Received: 11 May 2022 | Revised: 1 February 2023 | Accepted: 23 February 2023

DOI: 10.33012/navi.595

ORIGINAL ARTICLE

Authentication of Satellite-Based Augmentation Systems with Over-the-Air Rekeying Schemes

Jason Anderson* | **Sherman Lo** | **Andrew Neish** | **Todd Walter**

Aeronautics and Astronautics
Stanford, California
United States of America

Correspondence
Jason Anderson
Email: jand271@stanford.edu

Abstract

Here we delineate a complete satellite-based augmentation system (SBAS) authentication scheme, including over-the-air rekeying (OTAR), that uses the elliptic curve digital signature algorithm (ECDSA) and timed efficient stream loss-tolerant authentication (TESLA) without the quadrature (Q) channel. This scheme appends two new message types to the SBAS scheduler without overburdening the message schedule. We have taken special care to ensure that our scheme (1) meets the appropriate security requirements needed to prevent and deter spoofing; (2) is compatible with existing cryptographic standards; (3) is flexible, expandable, and future-proof to different cryptographic and implementation schemes; and (4) is backward compatible with legacy receivers. The scheme accommodates a diverse set of features, including authenticating core-constellation ephemerides. We discuss the SBAS provider and receiver machine state and its startup, including its use by aircraft that traverse differing SBAS coverage areas. We tested our scheme with existing SBAS simulation and analysis tools and found that it had negligible effects on current SBAS availability and continuity requirements.

Keywords

authentication, over-the-air rekeying, SBAS, TESLA

1 | INTRODUCTION

In this work, we delineate a complete satellite-based augmentation system (SBAS) authentication scheme, including over-the-air re-keying (OTAR) and discuss how this proposed scheme meets necessary security levels and desirable traits for SBAS stakeholders, including backward compatibility, data efficiency, and quick time to first authenticated fix (TFAF). Moreover, this new scheme can be expanded in response to additional stakeholder feedback. This work addresses the complete authentication scheme design, including the connecting receiver hardware requirements needed for maintenance schedules, key updates, and scheme maintenance, and uses a full-stack Monte-Carlo SBAS simulation to test and evaluate its performance. This work builds on and expands on our previous work (Anderson et al., 2021) and includes updated security details as well as additional results and definitions based upon SBAS Stakeholder feedback.

Licensed under CC-BY 4.0 © 2023 Institute of Navigation

SBASs, such as the wide-area augmentation system (WAAS) used in the United States, among other international equivalents, have become integral to the global navigation satellite system (GNSS) used in civilian aviation. International parties that choose to implement an SBAS (each is known as a Provider) use listening stations around their service volume to assess GNSS satellite positioning data and broadcast corrections widely via geostationary satellites. This information includes wide-area differential GNSS corrections and GNSS satellite information such as its health and integrity. Similar to most GNSS core-constellation signals, the SBAS signal is open and susceptible to spoofing. Given its ubiquitous use in civilian aviation, SBAS should be augmented with spoofing-resistant capabilities to ensure ongoing civilian aviation safety. As Providers agree to share a common SBAS message standard, our work seeks to specify how SBASs can be augmented to provide authenticated service that is resistant to spoofing.

SBAS is primarily a data service. It broadcasts data that assists GNSS users. Therefore, appending additional cryptographic data to the SBAS data would be a natural way to authenticate SBAS data for civilian users. Additional SBAS broadcast messages could deliver cryptographic signatures and key values to its users. Using the mathematical primitives underlying cryptographic authentication methods, users could assert that only a Provider was capable of generating a given set of SBAS data as well as the accompanying authenticating pseudorandom data. In this work, we refer to the authenticating data as "signatures". Signatures, together with the associated key data, are either "authenticating pseudorandom data" or "OTAR Segment", which are terms that refer to the cryptographic pseudorandom data itself or the chunks separated for transmission to a receiver, respectively. The term "authenticating pseudorandom data" is used because the data are not human-readable nor are they predictable without the use of private secrets. The security of the authenticating pseudorandom data assumes that (1) the Provider is the exclusive holder of certain secret identifying information and (2) there are no known efficient algorithms that can generate the authenticating pseudorandom data without the secret identifying information. If the identifying information (e.g., keys) is leaked, that information is then compromised and must be revoked. If an efficient algorithm is discovered, the relevant cryptographic primitives are known as broken and must be replaced.

The use of cryptographic authentication methods poses challenges to SBASs. The main challenge relates to the delivery of authenticating pseudorandom data via SBAS given current data-bandwidth constraints. Because SBAS is an open signal, secure SBAS authentication must rely on asymmetric cryptographic algorithms, for example, the elliptic curve digital signature algorithm (ECDSA). In this paper, we use the term ECDSA to include other, similar asymmetric cryptographic algorithms (e.g., EC-Schnorr). However, we will specify ECDSA without losing generality for concreteness, noting that certain parameters and characteristic security strengths listed here would be different if we were not using the ECDSA. A single ECDSA signature requires 512 bits to achieve the standard 128-bit security level, which dwarfs the 216 data bits permitted per SBAS message.

Some prior art has suggested the use of the quadrature (Q) channel to deliver authenticating pseudorandom data (Fernandez-Hernandez et al., 2021; Neish, Walter, & Powell, 2019); however, those solutions would require power currently used by the In-phase (I) channel. Use of the Q channel would strain the availability and continuity of SBAS systems at coverage area boundaries and will thus be undesirable to SBAS stakeholders. Other prior art suggested the use of a combination of ECDSA with another algorithm known as timed efficient stream loss-tolerant authentication (TESLA) (Fernández-Hernández et al., 2016; Neish, 2020; Various,

2021). Use of this algorithm provides more efficient use of authenticating pseudorandom data and is loss-tolerant. TESLA uses a delayed-release mechanism to authenticate data and requires less authenticating pseudorandom data than ECDSA. However, TESLA requires the Provider and the user to be loosely time-synchronized (Perrig et al., 2005). SBAS cannot use TESLA exclusively; TESLA must be used in tandem with ECDSA to achieve authentication security. In this work, we establish the following relationship between the proposed TESLA-ECDSA scheme. Using this scheme, TESLA authenticates the SBAS messages and ECDSA authenticates the SBAS's use of TESLA for periodic maintenance. While prior art has identified TESLA and ECDSA scheme parameters required to achieve authentication, the maintenance and maintenance requirements, such as how best to perform OTAR, remain largely unaddressed. This work addresses this challenge by suggesting a more efficient authentication maintenance scheme that does not require use of the Q channel. Moreover, this work leverages specific features of TESLA scheme to assert security efficiently (Caparra et al., 2016) and permit the introduction of additional features relevant to SBAS stakeholders.

Another challenge lies with receiver computational considerations. Some prior art has explored how TESLA and ECDSA computations would fare when used in GNSS, SBAS, and smartphone contexts (Cancela et al., 2019). TESLA frequently requires a more intense, one-time startup hashing computation upon receiver start followed by minimal hashing operations during standard operation. We note that modern commodity electronics frequently perform these operations and often include hardware-specific acceleration functions in their chips to increase computational efficiency and facilitate parallelism. Therefore, we expect that if these methods burden current receivers, manufacturers could augment their chips at a minimal cost to accommodate the desired computational loads.

Prior art suggested appending a single message type (MT) to SBAS for authentication and maintenance (Neish, 2020). Findings from this work suggest that SBAS might send this specific MT every six messages to deliver 190-bits of TESLA authentication data and 26-bits for scheme maintenance. While the scheme requires frequent delivery of the authentication-message, it will not overburden the SBAS MT schedule. We modified this earlier work by appending another MT to SBAS to replace the 26-bits mentioned above that was to be dedicated to maintenance. Our proposed additional MT was designed to be modular for the exclusive purpose of delivering all authenticating pseudorandom data related to scheme maintenance. This design allows for reasonable TFAF requirements and is agnostic to the ECDSA and TESLA scheme parameters. Therefore, minimal changes will be needed in the event of cryptographic primitive breakage. Moreover, the additional MT scales well with increases to security level requirements. It is also flexible and future-proof to accommodate anticipated feedback from Providers and SBAS stakeholders.

To evaluate the proposed method against prior art, we implement a full-stack SBAS simulation of our design by augmenting an existing SBAS simulation tool known as the Matlab Algorithm Availability Simulation Tool (MAAST) (Jan et al., 2001). MAAST was used previously to evaluate SBAS design and provides the results of a Monte Carlo simulation performed to evaluate how our design performs under message loss over the WAAS coverage area. Results using this tool revealed that our proposed design outperforms key performance indicators (KPIs), such as a shorter TFAF, a shorter time to authentication per message, and less sensitivity to loss tolerance, compared to other ideas currently under consideration.

When using TESLA, the Provider and the Users must be loosely time-synchronized. This poses an interesting "Catch-22" situation because the GNSS provides the time function. Therefore, our scheme must also include a

mechanism to resist a Replay Attack. A Replay Attack describes a situation in which a spoofer can listen and then replay messages at a slightly delayed rate. After some time, these induced time delays will allow the spoofer to violate the loose time-synchronized assumption and thus break the TESLA scheme. There are mechanisms described in prior art that receivers can use to establish trust in GNSS ranging signals (Fernandez-Hernandez et al., 2019; Psiaki & Humphreys, 2016); however, GNSS ranging signals have not yet been rigorously authenticated with cryptography. Prior art has also described how an onboard receiver clock might be used to detect this type of attack, given clock uncertainty (Fernandez-Hernandez et al., 2020). Other studies have investigated clock hardware and models that could be incorporated into receivers to enforce the synchronization assumption (Ardizzon et al., 2022). This work extends this prior art by discussing how onboard clocks, external clocks, and maintenance conditions can be used to mitigate the threat of Replay Attacks.

1.1 | Elliptic Curve Digital Signature Algorithm (ECDSA)

ECDSA is a standardized asymmetric authentication protocol. As we have not included all details in the following summary, we refer the reader to several widely-available detailed definitions in the cryptographic literature or on the Internet (Boneh & Shoup, 2017). The protocol specifies a signing function and a verifying function. For this protocol, let n be the security level of an instance of the protocol, which linearly describes the required computation that will exhaustively break the instance. The Provider generates a secure random $2n$-bit integer for long-term use as a secret private key. The Provider then derives a $2n$-bit integer from the private key and distributes it to receivers as a public key. The Provider then uses the signing function with the secret private key to derive signatures on messages. Each signature is a 2-tuple of $2n$-bit integers for a total of $4n$ bits. Receivers use the verifying function with the public key and signatures to assert that the private key holder generated the message and the signature. Because the protocol is secure, there is no known efficient algorithm that can compute the private key given the public key nor any that can compute the signature on a message without the private key. The protocol assumes that the receiver trusts that the public key is from Provider. The protocol is not loss-tolerant; if a single bit of a signature or message is lost, the protocol will fail to provide verification. Likewise, this protocol is not future-proof to attacks that might be conducted from theoretical quantum computers, as described by Neish, Walter, & Enge (2019).

