bar{k}_{i+\ell}^0, \bar{k}_i^0)$ in a look-up table, and repeats this process for many starting points. When the target key k_i is finally released, the adversary searches the table to see whether any value $\bar{k}_i^j = k_i$ was previously stored. If there is such a value, then its pair is the beginning of a spoofed chain. As before, since a chain of length ℓ has $(\ell+1)$ possible starting points converging to the same endpoint, the probability of randomly selecting a good starting point is $(\ell+1)/2^n$. Thus, the time required for a successful attack is the same as for the online attack: for a hashing time of $\hat{t}$, it takes $\ell \cdot \hat{t}$ seconds to generate a single chain, and the time it takes to try the required $2^n/(\ell+1)$ starting points is $\ell \cdot \hat{t} \cdot 2^n/(\ell+1)$, which converges to $2^n\hat{t}$ seconds for high values of ℓ, and is always more than $2^{n-1}\hat{t}$ seconds.

TABLE IV
Chain Attack Time (τ_{CA}) for GNSS-TESLA Key Sizes (n) Between 70 and 128 Bits, for a Hashing Rate of 2^{-57} Seconds-Per-Hash

Key size (n):	**70**	**72**	**80**	**82**	**88**	**90**	**96**	**104**	**112**	**120**	**128**
Seconds	4096	16384	4.19E+06	1.68E+07	1.07E+09	4.29E+09	2.75E+11	7.04E+13	1.80E+16	4.61E+18	1.18E+21
Days			48.55	194.18	1.24E+04	4.97E+04	3.18E+06	8.14E+08	2.08E+11	5.34E+13	1.37E+16
Years					34.04	136.18	8,716	2.23E+06	5.71E+08	1.46E+11	3.74E+13

Note that if the attacker had certainty about the values of α and t_{GNSS} it could start the offline attack before the root-key is disclosed. Since both are known only during the cryptoperiod of the chain, the adversary is limited in the amount of precomputation. Otherwise, if the adversary wishes to overcome this and start building databases before α is made known, she needs to guess α thus increase the work effort by a factor of 2^{α}. She also needs to predict the time associated to the root-key, so adding uncertainty to this value is recommended in [40].

G. Recommendations on Key Lengths and Cryptoperiods for GNSS-TESLA

Based on the above analysis, in order to choose the GNSS-TESLA key length, we recommend lower bounding the attack time by $2^{n-1}\hat{t}$ seconds, i.e., considering attacks on very short chains with frequent dissemination as a reference. This decision bounds attacks on any chain length at the cost of at most a single bit compared to longer chains and slower dissemination. In order to define $\hat{t}$, we can look at the currently available hashing devices. As per [45], a 1000-USD Antminer R4 can perform 8.6 TH/s (tera-hashes per second). Assuming that the timestamp and salt addition do not alter the hashing rate, and a 10-Million USD investment mounting parallel hashing devices, this would allow $8.6\dot{c}10^{16}$ hashes-per-second, or, rounding up, $\hat{t} = 2^{-57}$ seconds-per-hash. Further costs include energy consumption, cooling costs, and sufficient and efficient storage devices for the offline attack. This puts the attack way beyond the reach of most reasonable adversaries. We take 10 Million USD as a reference for the attack resources, taking into account standard practical security considerations such as the value of the authenticated information and its short applicability period [46], and the fact that signal re-radiation, even if expensive and sophisticated, may be more cost-effective to spoof a GNSS position [10].

1) *Hash Output Size:* In order to facilitate this and future analyses, we define the Chain Attack Time τ_{CA} as follows:

$$\tau_{CA} = 2^{n-\hat{h}-1} \tag{13}$$

where $\hat{h}$ is the two-exponent of the hashing rate, $\log_2(1/\hat{t})$, so $\hat{h} = 57$ in our case.

Table IV presents τ_{CA} for different values of GNSS-TESLA key lengths between 70 and 128 bits. The table shows that a key of 80 bits would allow one key recovery every 48.55 days and it is therefore not recommended. In the case of 96 bits, one successful attack is expected once every 8716 years on average. If we extrapolate a hashing rate to 10 years from now with a 1.5/year growth factor, which overestimates Moore's Law, then a successful attack is expected "only" once every 302 years on average.

2) *Chain Cryptoperiod:* Since the attack complexity is roughly the same when attacking short and long chains, we can assume that a successful key recovery implies that the rest of the chain was compromised entirely. In the worst-case scenario, the attack is successful already in the first attempt, compromising the entire chain.

In this setting, the cryptoperiod of the chain, i.e., its length, can be used to compartmentalize the breach and mitigate the adversary's ability to mount a persistent attack. Consider for example a parameter set where one breach is expected in every 120-day period. In the worst-case scenario, this breach would happen at the beginning of the chain compromising all of it. If the chain has also a 120-day cryptoperiod, an adversary would be able persistently to spoof data for the next 120 days. If the same 120-day period is instead covered by four 30-day chains, the spoofing period following a successful chain attack is reduced to 25% of that time, i.e., 30 days following a successful attack at the beginning of one of the chains.

3) *Combining the Hash Output Size and the Chain Cryptoperiod:* Considering the hash output size and the chain cryptoperiod together allows to derive the average spoof-per-cost period. Consider for example a 120-day chain with 80-bit keys. From Table IV we expect an average of one forgery every 48.55 days. We can derive that an adversary would be able to mount a persistent spoof for an average period of $120 - 48.55 = 71.45$ days before the chain is replaced. Alternatively, setting the cryptoperiod to 30 days with the same key length, the adversary would be able to spoof an average period of about $60 - 48.55 = 11.45$ days in every 60-day period (corresponding to two chains), or $2 \cdot 11.45 = 22.9$ days in a 120-day period (corresponding to four different chains).

Another approach would be to derive the worst-case success probability and use it to derive the attack cost. For example, to mount a 120-day attack on an 80-bit chain with a 120-day cryptoperiod the adversary would need to mount a single preimage attack with success probability 2^{-80}. Conversely, to mount a 120-day attack on four 80-bit chains with a 30-day cryptoperiod each, the adversary would need to mount four preimage attacks (one on each chain) with an overall probability of $2^{-80 \cdot 4} = 2^{-320}$.

4) *Summary:* In summary, comparing the roles of the hash output length and the chain cryptoperiod we see that the former determines the difficulty of finding a preimage and the latter the difficulty of using such a preimage to mount a persistent attack.

According to Table IV, and without accounting for future improvements in computing power, we find that hash output lengths of 88 bits or more appear to be resilient to preimage attacks within several decades. Hash output sizes of 96 bits or more are expected to remain resilient within any reasonable timeframe, even considering Moore's Law.

We further see that even if a preimage attack cannot be avoided (e.g., due to short hash output sizes), the effect of shorter keys can be mitigated by limiting the chain's cryptoperiod. Frequently replacing the chain would prevent an adversary from mounting persistent spoofing attacks even if they are successful in finding a preimage.

Note also that extending the key by a single bit requires the adversary to double her efforts to maintain the same success rate, hence extending the key length by three bits every two years would compensate for likely improvement in computing power. In any case, the crypto algorithms and their parameters should be re-evaluated once a year, taking into account changes in standards, improvement of cryptanalytic techniques, decrease in hardware costs, and disruptive technologies such as quantum computers.

In summary, while shorter key values appear to be secure nowadays, we recommend GNSS-TESLA chains with a cryptoperiod of one year or less, and a hash output (i.e., a TESLA key) size of 96 bits or more. Such parameters seem to provide enough safety margin for today's needs and those of the near future. This recommendation is the result of an application-oriented analysis and is shorter than standard cryptographic practices of 112 or 128 bits.

V. CONCLUSION

This article analyzed the length of MACs and hash-derived cryptographic keys for delayed-disclosure protocols, such as TESLA, applied to GNSS. Regarding MAC sizes, values of 17–40 bits are considered sufficient, depending on the user profile and receiver treatment policies. In particular, for the simplest, most lenient treatment policy in the receiver, values of 28 and 40 bits are considered safe, for standard and safety-critical users, respectively. Regarding TESLA key sizes, values of 96 or more bits are suggested. These sizes are shorter than those recommended by standardization bodies such as [24] or [30]. This discrepancy is due to the more restrictive application model we consider.