1.2 | Timed Efficient Stream Loss-Tolerant Authentication (TESLA)

TESLA is an authentication protocol that allows a receiver to authenticate messages from a Provider when used in tandem with other asymmetric authentication protocols. This protocol poses several relevant advantages over a purely asymmetric systems used by GNSS and SBAS systems. First, the protocol requires less authenticating pseudorandom data to authenticate messages from a Provider. Second, the protocol is loss-tolerant. Third, the computation required for receiver authentication is less strenuous. Figure 1 presents a conceptual diagram of the TESLA description to follow. Algorithms 1 and 2 provide a more concrete description of the protocol.

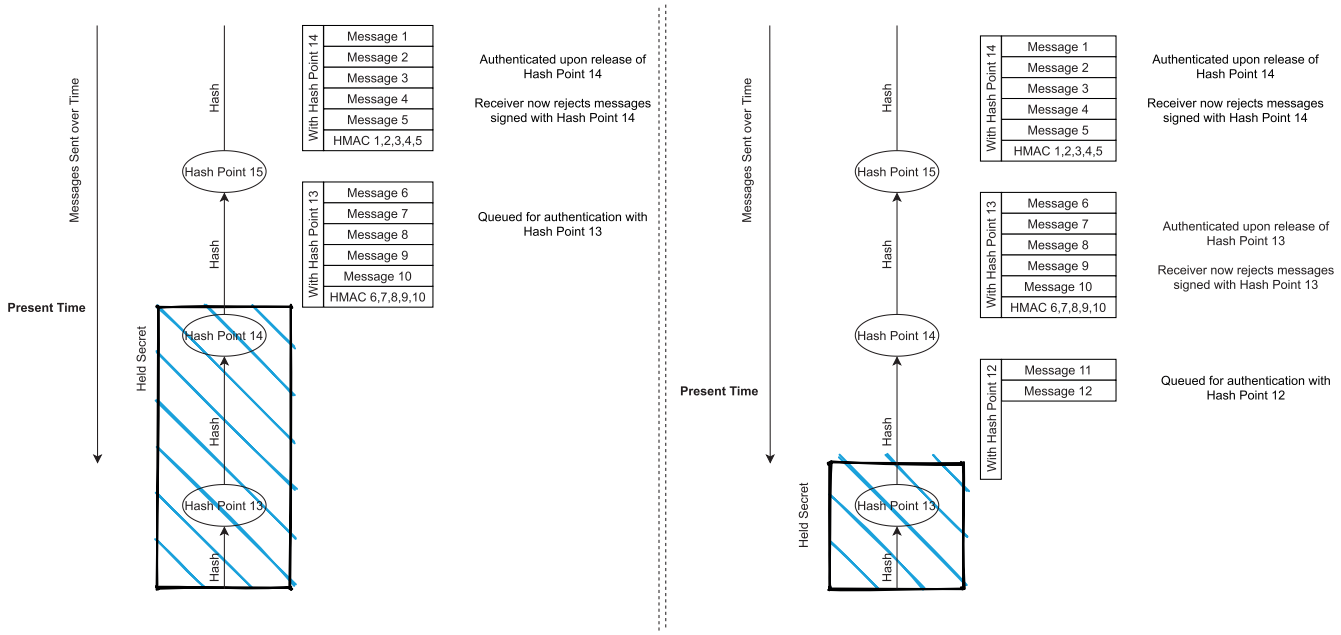

FIGURE 1 Conceptual diagram of TESLA that demonstrates the delayed-release key secrecy schedule
The right section of the diagram follows the left section in time. The diagonally cross-hashed boxes include information held secret by the Provider. The box recedes each time a hash point is released.

ALGORITHM 1
Provider Procedures for Single Satellite-Single Frequency Authenticated Message Distribution with TESLA

Generate a new Hash Path P with points $p_2^P \ldots p_{n+1}^P$ via Equation (1) and a secure random p_1^P
Broadcast p_{n+1}^P signed by a Level-2 ECDSA key and distributed via MT51
for $i = [n, n-1, \cdots, 2, 1]$ **do**
  Broadcast any 5 messages, called $m_1 \cdots m_5$
  Derive $k_1 \cdots k_5$ from Equation (2) with p_i and the sending time of $m_1 \cdots m_5$
  Derive $s_1 \cdots s_5$ from Equation (3) from $m_1 \cdots m_5$ and $k_1 \cdots k_5$
  Recall the correct Hash Point to release p_r (the Hash Point from the previous loop)
  if $i == n$ **then**
    $p_r = p_1^{P-1}$
  else
    $p_r = p_{i+1}^P$
  end if
  Send MT50 m_6 message with $s_1 \cdots s_5$ and p_r
end for
For continuous operation, compute and broadcast the next Hash Path End p_{n+1}^{P+1}, signed by a Level-2 ECDSA key and distributed via MT51, among the groups of sent messages $m_1 \cdots m_5$

TESLA uses only a single cryptographic primitive to generate a cryptographically-secure hash function. In this proposal, we select a salted SHA-256 that has been truncated to include the left-most 128-bits (described here as the "Hash Function" or $H(\cdot)$). Thus, we can describe the TESLA protocol concretely based on this selection without loss of generality. We truncate Hash Function output to the 128 most significant bits to generate 128-bit integers. Because the audience for this work includes primarily experts in navigation, we use the geometric terms "path" and "point" instead of "key chain" and "key", and describe

ALGORITHM 2
Receiver Procedures for Single Satellite-Single Frequency Authenticated Message Distribution with TESLA

while on **do**

Receive MT51 payload segments associated via the MT51 metadata and store into three hash tables H_1, H_2, and H_3. H_1 holds data for level-1 ECDSA keys. H_2 holds data for level-2 ECDSA keys and associated level-1 signatures on those keys. H_3 holds data for TESLA Hash Path Ends p^P_{n+1} and associated level-2 signatures on those Hash Path Ends. Each element in the Hash Tables store metadata such as the key expiration time and the relevant higher-level authenticating key. Await receipt of all needed unique MT51 OTAR Payload Segments, called the Authentication Stack, to assert authenticated Hash Path Ends. A Hash Path End element p^P_{n+1} stored in H_3 is authenticated if itself and signature are ECDSA verified by an authenticated element in H_2. A public ECDSA key element stored in H_2 is authenticated if itself and signature data are ECDSA verified by an authenticated element in H_1. A public ECDSA key element stored in H_1 is authenticated if it was prestored from the CA.

end while

while on **do**

Receive and cache unauthenticated non-MT50 messages $m_7 \cdots m_{11}$

if $m_7 \cdots m_{11}$ are MT51 **then**

Follow the procedures immediately above

end if

Receive and cache MT50 message m_{12} with signatures $s_7 \cdots s_{11}$ and Hash Point p_r

Note time of receipt of m_{12} as $t_r \leftarrow t_{12}$

Recall $m_1 \cdots m_5$ and recall m_6 with signatures $s_1 \cdots s_5$

$i \leftarrow 1$

$t_t \leftarrow t_r$

$p_t \leftarrow H(p_r, S^P, t_t)$ via Equation (1)

while $i \leq$ Maximum Iteration From Max Hash Path Length **do**

if $p_t \in H_3$ and $H_3(p_t)$ is authenticated and not expired via ECDSA through level-1 **then**

p_r is authenticated: check $m_1 \cdots m_5$ with p_r and $s_1 \cdots s_5$ via Equations (2) and (3) to authenticate $m_1 \cdots m_5$

Break while loop

else

$t_t \leftarrow t_t - 6$

$p_t \leftarrow H(p_t, S^P, t_t)$ via Equation (1)

end if

$i \leftarrow i + 1$

Note: caching previously authenticated p_t prevents excessive hashing to Hash Path End each iteration. There is no need to hash down to the Hash Path End more than once.

end while

Note: after this iteration $m_1 \cdots m_5$ and m_{12} are authenticated. $m_7 \cdots m_{11}$ are authenticated at the next iteration.

$m_1 \cdots m_5 \leftarrow m_7 \cdots m_{11}$

$m_6 \leftarrow m_{12}$ This procedure can be augmented to cached messages saved while awaiting a complete Authentication Stack.

end while

TESLA geometrically as a "one-way path". For this case, let each 128-bit integer be identified as a Hash Point, and let a collection of Hash Points that are consecutively related via the Hash Function be defined as a Hash Path. Non-consecutive Hash Points further along the Hash Path will relate via repeated application of the Hash Function. The Hash Function is secure and there are no known efficient algorithms that can compute the input Hash Point to generate the Hash Function

that yields a specific output Hash Point. In this manuscript, we refer to a specific input Hash Point as the "preimage" Hash Point of a specific output Hash Point. In other words, while it is trivially easy to compute the output Hash Point of the Hash Function given the preimage Hash Point, one will only be able to locate a preimage Hash Point after an exhaustive search. This is a one-way path. The domain of 128-bit integers, together with a randomized 128-bit salt inclusion to the Hash Function, will render pre-computation attacks (also known as Rainbow Table Attacks) infeasible with modern supercomputers because it meets 128-bit security. Our use of the term Hash Point also serves to avoid confusion with the overuse of the term "key" in TESLA and ECDSA applications. Authentication of private and public keys relates to ECDSA. Many keys will be derived from TESLA Hash Points to achieve the required authentication data-bandwidth efficiency.