Regarding MACs, a standardization body providing a general recommendation must take into account a forgery attack against the MAC function, where the MAC and its corresponding key are long-lived. However, the case for GNSS-TESLA is different. In practical terms, a generic forgery attack against the MAC is restricted to random guesses, and since the MAC is also short-lived, we conclude that MAC sizes shorter than the standard recommendation can be used in some conditions. We then analyze the risk of a key recovery attack targeting the key generation mechanism, i.e., the TESLA chain. The adversary tries to find a preimage of a known output of a hash function (i.e., the key) by attacking the entire chain or parts of it. The analysis on preimage attacks allows us to conclude that shorter-than-recommended hash output lengths (i.e., MAC key sizes) are sufficient. This analysis is based on a notably simpler model than that in previous literature, validated as part of this work.

While the recommendations are tailored for GNSS-TESLA, they may be of use for other delayed-disclosure protocols, at data or signal level, used for GNSS authentication.

Our work also recommends that GNSS authentication providers publish clear implementation guidelines explaining the minimum or desired authentication treatment in the receiver.

ACKNOWLEDGMENT

The content of this article does not necessarily reflect the official position the authors' organizations. Responsibility for the information and views set out in this article lies entirely with the authors.

REFERENCES

[1] K. D. Wesson, J. N. Gross, T. E. Humphreys, and B. L. Evans
GNSS signal authentication via power and distortion monitoring
IEEE Trans. Aerosp. Electron. Syst., vol. 54, no. 2, pp. 739–754, Apr. 2018.

[2] J. N. Gross, C. Kilic, and T. E. Humphreys
Maximum-Likelihood power-distortion monitoring for GNSS-Signal authentication
IEEE Trans. Aerosp. Electron. Syst., vol. 55, no. 1, pp. 469–475, Feb. 2019.

[3] Ç. Tanıl, S. Khanafseh, M. Joerger, and B. Pervan
An INS monitor to detect GNSS spoofers capable of tracking vehicle position
IEEE Trans. Aerosp. Electron. Syst., vol. 54, no. 1, pp. 131–143, Feb. 2018.

[4] European Commission, Brussels, Belgium, Commission Implementing Decision (EU) 2017/224 of 8 February 2017 (CS Implemeting Act), 2017.

[5] J. M. Anderson *et al.*,
Chips-message robust authentication (Chimera) for GPS civilian signals
In *Proc. 32nd Int. Tech. Meeting Satell. Division Inst. Navig.*, 2017, pp. 2388–2416.

[6] R. Hirokawa and S. Fujita
A message authentication proposal for satellite-based nationwide PPP-RTK correction service
In *Proc. 34th Int. Tech. Meeting Satell. Division Inst. Navig.*, 2019, pp. 1798–1811.

[7] C. Wullems, O. Pozzobon, and K. Kubik
Signal authentication and integrity schemes for next generation global navigation satellite systems
In *Proc. Eur. Navig. Conf.*, 2005.

[8] G. T. Becker, S. Lo, D. D. Lorenzo, D. Qiu, C. Paar, and P. Enge
Efficient authentication mechanisms for navigation systems – a radio-navigation case study
In *Proc. 22nd Int. Tech. Meeting Satell. Division Inst. Navig.*, 2009, pp. 901–912.

[9] K. Wesson, M. Rothlisberger, and T. Humphreys
Practical cryptographic civil GPS signal authentication
Navig., J. Inst. Navig., vol. 59, no. 3, pp. 177–193, 2012.

[10] T. Humphreys
Detection strategy for cryptographic GNSS anti-spoofing
IEEE Trans. Aerosp. Electron. Syst., vol. 49, no. 2, pp. 1073–1090, Apr. 2013.

[11] G. Caparra, S. Sturaro, N. Laurenti, C. Wullems, and R. T. Ioannides
A novel navigation message authentication scheme for GNSS open service
In *Proc. Int. Tech. Meeting Satell. Division Inst. Navig.*, 2016, pp. 2938–2947.

[12] C. Sarto *et al.*,
Implementation and testing of OSNMA for Galileo
In *Proc. 32nd Int. Tech. Meeting Satell. Division Inst. Navig.*, 2017, pp. 901–912.

[13] European Commission, Brussels, Belgium Galileo navigation message authentication specification for signal-in-space testing – v1.0
2016.