To use a Hash Path to authenticate messages via TESLA, the Provider computes a Hash Path derived from a random starting Hash Point. The Hash Path must remain secret. The Provider broadcasts the final Hash Point along the Hash Path (i.e., the "Hash Path End") together with an ECDSA signature derived therefrom. Receivers recognize the Hash Path End as authenticated based on the ECDSA signature. The Provider uses the secret preimage Hash Point to the Hash Path End to derive hash-based message authentication code (HMAC) keys to send symmetric authentication signatures along with the standard message set. We propose using keyed-hash message authentication codes that use the Hash Function as its primitive for message signatures (i.e., the function, "HMAC" and the HMAC signatures themselves which are known as "HMACs"). We will continue to describe the protocol concretely using our selection without loss of generality. We truncate the HMACs to the left-most bits so that they will fit into SBAS messages. Providers and receivers agree on a schedule in which the Provider will (1) stop using the preimage Hash Point of the Hash Path End to authenticate messages, (2) broadcast that preimage Hash Point for receivers to authenticate messages, and (3) use the next preimage Hash Point along the Hash Path to authenticate new messages. Once a particular preimage Hash Point has been broadcast, receivers cannot accept new signatures derived therefrom. Because the HMACs were received when a specific preimage Hash Point was known only to Provider, it is understood that the Provider must have generated the messages. Each time the Provider releases a Hash Point, the Provider moves back one Hash Point along the secret Hash Path to derive a new HMAC. Given the security of the Hash Function, the Hash Point along the Hash Path located just before the released Hash Point remains secret, and thus becomes the new HMAC key for the next set of messages. The authentication security along the Hash Path hinges on (1) the security of the Hash Function and (2) the loose time-synchronization of the Provider and the receivers.

To complete authenticating security, TESLA must be used in tandem with an asymmetric authentication protocol. TESLA is secure along the Hash Path length. However, Hash Paths are finite and must be generated periodically. An asymmetric authentication protocol must sign the Hash Path End, which is the first Hash Point known to the receiver. In other words, for every Hash Path generated by the Provider, the Provider must use an asymmetric signature to ensure authentication security along the entire Hash Path. Moreover, the Provider and the receiver must be loosely time-synchronized. This poses a type of "Catch-22" problem because GNSS and SBAS Providers are the source of time information. This suggests that it might be helpful to avoid using TESLA in any form and to focus only an asymmetric protocol. Later in this work, we will show that the use of TESLA leads to superior loss-tolerance and requires less authenticating of pseudorandom data while accounting for the time-synchronization issues.

2 | DEFINITION OF THE SBAS AUTHENTICATION SCHEME

The SBAS authentication proposal proposes appending two MTs to the schedule identified here as MT50 and MT51. MT50 is used to authenticate the actual SBAS messages via the TESLA protocol. MT51 is used to (1) authenticate the Hash Path Ends via ECDSA as the Provider uses a series of Hash Paths in its standard operation, and (2) to provide OTAR and perform system-level maintenance of the cryptographic authentication scheme. The scheme in Figure 2 is a conceptual diagram of an overview and the relationships between MT50, MT51, and the cryptographic authentication method employed. In Section 2, we present precise definitions of our proposed SBAS authentication scheme. Sections 3 and 4 provide explanations and reasoning for our proposed definitions. While the definitions presented here are for use in SBAS L5 signals, given the spare bits remaining in each definition, this scheme can also be used for SBAS L1 signals by modifying the preambles (noted with reserved bits in definitions).

2.1 | ECDSA Key Structure

We propose a two-level ECDSA key structure, as suggested by Neish (2020).

Level-1 keys will be 256-bit-security ECDSA keys managed internationally by a trusted Certificate Authority (CA). Each level-1 key will be in use for 100 weeks. The CA will compute a large number of level-1 keys for use in the perpetual future and will encrypt each key individually via AES-128 with different AES encryption keys (one for each level-1 key) maintained as secret by the CA. The CA will distribute the AES-ciphertext to receiver manufacturers. Receivers will be preloaded with the collection of 512-bit public keys and encrypted with 128-bit keys via the AES-128. As level-1 keys expire, the CA will distribute the keys to decrypt the AES-ciphertext, one at a time, for the Provider to distribute via MT51. As the receiver receives the key to decrypt its onboard AES-ciphertext, it will update its current level-1 ECDSA public key. Each level-1 public key will be 512-bits. Any signature derived therefrom will be a 1024-bits.

Level-2 keys will be 128-bit-security ECDSA keys managed by the Provider. Each level-2 key will be in use for ten weeks. To create a new level-2 key, the Provider will generate a secure-random private ECDSA key and an associated public key. The Provider will then submit the new public key to the CA for a signature from the CA's current level-1 key. Provider will then distribute the new public key and the associated authenticating signature over the SBAS. The receiver will receive a new level-2 public key and the authenticating signature, verifying the received new level-2 public key with the associated decrypted level-1 public key.

The Provider will use the level-2 keys to authenticate the TESLA Hash Path Ends. Keys derived from the TESLA Hash Paths will be used to authenticate the bulk of SBAS messages with HMACs. For all levels, the authenticating pseudorandom data delivered will accompany data (e.g., SBAS message preamble, MTs, and other data) that must be sent as per the definitions described below. A specific signature must be derived from the entire SBAS message used to deliver that particular key. Concretely, when a level-1 key authenticates a level-2 key, the level-1 signature must be derived from the entire set of messages used to deliver the level-2 key and the expiration time of the accompanying key. In other words, the level-1 signature must be derived from the complete messages containing overhead data, not just from the level-2 key itself. If this does not take place, then the accompanying data,

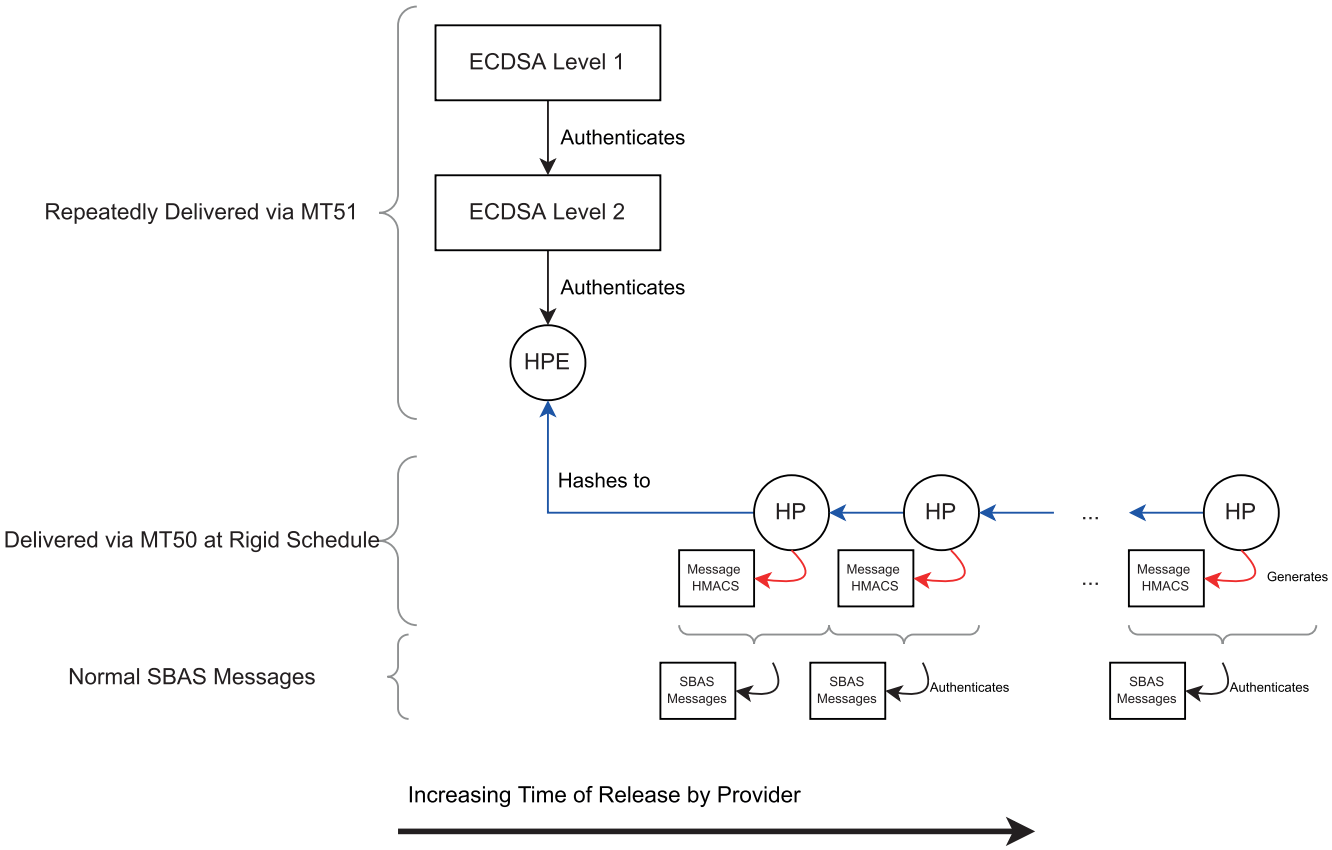

FIGURE 2 Conceptual diagram depicting an overview of the entire scheme presented in this work

The objects depicted are defined and described in the Sections to follow. Multiple levels of ECDSA authenticate a Hash Path End (indicated as HPE in the diagram). Preimage Hash Points (HP in the diagram) together with HMACs are used to authenticate SBAS messages. Black arrows represent the direction of authentication; blue arrows, hashing operation, and red arrows, HMAC operations. The diagram reads from left-to-right with increasing time of release by the Provider, i.e., items to the right are released later by the Provider than items to the left.

most notably the key expiration times, will not be secured by the cryptographic primitives.

2.2 | TESLA Hash Path and HMAC Keys

Each TESLA Hash Path will be used over one week. The Provider will generate an entire Hash Path before its actual use and then broadcast the Hash Path End signed by the current level-2 ECDSA key via MT51. Each Hash Point, except the Hash Path End, will be associated with at least five HMAC keys that will be used to authenticate at least five messages with HMAC, depending on the number of iterations of Equation (2). Therefore, a Hash Path will include 100,801 Hash Points, one for each sixth second for the week, and one for the Hash Path End.

To generate a Hash Path P, the Provider will derive a secure random 128-bit salt S^P from level-2 ECDSA authentication, as described in Section 3.2.1. Let the Hash Points of P be denoted p_i^P, and let t_i be the time at which the Provider publicly releases p_i^P via broadcast. Here, t_i is an integer time (e.g., time in seconds since the GPS epoch). We propose Equation (1) to define the Hash Path where $||$ denotes bit concatenation and $\oslash$ denotes integer division.

$$p_{i+1}^{P} = H\left(p_i^P \,||\, S^P \,||\, (t_i \oslash 6)\right) \tag{1}$$

The purpose of the integer division is explained in Section 3.2.2. We propose a left-most 16-bit truncated signature from HMAC that authenticates each SBAS message delivered via MT50. Each message m_j, sent at t_j, will be provided with a unique HMAC key k_j. The key k_j for each message m_j will be generated according to Equation (2). The signature s_j derived therefrom will be according to Equation (3). t_j is the integer time that the authenticated message will be broadcast and received. PRN is the pseudorandom code associated with the broadcasting geostationary satellite. Frequency is the frequency band of the particular transmission (e.g., a string containing L1 or L5). We discuss the necessity of the concatenation and HMAC operations of Equation (2) in Sections 3.2 and 3.3. Equation (2) ensures that the HMAC for each message from each satellite has its own key.

$$k_j = \text{HMAC}(p_i^P, t_j \,||\, \text{PRN} \,||\, \text{Frequency}) \tag{2}$$

$$s_j = \text{HMAC}(k_j, m_j) \tag{3}$$

The output of Equation (3) is truncated to its 16 most-significant bits. We note that because Equation (1) uses a concatenation operation because the input data is less than 512-bits, the block size for the selected Hash Function. Equation (1) would need to be modified if the input data were to be larger than 512-bits; this will be critical to mitigate length extension attacks (Boneh & Shoup, 2017).

Section 2.5 and Algorithms 1 and 2 describe how the Provider and receivers should perfrom authentication, queueing, and caching to verify messages.

2.3 | Message Type (MT) 50

The findings shown in Table 1 present our proposed definition of MT50 with bit allocations. The Provider will send an MT50 with every six messages. The delayed key release used to authenticate messages means that each message will be authenticated between 7 and 11 seconds after its broadcast. Provider should set the cadence of the integrity messages so that each immediately precedes the scheduled MT50s to minimize the time needed for their authentication. If the receiver cannot authenticate a message because of a lost MT50, it generally disregards the message. (Information that alerts the receivers to decrease the level of trust need not be disregarded). Within the message definition, there are five 16-bit HMACs and one 128-bit Hash Point. Once the receiver receives an MT50, the five HMACs included correspond to the previous messages with a secret Hash Point known only to the Provider at the message sending time. The 128-bit Hash Point included corresponds to the HMACS included in the previous MT50 message sent six seconds earlier. Figure 3 provides a conceptual diagram of the delayed Hash Point key release.

TABLE 1
Bit Allocation for the Proposed MT50

Preamble	MT	Reserved	HMAC1	HMAC2	HMAC3	HMAC4	HMAC5	Hash Point	Spare	CRC
4	6	4	16	16	16	16	16	128	4	24

Note: As per SBAS definitions there are 250 bits per message.

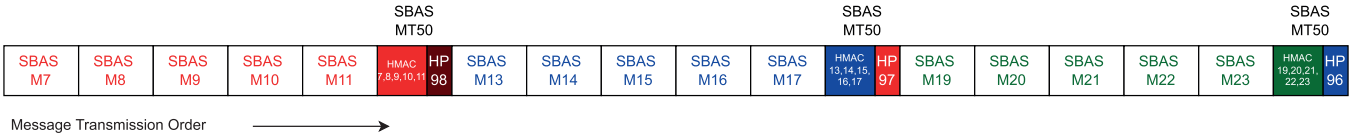

FIGURE 3 A conceptual diagram of how consecutive MT50 messages relate to each other
The colors correspond to a specific Hash Point along the Hash Path. Each MT50 includes the HMACs of the five previous messages and the Hash Point used for the HMACs sent with six earlier messages.

T	148	149	150	151	152	153	154	155	156	157	158	159	160	161	162
T mod 6	4	5	0	1	2	3	4	5	0	1	2	3	4	5	0
T ⊘ 6	24	24	25	25	25	25	25	25	26	26	26	26	26	26	27
Nominal	M	I	50	M	M	M	M	I	50	M	M	M	M	I	50
Alert 1	M	I	50	I	I	I	I	I	50	M	M	M	M	I	50
Alert 2	M	I	50	M	I	I	I	I	50	M	M	M	M	I	50
Alert 3	M	I	50	M	M	I	I	I	I	50	M	M	M	I	50
Alert 4	M	I	50	M	M	M	I	I	I	I	50	M	M	I	50
Alert 5	M	I	50	M	M	M	M	I	I	I	I	50	M	I	50
Alert 6	M	I	50	M	M	M	M	I	I	I	I	I	50	I	50

FIGURE 4 Conceptual diagram of accommodations made by the counter scheme to perturbations in the MT50 schedule
In the diagram, "m" denotes a standard message, and "I" represents an integrity message. Integrity messages that would be sent on a nominal schedule are marked blue and additional integrity messages during an alert are marked in yellow. Note that $T \oslash 6$ does not change in the event of an alert message MT50 delay, as shown in red; thus, the Hash Path is preserved.

As per the SBAS specifications, in the event of a GNSS integrity alert, an alert message must be sent by the Provider with four messages in a row. Therefore, occasionally an alert message will take priority over an MT50. The salted Hash Function described in Equation (1) accommodates small perturbations to the schedule resulting from alert messages as described in Section 3.2.2 and Figure 4. Even with an MT50 delay, each MT50 must sign the messages that it would have signed without an alert as described in Section 3.2.2.

2.4 | Message Type (MT) 51

In this work, we provide two MT51 definitions with a 128-bit OTAR Payload Segment and an 84-bit metadata section. This version includes many features that could be relevant to SBAS stakeholders. We have not specified which features should be incorporated. This will be deferred until all SBAS stakeholders have had their considerations heard. The information in Section 4.1 discusses how to modify the 128-84-bit allocation if SBAS stakeholders would prefer not to use the additional features described later in the text. However, given the academic context of this work, and because our design meets the key performance indicators (KPIs),

we choose to publish the 128-bit feature-rich design. Tables 2 and 3 provide our proposed definition of MT51 with bit allocations. The Provider must broadcast MT51 messages for approximately 1 in every 18 messages for MT51 as described in Section 4.4. These messages do not need to be sent on a rigid schedule; they can be sent in the extra space within the current SBAS schedule.

The payload section for MT51 is only 128-bits which leaves 84-bits for metadata that describes how a receiver should interpret the 128-bit payload. The larger 84-bits allows users to introduce additional features, such as parallel and redundant key management as well as several other features explained later in the

TABLE 2
Bit Allocation Proposed for MT51 at 250 bits per Message

MT51 Bit Allocation					
Preamble	**MT**	**Reserved**	**Payload Metadata**	**OTAR Payload Segment**	**CRC**
4	6	4	84	128	24

Note: Table 3 describes the metadata which specifies how a receiver should interpret the payload. The Authentication Stack, defined in Section 2.5, is composed of a total 2048 bits and required the receipt of the 16 unique messages to OTAR.

TABLE 3
Bit Allocation of Payload Metadata

MT51 Payload Metadata		
Section	**Bits**	**Value**
Germane Service Provider ID	5	WAAS, EGNOS, MSBAS, GAGAN, et al.
Germane Key Level	2	0 - Spare 1 - ECDSA AES key to decrypt Level 1 ECDSA Public Key 2 - ECDSA Level 2 Public Key 3 - TESLA Hash Path End Hash Point
Germane Key Hash	16	Truncated Unsalted 16-bit Hash of Entire Germane Key
Germane Key Expiration	32	Absolute GPS time (i.e., seconds since GPS epoch) of Germane Key Expiration
Authenticating Key Hash	16	Truncated Unsalted 16-bit Hash of Entire Authenticating Key
Payload Type	2	0 - Public Key, AES Decryption Key, or Hash Path End 1 - Authenticating pseudorandom data derived from the Authenticating Key 2 - Spare 3 - Core Constellation Broadcast Ephemerides Authentication
Payload Segment Number	4	The ordered segment number of Germane authenticating pseudorandom data
Parity Bit	1	Parity bit for a compressed public key
Spare	6	Additional features possible discussed in Section 4.1
Sum Total	84	

Note: To distinguish the key updated with a specific MT51 and the key used to authenticate that MT51, we call the key associated with the MT51-delivered payload the Germane Key and the key used to authenticate that delivered key the Authenticating Key.

TABLE 4
List of All Unique MT51s in an example Authentication Stack as defined in Section 2.5

Unique MT51 Number	OTAR Payload Segment Content
1	AES-128 Key to decrypt receiver-stored Level-1 ECDSA Public Key
2	ECDSA Level-2 Public Key, Segment 1
3	ECDSA Level-2 Public Key, Segment 2
4	ECDSA Level-1 Signature of ECDSA Level-2 Public Key, Segment 1
5	ECDSA Level-1 Signature of ECDSA Level-2 Public Key, Segment 2
⋮	⋮
11	ECDSA Level-1 Signature of ECDSA Level-2 Public Key, Segment 8
12	TESLA Hash Path End
13	ECDSA Level-2 Signature of TESLA Hash Path End, Segment 1
14	ECDSA Level-2 Signature of TESLA Hash Path End, Segment 2
15	ECDSA Level-2 Signature of TESLA Hash Path End, Segment 3
16	ECDSA Level-2 Signature of TESLA Hash Path End, Segment 4

Note: This set is broadcast repeatedly by the Provider for cold-start receivers. A receiver must receive all of the unique MT51s to initiate authentication.

text. Table 4 provides an sample set of unique MT51 messages, each containing a 128-bit segment per message, that form an Authentication Stack (defined in Section 2.5). To distinguish the key updated with a specific MT51 and the key used to authenticate that MT51, we call the key associated with the MT51-delivered payload the Germane Key and the key used to authenticate that delivered payload the Authenticating Key. The metadata specifies the following, matching the order shown in Table 3. (1) it will identify the system to which the Germane Key applies; (2) it will specify whether the payload is an ECDSA public key, an AES decryption key for an ECDSA public key, or a TESLA Hash Path End; (3) it will provide a 16-bit hash of the entire Germane Key so that the receiver can immediately associate the OTAR Payload Segment with a specific key; (4) it will designate the expiration time of the Germane Key; (5) it will provide a 16-bit hash of the entire authenticating key so that the receiver can immediately associate the authenticating pseudorandom data OTAR Payload Segment with a specific authenticating key; (6) it will specify whether the payload itself is a key or authentication signature; (7) it will identify the segment number of the authenticating pseudorandom data so that the receiver can aggregate the authenticating pseudorandom data segments over time.