[14] A. M. Neish
Establishing trust through authentication in satellite based augmentation systems
Ph.D. dissertation, Stanford Univ., Stanford, CA, USA, 2020.

[15] A. Neish, T. Walter, and J. D. Powell
SBAS data authentication: A concept operations
In *Proc. 34th Int. Tech. Meeting Satell. Division Inst. Navig*., 2019, pp. 1812–1823.

[16] I. Fernández-Hernández *et al.*,
Impact analysis of SBAS authentication
Navig., J. Inst. Navig., vol. 65, no. 4, pp. 517–532, 2018.

[17] L. Scott
Anti-Spoofing & authenticated signal architectures for civil navigation systems
In *Proc. 16th Int. Tech. Meeting Satell. Division Inst. Navig.*, 2003, pp. 1543–1552.

[18] IT Security Techniques – Lightweight Cryptography – Part 7: Broadcast Authentication Protocols, ISO/IEC 29192-7, 2018.

[19] A. Perrig, R. Canetti, J. Tygar, and D. Song
Efficient authentication and signing of multicast streams over lossy channels
In *Proc. IEEE Symp. Secur. Privacy*, 2000, pp. 56–73.

[20] S. Lo and P. Enge
Aviation augmentation system broadcasts
In *Proc. IEEE/ION Position Location Navig. Symp.*, 2010, pp. 708–717.

[21] I. Fernandez-Hernandez, V. Rijmen, G. Seco-Granados, J. Simon, I. Rodriguez, and J. D. Calle
A navigation message authentication proposal for the Galileo open service
Navig., J. Inst. Navig., vol. 63, no. 1, pp. pp. 85–102, 2016.

[22] European Commission, Brussels, Belgium Galileo navigation message authentication specification for signal-in-space testing – v1.1
2018.

[23] Information Technology - Security Techniques - Message Authentication Codes (MACs) - Part 1: mechanisms Using a Block Cipher, ISO/IEC 9797-1:2011, 2011.

[24] The Keyed-Hash Message Authentication Code (HMAC), FIPS PUB 198-1, 2008.

[25] Recommendation for block cipher modes of operation: The CMAC mode for authentication
NIST, Gaithersburg, MD, USA, NIST Special Publication 800-38B, 2004.

[26] A. Menezes, P. Van Oorschot, and S. Vanstone,
Handbook of Applied Cryptography. Boca Raton, FL, USA: CRC Press, 1997.

[27] I. Fernandez-Hernandez, T. Walter, A. Neish, and O'C. Driscoll
Independent time synchronization for resilient GNSS receiv.
In *Proc. Int. Tech. Meeting Inst. Navig*., 2020, pp. 964–978.

[28] Q. Dang,
Recommendation for applications using approved hash algorithms
NIST, Gaithersburg, MD, USA, NIST Special Publication 800-107 - Revision 1, 2012.

[29] Information Technology - Security Techniques - Message Authentication Codes (MACs) - Part 2: Mechanisms Using a Dedicated Hash Function, ISO/IEC 9797-2:2011, 2011.

[30] ENISA, Heraklion, Greece, "Algorithms, key sizes and parameters report - recommendations
2013.

[31] I. Fernández-Hernández
GNSS authentication: Design parameters and service concepts
In *Proc. Eur. Navig. Conf.*, 2014.

[32] J. T. Curran, M. Paonni, and J. Bishop
Securing the open-service: A candidate navigation message authentication
In *Proc. Eur. Navig. Conf.*, 2014.

[33] European Union, Brussels, Belgium OSSISICD: Open service signal in space interface control document, issue 1.3
2016.

[34] GPS Directorate, NAVSTAR GPS space segment/navigation user segment - IS-GPS-200 - K
U.S. Coast Guard Navigation Center, Alexandria, VA, USA, 2019.

[35] International Civil Aviation Organisation International standards and recommended practices - Annex 10 - Aeronautical Telecommunications - Vol 1 - Radio Navigation aids - Sixth Edition - No. 90
ICAO publications, 2016.

[36] Global Positioning System Standard Positioning Service Performance Standard, 4th ed., U.S. DoD, Washington, DC, USA, 2008.