The signatures used to authenticate a particular key must be derived from the entire set of full MT51 messages used to deliver them. The metadata associated with authenticating pseudorandom data includes the expiration time. These keys are only secure for specific lengths of time, as described in Section 2.1 and 2.2; hence, the expiration time must be authenticated together with the corresponding authenticating pseudorandom data so that each key is retired securely. After the expiration of a particular key, receivers must reject all messages signed with the expired key.

2.5 | Procedures

Algorithm 1 describes the procedure needed by the Providers to use TESLA securely. In addition, the Providers must assemble OTAR data by coordinating with the CA as described in Section 2.1. Algorithm 2 captures how a receiver

should operate beginning with a cold start and includes specifications of some of the onboard data structures that should be used to track key maintenance. The term "cold start" is used to describe use of a receiver that has been off for an extended period with onboard Level-2 ECDSA or TESLA information that has expired according to its onboard clock. Upon cold start, a receiver must track and record incoming SBAS messages. A receiver should not use any unauthenticated data. NB: the complete set of MT51s is self-authenticating. Therefore, it must track the incoming MT51 messages until it has received a complete set of unexpired Level-1 ECDSA public key, Level-2 ECDSA public key, TESLA Hash Path End, and the associated signatures (collectively known as the "Authentication Stack"). The receiver must track and store MT51 messages until the aggregate OTAR Payloads of authenticating pseudorandom data provides a successful ECDSA authentication of the entire Authentication Stack, including a level-1 key onto a level-2 key and a level-2 key onto the Hash Path End. Until it has received and verified a complete set of unique MT51s included in the Authentication Stack, the receiver cannot assert an authenticated fix and must ignore the SBAS corrections and integrity data (i.e., non-MT51 messages) derived from incoming messages. Once the Authentication Stack is received and verified by ECDSA, the receiver can process and authenticate MT50 messages via TESLA and can also associate the MT50-delivered HMACs with messages to authenticate and process the SBAS correction and integrity data. The Provider repeatedly broadcasts the current Authentication Stack, as discussed in Section 4.4 to accommodate random receiver startups.

Algorithms 1 and 2 delineate only the single satellite and single frequency cases. To augment those algorithms to handle the multiple satellite or multiple frequencies, five messages should be signed and verified for each satellite and each frequency, as described by Equations (2) and (3). This means that the MT51 Authentication Stack is used for all satellites and frequencies, but each message for each satellite and frequency is provided with its own authentication via TESLA. By reusing he same Authentication Stack for all satellites and frequencies, we exploit the scalability of TESLA allowing for smaller TFAFs (see Section 3.2).

Upon full receipt of the Authentication Stack, receivers must hash the Hash Point from the most recent MT50 to the MT51-provided Hash Path End. In the special case in which the receiver has only been off a short time, turned offline, and then turned back online during the use of the same Hash Path, the receiver TFAF is the time required to hash to the Hash Path End. The worst-case number of hash computations is the length of the Hash Path (about 100,000) and will occur when a receiver first turns on at or near the expiration of the Hash Path. With standard commodity hardware (e.g., an Intel Core i5 Processor), a worst-case time of approximately 10 seconds will be observed if a receiver is turned on immediately before a Hash Path expires. We measured this time by experimenting with personal laptops that were not specifically built for this process. With hardware acceleration, this process time could decrease and evaluated in parallel with other standard receiver processes. This initial hashing computation only occurs when the receiver is turned on and should not hinder processing SBAS navigation data in real-time after an authenticated fix.

2.5.1 | *Modification for Metadata Removal*

In Section 4.1, we discuss whether certain pieces of the metadata are strictly necessary and whether certain features of the design of this work are useful to all SBAS stakeholders. If the metadata are stripped from MT51 in a final design, then

Section 2.5 must be modified. For instance, a minimum-metadata MT51 design could only contain the page number of aggregate metadata.

In Algorithm 2, the receiver stores keys in hash tables because the OTAR is not rigidly managed. Different keys can authenticate other keys. For MT51, the receiver must check that the TESLA Hash Path End is signed correctly by the metadata-specified level-2 key and so on from level-2 to level-1. For another MT51 design with only the page number as the metadata, either the unique OTAR Payload Segments do or do not aggregate to generate a consistent Authentication Stack. Rather than store keys in hash tables, the receiver will simply aggregate the OTAR Payload Segments for a complete Authentication Stack in an order specified by the Provider, such as TESLA Hash Path End || ECDSA Level-2 Authentication of TESLA Hash Path End || ECDSA Level-2 Key || ECDSA. Once the entire Authentication Stack aggregate authenticates via ECDSA, the receiver achieves its TFAF and can begin authenticating the bulk of SBAS messages via MT50.

3 | TESLA DESIGN METHODOLOGY

3.1 | TESLA Loss Tolerance

TESLA allows receivers to derive missed Hash Points from Hash Points released later along the Provider's pre-computed Hash Path that has been kept secret. For example, suppose a receiver misses an MT50 message and therefore does not receive a Hash Point. The receiver can derive that missing Hash Point by computing the Hash of the next released Hash Point. In another example, suppose a receiver misses several day's worth of Hash Points; upon receipt of a new Hash Point, the receiver can hash all the way down to the Hash Path End as described in Equation (1) to document the new messages' immediate authenticity. This property scales along the length of the Hash Path. If a receiver is off longer than the Hash Path's length or applicability, it will need to OTAR the Hash Path End via MT51 and ECDSA.

Regarding loss tolerance of messages and the security of 16-bit HMACs, we implement a main idea previously described by Neish (2020). The smaller 16-bit HMAC design of MT50 aids in general tolerance of message loss. Suppose, instead, each MT50 contained a single HMAC that authenticated the previous five messages as a group. If any of the five earlier messages were lost, the receiver would not verify any of the messages in this group. Therefore, to accommodate message loss, we specify that each HMAC from the set of five smaller HMACs individually authenticates each of the five previous messages. Because the 16-bit length of the HMACs is unusually small with respect to cryptographic authentication, we take special care to specify our spoofing detection procedures and ensure cryptographic independence of keys, as described in Section 3.3.

3.2 | TESLA Efficiency

The number of authenticating pseudorandom data bits required to perform OTAR of a Hash Path does not increase with Hash Path length. While complete OTAR requires a daunting 2048 bits, as described in Table 5, the bits do not scale with Hash Path length because the Provider sends only a Hash Path End authenticated with ECDSA. If the Provider finds that the overhead required to perform OTAR of a Hash Path has become too burdensome on the schedule, the Provider

TABLE 5
Delineation of the Number of MT51s Required to Complete a Single OTAR for a Specific Key

Key Level	Security Level	Public Key Length	OTAR Bit Requirement	MT51s Req.
1	256	512	128	1
2	128	256	1280	10
TESLA	128	128	640	5

Note: Level-1 keys require only the 128-bit AES decryption key, hence a single MT51. Level-2 keys require 256-bit key, and 1024 bits of authenticating pseudorandom data (twice the Level-1 public key length), hence, 1280 bits and 10 MT51s. TESLA Hash Path Ends require 128-bit Hash Point and 512 bits of authenticating pseudorandom data (twice the length of the level-2 public key), hence 640 bits and 5 MT51s. Each unique MT51 is required for OTAR.

can increase the Hash Path's length and decrease the OTAR transmission frequency. Increasing the Hash Path length requires the Provider to compute a longer Hash Path for each OTAR and and for receivers to hash more in the event of cold start. However, this burden is negligible because commodity hardware can compute Hash Paths on the lengths specified in this work within seconds, as described in Section 2.5. In any case, MT51 can reassign Hash Points mid-way through a Hash Path to decrease this burden. For instance, while a Provider could generate week-long Hash Paths, it could assign Hash Path Ends each day or each hour to alleviate the burden of the initial hashing operation. Given the 128-bit Hash Point length, a Hash Path length on the order of decades is safe from attack (Neish, 2020). Therefore, the OTAR transmission frequency is primarily driven by the desired TFAF, as discussed in Section 4.4.

To meet a standard 128-bit security level for TESLA, we must use cryptographically-independent 128-bit long keys for each HMAC. It is generally desirable to minimize the number of bits required for TESLA Hash Path distribution. We achieve this by deriving all cryptographic keys from the same 128-bit Hash Path. Specifically, each iteration of Equation (2), from each combination of time, satellite, and frequency, is derived from the same 128-bit Hash Point distribution. Thus, we do not need to generate a separate Hash Path for each stream of authenticated information. Provided each of the keys derived is cryptographically independent, the outcome is cryptographically secure. Ensuring cryptographically-independent keys is achieved by the intermediate HMAC operation shown by Equation (2) as described in Section 3.3. This allows a single Hash Point to authenticate multiple messages and a single Hash Path for a set of SBAS satellites.

While the Providers could maintain separate Hash Paths for each satellite, using a single Hash Path may decrease the time required to receive the Authentication Stack by cold start receivers. For instance, WAAS uses three geostationary satellites. WAAS could use the same Hash Path for all of its satellites and broadcast the set of unique MT51 messages that make up the Authentication Stack out-of-phase, thereby decreasing the TFAF by 66.6%, while accommodating receivers that do not track each satellite. We can extend this argument for the L1 and L5 frequency bands by decreasing the TFAF by 83.3%. Our selection of a TFAF for a receiver tracking by a single geostationary SBAS satellite on a single frequency is discussed in Section 4.4.

3.2.1 | *ECDSA-derived Hash Path Salt*

The security of each TESLA Hash Path hinges on the difficulties involved in computing any earlier preimage Hash Point before it is released by the Provider.

Unsalted Hash Paths are susceptible to pre-computation attacks, also known as Rainbow Table Attacks (Boneh & Shoup, 2017). To perform these attacks, an attacker pre-computes a large number of Hash Paths and stores them in hopes that one of the pre-computed Hash Paths contains a currently secure mid Hash Path Hash Point. If this were to occur, the attacker then has saved Hash Points located earlier on the no-longer-secure Hash Path and can thus spoof SBAS-authenticated messages. To prevent this from occurring, we must introduce random variations (known as "salt") to the Hash Function which will render pre-computation attacks unfeasible.

A well-designed salt scheme will have the following characteristics: (1) the salt scheme must be sufficiently strong to deter pre-computation attacks; (2) the scheme must accommodate spontaneous and episodic message loss (e.g., receiver interference or receiver offline cold start); and (3) he scheme must not impose a burden on the message scheduler. A 128-bit salt would suffice for the security requirements. Given the modular design of the proposed MT51, one could append a designation for the salt of a particular Hash Path to the metadata definitions shown in Table 3 This would require an additional message sent by OTAR to a Hash Path. This would be unlikely to pose a burden on the SBAS schedule. However, we propose an alternative that saves an MT51 message by basing the Hash Path salt on the level-2 signature protocol. We suggest that the Provider might compute a Hash Path Salt S^P for Equation (1) via Algorithm 3.

In this case, the Provider computes a cryptographically-secure nonce for every ECDSA signature. Using Algorithm 3, the salt derives a public quantity derived from a nonce. For ECDSA, this is the curve point C. The analogous number in EC-Schnorr signatures is usually called r. This C is cryptographically-secure and random because it is derived from the cryptographically-secure nonce generated at signing. Use of this entity as the Hash Path salt saves an MT51 message without compromising the signing key or the Hash Path.

While this scheme provides the advantage of one less message for OTAR, it impedes the scheme's flexibility. Using a nonce more than once reveals will reveal the secret private key used to authenticate the data. This means that the Hash Path and its authentication signatures are immutable. The Provider cannot remove a Hash Point mid Hash Path to save receivers the computation of hashing down to the signed Hash Path End because the salt derived from it would then be different. Providers also cannot change any of the metadata (e.g., the expiration time). To reincorporate these two features, as mentioned above, the Provider could augment the MT51 metadata and distribute the salt as a separate MT51.

ALGORITHM 3
Transmitting Salt S^P without additional message with ECDSA

Provider
Provider generates cryptographically-secure nonce K
With elliptic curve base point G, Provider computes elliptic curve point $C = K \times G$
$S^P = H(C)$
Provider computes hash path $p_2^P \ldots p_{n+1}^P$ with Equation (1)
Provider generates ECDSA signature for p_{n+1}^P with C and broadcasts before actual use of Hash Path
Receiver
Receiver receives p_{n+1}^P with ECDSA signature for authentication
Receiver derives C from ECDSA signature
$S^P = H(C)$
Receiver authenticates new message on a new Hash Path upon receipt of p_n^P.

3.2.2 | TESLA Hash Path Counter from Time

In the standard TESLA formulation, the Hash Function uses integer counter denoting the number of Hash Path Hash Points from the Hash Path start. We propose an analogous approach involving the integer time of message dispatch and arrival. Dispatch and arrival times, rounded down to the nearest integer, are the same for the Provider and the receiver. This is because messages are sent each second and transmission time-of-flight is less than one second. There are several advantages to using the time, instead of the integer number of points from the Hash Path start. Upon start, a receiver is capable of verifying a Hash Point since Equation (1) is a function of current integer time and the ECDSA authenticating pseudorandom data. Moreover, it allows Hash Path switching to occur on a non-rigid schedule, which also helps to maintain security (Caparra et al., 2016). While we specified a one week interval, and it would be natural to have a new Hash Path begin at the beginning of a GNSS week, the Provider need not communicate the start and length of Hash Paths as overhead or metadata for a non-rigid schedule because of the hashing properties of the Hash Point. If the Provider were to switch Hash Paths arbitrarily, there is a constant-time complexity for the receiver which will need to check if a new Hash Point hashes to the current Hash Path or a new one, as shown in Algorithm 2.

Time-based counter aids with security can be used given the loose-time synchronization assumption. The TESLA protocol assumes that Provider and receiver are loosely time-synchronized. To break the TESLA protocol security, an attacker must hack the receiver time to six seconds behind the Provider time. In Equations (1) and (2), we proposed including the times t_i and t_j in the TESLA counter and the Hash-Point-to-HMAC-key derivations. As proposed, if the Provider and the receiver are not time-synchronized within one second, the authentication scheme fails to certify messages as authenticated because all the keys were derived from the integer time in seconds. Thus, SBAS message spoofing becomes more complex since any SBAS spoofer must also spoof the GNSS time.

During standard operation, the Provider will send an MT50 every six seconds. However, the SBAS alert requirements specified that, upon an alert, alert messages must be sent immediately for four consecutive seconds. Since the Provider computes the entire Hash Path before its use, including assuming the t_i's associated with each Hash Point, alerts will interfere with the six-second MT50 schedule and the TESLA counter that we have proposed. To accommodate perturbations of the authentication schedule, we propose (1) nominally sending each MT50s when $t_i \bmod 6 = 0$ and (2) performing an integer division by six on time t_i as shown in Equation (1). If an alert occurs leading to four consecutive messages that displace a TESLA authentication between one and four seconds after $t_i \bmod 6 = 0$, the Hash Path is preserved because $t_i \oslash 6 = (t_i + 4) \oslash 6$ when $t_i \bmod 6 \leq 4$. Figure 4 provides a conceptual diagram of how the TESLA counter is preserved in the event of an alert message.

In the event of an alert, we must modify the scheme to maintain the security of the time-synchronization schedule. Nominally, the earliest that a receiver will authenticate a message is six seconds after its transmission. This serves to protect the scheme and delay an attack for up to six seconds. That length of time is the minimum spread between a delivered HMAC and the corresponding Hash Point used to generate it. As shown in Figure 4, this six-second minimum is violated unless one of the following two proposed modifications is implemented. Option 1: In the event of an alert, the Hash Point after the normal Hash Point authenticates messages during an alert. Consistent with the findings presented in Figure 4, the

MT50 in column 168 (not shown) would authenticate the messages of Alerts 3 through 6. Option 2: The receiver does not accept the delayed MT50 until it has been authenticated by a 16-bit HMAC as any other message. Consistent with with Figure 4, the receiver does not use the delayed MT50s shown incolumns 157 though 160 until it receives a valid HMAC from the MT50 in column 162 and the Hash Point delivered in column 168 (not shown). Both Option 1 or Option 2 provides the same level of security by returning the minimum HMAC-to-Hash-Point delay back to six seconds. Based on our implementation work with MAAST, we claim that Option 2 may be easier to implement with current receivers and software. To limit the information lost, a delayed MT50 should sign the original messages corresponding to the loosely-synchronized schedule. In Figure 4 in the row labeled Alert 6: (1) the MT50 of column 160 must sign the messages of columns 151 through 155; (2) the MT50 of column 162 must sign the messages of columns 157 through 161; and (3) the non-MT50 message of column 156 will remain unauthenticated. For the rows labeled alerts 3 through 6, the integrity messages of column 156 will also remain unauthenticated. This is acceptable because those messages tell receivers not to use the service; the surrounding integrity messages will be authenticated regardless. Without this requirement, messages of substance will not be authenticated.

3.3 | TESLA Security

Previous work identified the appropriate security-level lengths under conservative adversary models (Neish, 2020). These findings assert that a TESLA Hash Point length of 115-bits and an HMAC length of 15-bits will be sufficient to deter a supercomputer-level attack over the time-between-authentication interval and assert a sufficiently low probability of success. In this work, we have rounded up these numbers to the nearest base-2 number, at 128 and 16, respectively. Increasing these lengths adds additional security-level protection. The primary reason for 128 Hash Point lengths is that, as specified, MT51 can accommodate 128 bits. Section 4 discusses why 128 bits was selected to aid in the scheme's maintenance. The selection of 16-bit HMAC lengths follows our general strategy of 2-bit lengths and aids in the flexibility of the scheme, as discussed in Section 4.1.

Until the Hash Point is released, the HMACs are indistinguishable from random bits. Given the security of the salted Hash Path, the probability that an adversary could generate a preimage Hash Point is 2^{-128}. This probability is sufficiently low so that it should not be expected to occur ever even with the support of vast computational resources. As described in Section 3.2, given the desire to use the 128-bit Hash Path efficiently at a particular Hash Point time interval, we desire to authenticate many different pieces of information simultaneously (e.g., five messages per Hash Point, Section 4.2). Thus, we have taken a conservative approach in our construction to mitigate any anticipated vulnerabilities when implementing potential feature extensions in this SBAS TESLA design or any other GNSS-TESLA constellation concepts (Anderson et al., 2022; O'Hanlon et al., 2022). To discourage implementation errors when applying these design concepts, we ensure that each piece of information authenticated with an HMAC has its own cryptographically-independent HMAC key as described in Equation (2).

Equation (2) also takes the secure Hash Point and uses HMAC to derive cryptographically-independent keys by applying HMAC with the Hash Point as the HMAC key field together with unique contextual information in the HMAC message field. In the specific case of Equation (2), the HMAC message field includes the

message time, the satellite PRN code, and the frequency. The time parameter allows a single Hash Point to authenticate the five messages individually because each message is sent at a different time. The PRN code allows all of the satellites to share the same Hash Path because each satellite has its own unique PRN code; the same is the case with the frequency. Our time, PRN code, and frequency band choices are arbitrary, except that our selection guarantees uniqueness; each message from each satellite and from each band is provided with a cryptographically-independent key that can be used to derive the HMAC. Any unique identifying information, such as a counter, would suffice to ensure that the output of Equation (2) is cryptographically independent.

There are several ways to construct a secure scheme without the intermediate HMAC operation of Equation (2), for example, prepending the context to the signed data. Careful consideration must be take to avoid implementation errors that might introduce vulnerabilities including, but not limited to, (1) allowing an adversary to spoof a message from one satellite so that it would appear to be coming from another satellite (or another frequency); (2) allowing an adversary to spoof a message from another time (given that we sign five messages at a time using this scheme); (3) allowing an adversary to prefix the context of one message to another to engage in a swap and provide context for confusion attacks (e.g., especially when the signed data are not of fixed length); (4) allowing an adversary to engage in related-key attacks (Peyrin et al., 2012); or (5) deriving additional data using this key (e.g., in signal watermarking concepts as described in Anderson et al. (2022); O'Hanlon et al. (2022)) performed in a reversible manner, allowing an adversary to have access to an unreleased TESLA Hash Point. Our construction with Equation (2) separates context and content to mitigate these issues and provides a clear way to extend authentication by generating additional and irreversible cryptographically-independent information (Section 4.2 (Anderson et al., 2022; O'Hanlon et al., 2022)) in an implementation-error-proof manner.