[37] Recommendation for Key Management - Part 1: General, Special Publication 800-57 Part 1 Revision 5, 2020.

[38] I. Fernandez-Hernandez, V. Rijmen, G. Seco-Granados, J. Simón, and I. Rodríguez
Design drivers, solutions robustness assessment navigation message authentication for Galileo open service
In *Proc. 27th Int. Tech. Meeting Satell. Division Inst. Navig.*, 2014, pp. 2810–2827.

[39] P. Walker *et al.*,
Galileo open service authentication: A complete service design and provision analysis
In *Proc. 28th Int. Tech. Meeting Satell. Division Inst. Navig.*, 2015, pp. 3383–3396.

[40] G. Caparra, S. Sturaro, N. Laurenti, and C. Wullems
Evaluating the security of One-way key chains in TESLA-based GNSS navigation message authentication schemes
In *Proc. IEEE Int. Conf. Localization GNSS*, 2016, pp. 1–6.

[41] A. Neish, T. Walter, and P. Enge
Parameter selection for the TESLA keychain
In *Proc. 31st Int. Tech. Meeting Satell. Division Inst. Navig.*, 2018, pp. 2155–2171.

[42] Secure Hash Standard (SHS), FIPS PUB 180-4, 2012.

[43] permutation-Based Hash and Extendable-Output Functions, FIPS PUB 202: SHA-3 standard, 2015.

[44] M. Hellman
A cryptanalytic time-memory trade-off
IEEE Trans. Inf. Theory, vol. 26, no. 4, pp. 401–406, Jul. 1980.

[45] buybitcoinworldwide.com
2020. [Online]. Available: https://www.buybitcoinworldwide.com/mining/hardware/. Accessed on: Mar. 2020.

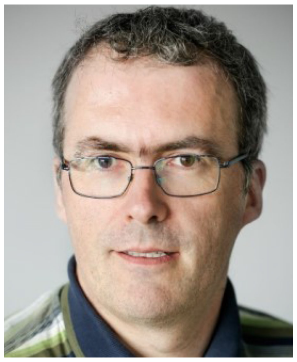
[46] F. Piper and S. Murphy
Cryptography – A Very Short Introduction. London, U.K.: Oxford Univ. Press, 2002.

Ignacio Fernández-Hernández received the electronic engineering degree from ICAI, Madrid, Spain, in 2001, the M.B.A. degree from LBS, London, U.K., in 2011, and the Ph.D. degree in electronic systems from Aalborg University, Aalborg, Denmark, in 2015, for his research on snapshot positioning and GNSS authentication.

He works for the European Commission DG DEFIS, where he has led the definition of authentication and high-accuracy services for Galileo. He was a Visiting Scholar with the GPS Laboratory, Department of Aeronautics and Astronautics, Stanford University, in 2017, and currently is a Visiting Professor with KU Leuven, Leuven, Belgium.

Tomer Ashur received the Ph.D. degree from KU Leuven, Leuven, Belgium, in 2017, after completing the dissertation on Cryptanalysis of Symmetric-Key Primitives.

He is an FWO Postdoctoral-Fellow with KU Leuven, and a Visiting Assistant Professor with TU Eindhoven, Eindhoven, The Netherlands. He was Teaching Assistant with the University of Haifa; CIO of Mediton Healthcare Services; a Project Manager with Katz Delivering Services; Head of Support with Safend Inc., and a communication Officer (OF-2) with the Israel Defense Forces.



Vincent Rijmen received the graduate degree from KU Leuven, Leuven, Belgium, in 1993.

He finished his doctoral dissertation on the design and analysis of block ciphers, in 1997. He is co-designer of the algorithm Rijndael, which in October 2000 was selected by the National Institute for Standards and Technology (NIST) to become the Advanced Encryption Standard (AES), the successor to the existing Data Encryption Standard (DES). Next to the design and analysis of encryption algorithms, his second main research interests include implementation techniques that counter side-channel attacks. He has been a Chief Cryptographer of Cryptomathic, a European company developing software for cryptographic applications, and a Professor with the institute IAIK of the Graz University of Technology (Austria). He is currently a Full Professor with the Department of Electrical Engineering, KU Leuven, and an Adjunct Professor with the Department of Informatics, University of Bergen, Bergen, Norway.