Because the Hash Point is not known to an adversary when the HMACs are released, the probability that an adversary can forge a 16-bit HMAC is 2^{-16}. To provide adequate protection against forgery, we must specify a conservative approach for forgery detection. For example, if any of the smaller HMACs fails the verification algorithm, the receiver must discard all non-MT51 information from that particular SBAS satellite and restart collecting new SBAS data. While 2^{-16} is a relatively high probability, this is sufficient for the SBAS context because (1) an adversary does not yet have access to the delay-released key nor an HMAC verification oracle (from the cryptography security context), and (2) once forgery has been detected, all prior SBAS data will be immediately discarded. While the details will be presented in a forthcoming work, the receiver logic will be set up so that a single message forgery event will not result in an integrity failure. this will decrease the likelihood of harmful forgery to $2^{-32} < 10^{-9}$. The likelihood that an adversary could forge so many messages successfully is small enought to meet the security level required by the stakeholders.

Our selection of SHA-256 for the Hash Path and HMAC generation is not necessarily required. We select SHA-256 because it is standard and widely-used. However, the Providers may wish to consider other standardized hashing functions such as SHA-384 or those from SHA3 group to address other concerns, for example, future-proofing and hardware concerns. We selected HMAC-SHA-256, which includes SHA-256 as its primitive, to simplify the protocol. The widespread and straightforward use of SHA-256 aids in the continued security of the proposed scheme. If the SHA-256 security is broken, it will be widely-publicized and quickly exchanged for another hash function. Providers would need to replace only a single

function in their implementation to continue operation, much like the recent SHA-1 deprecation.

4 | OVER-THE-AIR REKEYING (OTAR) DESIGN METHODOLOGY

The proposed design of MT51 is fundamentally modular. The design of MT51 serves to deliver pseudorandom data so that its use can be flexible and recycled among different OTAR designs and applications that require delivery of pseudorandom data, including keys, signatures, and Hash Points. Its purpose is to deliver large chunks of pseudorandom data to maintain the SBAS authentication scheme and the associated metadata that addresses the way in which the receiver should interpret that authenticating pseudorandom data. The same message definition delivers TESLA Hash Path Ends, level-2 keys, level-1-key decryption keys, and the authenticating pseudorandom data required for authentication.

Our choice to allow 128-bits per MT51 serves several purposes. We selected 128 and 256-bit security level ECDSA keys since there exist standardized, secure elliptic curves at these security levels with each divisible by 128. The public keys and derived signatures are two- and four-times the security level length, respectively. These quantities are divisible by 128, meaning that there will be integer numbers of OTAR Payload Segments without wasted zero padding. Table 5 exhibits the number of messages required to perform OTAR at each of the key levels. We recognize that we could use the 192-bit level without modifying our scheme because all of the 192-bit level data are also divisible by 128. The design is agnostic to the asymmetric scheme and is recycled for OTAR of the Hash Path Ends, thereby expanding its use for maintenance of the entire scheme, not just the asymmetric portion. The proposed MT51 standard would require no changes if a more efficient authentication scheme (e.g., EC-Schnorr), a quantum-secure scheme, or a different security level replaced the proposed asymmetric authentication scheme. The modular design also facilitates easy expansion of additional features described in Sections 4.1 and 4.2 that may desirable to several of the SBAS stakeholders.

MT50 and MT51 are agnostic to the asymmetric cryptographic security scheme and hash function primitives. The SBAS MT scheme need not change When the security of a primitive becomes compromised. However, the Providers and receivers will need to change the primitives that are used in modular fashion. Since MT51 provides 128-bits of authenticating pseudorandom data per message, and standardized cryptographic primitive lengths are generally integer factors of 128, changes to the scheme will by definition increase or decrease the number of segments required to transmit information. If the security of the 128-bit truncated SHA-256 is compromised, SBAS could double the Hash Point length space without affecting the MT50 frequency. Suppose the Hash Point space is the set of 256-bit integers, analogous to untruncated SHA-256. Each MT50 will transmit half of a Hash Point, and each HMAC key will be derived from two consecutive MT50 messages. This increases the time-between authentication events by a few seconds; however, it doubles the Hash Path security and maintains its loss-tolerant properties.

4.1 | MT51 Metadata Design

We specified the inclusion of a Germane and Authenticating Key hash within the authenticating pseudorandom data metadata. Since the payload is pseudorandom,

metadata must exist so that the receiver can associate the unique MT51 payloads. Any identifying feature would suffice, for example, the ordered key number which rolls over every $2^{16} = 65536$ keys. However, the use of hash on the entire key provides additional features. The key schedule need not be rigid, linear, or sequential. The Provider and the CA can maintain several redundant level-2 and level-1 keys, respectively. Multiple keys, potentially managed in isolation by the Provider and the CA, might provide redundant security. The Provider would need to check to be certain that the two unexpired keys do not share the same 16-bit identifying hash; however, this would be a rare occurrence and the Provider would simply need to draw another key at random in that event.

SBAS Providers could broadcast each other's Hash Path Ends and key maintenance features to promote service continuity. Hence, we included the SBAS Service Provider ID in the metadata. SBAS Providers would not need to have access to one another's secret data; they would only need to serve as repeaters. Whereas Section 4.4 discusses higher MT51-frequency to support the local SBAS authentication from cold receiver start, this feature would be low frequency and serve to support the local Provider schedule. Having Providers broadcasting each other keys at low frequency would contribute to TFAF when transferring to a new SBAS Provider. For example, we consider the case of two adjacent SBAS systems, WAAS and EGNOS, and an aircraft traversing from Europe to North America. It will take several hours for an aircraft to make this journey. If EGNOS broadcasts the WAAS keys once an hour, any westward-bound aircraft will receive the WAAS keys it needs to operate before reaching North America. This decreases the WAAS TFAF to zero. At a minimum, SBAS Providers could broadcast keys from adjacent SBAS Providers. Without this feature, when an aircraft enters a new SBAS service volume the first time in a given week (or 10 or 100 weeks), it must act as a cold-start receiver for as long as required to collect the Authentication Stack, as specified in Section 4.4.

Spare bits in the authenticating pseudorandom metadata could accommodate other features. These include scheme hyperparameters, such as the choice of a hash function or key or HMAC length. MT51 messages could authenticate specific messages immediately with ECDSA, including those identified by 16-bit hashes. If keys are ever compromised, MT51 could also disseminate key revocations. A revocation MT51 would include eight 16-bit hashes of the affected keys and would only need to be broadcast until the keys have been removed using standard procedure. Because each MT51 broadcast is accompanied by a germane expiration time, the Provider can arbitrarily shorten the applicability of a particular key by rebroadcasting the MT51 with a different expiration time, under the assumption that receivers actually receive and record that updated broadcast and that the Hash Path salt is not disrupted, as described in Section 3.2.1. Receivers would need to remember key expiration time changes and key revocations. Receivers that did not receive these updates would remain vulnerable. If the Provider can remove data, the metadata must include a parameter that identifies a given specific authentication. This is because without the germane key hash, authenticating key hash, segment number, and authentication instance number, the individual segments of pseudorandom data will not be associated. MT51 can manage other authentication schemes, including the program adopted by GNSS as described in Section 4.2. A GNSS authentication, especially one built on TESLA, would save bandwidth by deferring the its maintenance to SBAS.

There are many features that can be accommodated by the scheme's modularity; however, if SBAS stakeholders prefer not to use these features, then the spare meta bits could also be used to aid in maintaining resistance to message loss.

Providers could add redundant HMACS to the MT51 message and thus authenticate messages redundantly to alleviate MT50 message loss. Moreover, the delivery of 128-bits of authenticating pseudorandom data could be augmented via the use of fountain codes (Fernandez-Hernandez et al., 2017). The use of fountain codes may increase the reliability of delivering MT51 messages and decrease the number of transmitted messages required for an authenticated first fix. The selection of 128-bits supports the scheme's efficiency because all data points are integrally divisible by 128. Special care must be taken to ensure that using fountain codes in the spare bits to augment those 128-bits would not change this integral-divisible property and thus will maintain efficiency.

A final MT51 design must accommodate all the features relevant to SBAS stakeholders while maintaining the KPIs. The most relevant KPI for this discussion is TFAF, because the use of more features will require more metadata. This will increase the number of unique MT51s required for OTAR. We proposed MT51 because the minimum MT51 OTAR Payload Segment size was 128-bits. The conveniences associated with this choice are discussed in other sections. This choice permitted us to specify maximum metadata. However, as defined in Table 3, there could be redundant identifying information given certain assumptions about key management. Furthermore, some of the bits of metadata are never used (e.g., delivery of a 128-bit salt will never require more than one page). Because this work reflects an academic context, we are concerned with the breadth of features possible, given that our 128-bit design already meets the KPIs. Therefore, we acknowledge that there are other ways of constructing MT51 metadata that will avoid definition redundancy and spare bits. As SBAS Stakeholders discuss their desires (e.g., for the metadata to become more or less crowded, and noting that even MT51 has some spare bits. Therefore, we offer several suggestions on how to consider balancing the design while maintaining scheme modularity, simplicity, and flexibility.

For example, a plausible MT51 design could include a 192-bit OTAR payload or a larger payload. This can be achieved via any of the following: (1) the SBAS Provider could identify keys by their expiration time, thus removing the need for the Germane and Authenticating Key Hashes in the metadata; however, this would eliminate the possibility of parallel and redundant key management; (2) any metadata regarding the keys could be placed on its own page, provided the authenticating signatures were derived from that data to ensure its security; and (3) the SBAS Provider and receiver could agree on a defined ordering, for example, that depicted in Table 4, for an aggregate OTAR payload. We note that the 128-bit MT51 design does not require the receiver to presume anything about the OTAR schedule or use. In our simulation, the receiver implementation proceeds without knowledge of the 100, 10, and 1 week cadence, nor any insight into whether the keys are managed rigidly or linearly.

A 192-bit OTAR Payload Segment length almost maintains the integer-divisibility. An Authentication Stack from Table 5 using AES-256 to encrypt level-1 keys and having a separate TESLA salt MT51 would be integer-divisible by 192 with 2304 total bits and 12 192-bit MT51s. Alternatively, as features are removed, rather than shifting bit allocations from the metadata to the OTAR Payload Segment, SBAS Providers could instead arbitrarily replace metadata bits with fountain codes. This suggestion would maintain the integer-divisibility of MT51 on the authenticating pseudorandom data, thereby ensuring there is no wasted zero-padding on the delivered data, and will also add the loss-tolerant properties of MT51.

We implore the SBAS Stakeholders to have a future-proofing mindset that includes consideration of attributes that may be needed decades in the future. For example, cryptographic primitives will break and will need to be replaced.

Likewise, quantum-computer resistant algorithms may be needed. The best way to maintain a future-proof mindset is to make MT51 agnostic to the data it is delivering and capable of expansion to any future pseudorandom data delivery requirements.

At a 128-bit OTAR Payload Segment, the required delivery of 16 unique MT51 already meets the KPIs. However, increasing the OTAR Payload Segment would decrease the number of unique MT51s. This would decrease the burden on the SBAS Schedule, decrease TFAF, and have a less of an effect on continuity and availability, among other considerations. Therefore, we expect the results shown in Section 6 to be repeatable in designs that feature a larger than 128-bit OTAR Payload Segment.

4.2 | Core Constellation Ephemerides Authentication

Another feature that is accessible because of this scheme's modularity is authentication of navigation messages from a GNSS application. GNSS messages have limited bandwidth and must maintain backward compatibility. MT51 provides a natural pathway that might be used to authenticate GNSS data with the 128-bit payload and spare meta-data bits. Table 3 provides the required metadata designations, including (1) the germane core constellation system; (2) the type of authenticating pseudorandom data; and (3) the authenticating pseudorandom data segment number. However, the payload will need to be modified to accommodate realistic operations.

Given the Satellite-Receiver-Earth geometry, we suggest that the 128-bit payload might be split into eight 16-bit payloads, as shown in Table 6. Each 16-bit HMAC within the MT51 payload corresponds to a specific satellite's broadcast ephemerides. The MT51 metadata informs the receiver which satellites correspond to the payload HMACs. By splitting the payload into smaller HMACs, the receiver need not have access to the entire set of ephemerides to authenticate data from individual satellites.

$$k_j^{\text{Ephemeride}_{\text{SVN}}} = \text{HMAC}(p_i^P, t_j \,||\, \text{PRN}_{\text{SBAS}} \,||\, \text{Frequency}_{\text{SBAS}} \,||\, \text{SVN}_{\text{Core}}) \tag{4}$$

$$s_j^{\text{Ephemeride}_{\text{SVN}}} = \text{HMAC}(k_j, \text{Ephemeride}_{\text{SVN}}) \tag{5}$$

Authenticating core constellation ephemerides exploits a main feature of TESLA which is its bandwidth efficiency. Core constellation ephemerides can derive from the same Hash Path as those distributed by MT50, meaning it would require minimal additional SBAS bandwidth. Equations (4) and (5) provide formulae that can be used to generate the necessary HMACs, where $k_j^{\text{Ephemeride}_{\text{SVN}}}$ is the HMAC key for a particular ephemeris' HMAC, $s_j^{\text{Ephemeride}_{\text{SVN}}}$ is the HMAC broadcast and used for authentication. p_i^P, t_j; PRN, Frequency, and SVN share the same definitions as those presented in Equation (1) and (2). Other information should be incorporated

TABLE 6
Bit Allocation for a Specific MT51 to Authenticate a Subset of Ephemerides With a 128-bit Payload.

Metadata, etc	HMAC1	HMAC2	HMAC3	HMAC4	HMAC5	HMAC6	HMAC7	HMAC8	CRC
98	16	16	16	16	16	16	16	16	24

into these definitions (e.g., whether the signature is for the current or previous ephemeris). These details will be considered in our future work.

There are three important security details that remain to be addressed. First, similar to MT50, if any of one of the HMACs returns unauthenticated, the entire set of ephemerides data must be discarded. Second, the HMACs derive from a cryptographically-independent keys derived from a TESLA Hash Point, similar to that described in Section 3.3. Equations (4) and (5) use the sending time t_j, the SBAS PRN and frequency, and the core constellation satellite vehicle number. Third, the ephemeris HMACs must be sent before the release of the corresponding Hash Point according to the loose-time synchronized schedule (i.e., at least six seconds in advance).

SBAS could authenticate the broadcast ephemerides of every satellite globally and the receiver can draw from the entire set of HMACs, including the few within the view of the receiver. The metadata segment number informs the receiver from which set of eight satellites the HMACs are derive. For GPS, this would mean four MT51s. However, MT1 and MT31 already provide an issues-of-data mask of the 92-most relevant satellites currently under correction for a particular SBAS. Therefore, we propose that the order of the eight HMACs might correspond to the order prescribed in the issues of data already provided to the receiver. The issue of data index, either the IODP or IODM, must be placed in the unused sections of the metadata, and the metadata segment number field would determine the relevant eight HMACs, in order, among the set of 92. Since the issue of data index requires only two bits, SBAS could use the unused Germane Key Hash, Germane Key Expiration, and Authenticating Key Hash to send 12 HMACs per MT51. Alternatively, one could define a new message type that delivers 13 HMACs, the issue of data index, and the segment number.

4.3 | MT50 Redundant Authentication

MT51s are self-authenticating messages resulting from the use of ECDSA. Upon receipt of the entire Authentication Stack, the receiver can assert authenticity to the level-1 key maintained by the CA. However, the MT51 messages are among the SBAS messages that are authenticated by TESLA and MT50. If the receiver has an ECDSA-verified Authentication Stack that can authenticate the bulk of SBAS messages via MT50, the receiver can use TESLA to verify the next Authentication Stack when it is delivered by MT51. In this manner, a receiver can assert the authenticity of the next Hash Path End without awaiting the associated authenticating pseudo-random data since an HMAC in the following MT50 will assert authenticity down to the current Hash Path End through the current level-1 key. In Section 4.4, we suggest that the next Authentication Stack might be rebroadcasted each hour. The ECDSA- and TESLA-based authentication of the next Authentication Stack will then be redundant to a receiver that has undergone an authenticated fix. Therefore, an argument can be made that broadcasting the hour-long frequency of the next Authentication Stack does not add to the scheme. Alternatively, MT50s could not authenticate MT51s will instead authenticate messages redundantly. These considerations will ultimately depend on the manufacturers' implementation preferences and whether this logic should be incorporated into the process. To avoid implementation errors that might serve to exploit the authentication scheme, we suggest that, given stakeholder agnosticism, the simplest version might be selected. We believe that the scheme delineated above, where MT51 data is redundantly authenticated with MT50, is the simpler scheme overall.

4.4 | MT51 Schedule Frequency

As discussed earlier, we propose that the cryptoperiod of the level-1, level-2, and TESLA Hash Paths might be 100, 10, and 1 week, respectively. Our selections are somewhat arbitrary and were chosen in an attempt to balance the bandwidth required to rotate key and Hash Path instances with security considerations such as the resources required for a brute-force attack and the likelihood that a key becomes compromised (e.g., leaked). Feasible scheme designs might permit level-1 cryptoperiods to be years longer, level-2 cryptoperiods to be as short as one month or one week, and Hash Paths to rotate every hour or at least once a day. While we will leave this consideration for future work, any choice made must meet minimum security requirements that are associated with the required computational time needed for exhaustive guessing of the keys. Upon cold receiver startup, we would like the TFAF be reasonably short. Therefore, the Authentication Stack must be sent out periodically. As shown in Table 5, a complete Authentication Stack requires 16 MT51s. We propose that the Provider might broadcast the current Authentication Stack every five minutes. In this way, the cold-start receiver achieves the first authenticated fix within the first five minutes, assuming no message loss. This will require a message to be transmitted from the 16 unique MT51s at a rate of 1 out of every 18 messages. With multiple geostationary satellites and frequencies, the satellites can broadcast the unique MT51s out of phase to decrease the TFAF as discussed in Section 3.2.

Furthermore, the next keys, applicable immediately upon expiration of the current keys, must be sent well before they are needed to facilitate a seamless transition as the current keys expire. To protect the security of a specific new key, it would be prudent to send the new key just before it will be put into use. For instance, we suggest that the Provider might send the next 100-week level-1 key repeatedly beginning five weeks before its actual use. The next 10-week level-2 and 1-week TESLA Hash Path End might be sent one week before their actual use. To demonstrate its minimal impact on the MT51 SBAS bandwidth, if a Provider sent the next Authentication Stack once every hour, together with the current Authentication Stack once every five minutes, the Provider would need to send a message from the set of 32 unique MT51s at a rate of 1 out of every 17 messages, rounded up. This assumes that each OTAR Payload Segment of the Authentication Stack is broadcast at the same frequency as a baseline. Since some sections of the Authentication Stack change less frequently, further optimization could be performed to balance the TFAF with the likelihood that receivers are off for longer than for one week, 10 weeks, or 100 weeks. For instance, the slowly varying OTAR Payload Segments could be broadcast once every 15 minutes with the weekly-varying OTAR Payload Segments broadcast every two minutes.

We note that Providers and receivers may not need to adhere to a set schedule for the key periods or the Hash Paths. As discussed above, the MT51 metadata and modularity afford flexibility with respect to the expiration of particular keys and the agreed-upon schedule for their expiration. Moreover, switching Hash Paths without warning does not add computational complexity to receiver verification. Each Hash Point is a hash, and hashes have constant-time look-up computational complexity, as presented in Algorithm 2. This means that if the Provider were to switch Hash Paths without warning, the receiver would need only one additional computation to check that the new Hash Point is the preimage to another ECDSA-verified Hash Path End. This means that SBAS could incorporate unscheduled or stochastic Hash Path switches, provided the appropriate MT51s are sent in advance, noting that the MT51s specify only Hash Path End expiration times and not their actual

use time. If stakeholders elect not to have a rigid Hash Path schedule, a maximum Hash Path length upper bound must be specified so that receivers do not get stuck in an infinite loop as they attempt to hash down to an ECDSA-authenticated Hash Path End.

The selection of the level-1 cryptoperiod, level-2 cryptoperiod, and Hash Path length pose trade-offs among various SBAS stakeholders. With a very long Hash Path, aircraft are less likely to find themselves without the current Hash Path End via MT51 since the Hash Path End expires less frequently. Upon startup after a shutdown less than the Hash Path End cryptoperiod, receivers need not await delivery of an updated Authentication Stack via MT51, however, they must accommodate a more intense startup hashing operation to hash the current MT50 Hash Point to the authenticated Hash Path End. With a very short Hash Path (e.g., a Hash Path cryptoperiod of one day), receivers are more likely to be missing part of the Authentication Stack after a shutdown because the missing parts expired while the receiver was shut down. However, a shorter Hash Path will mean that less hash processing will be required at startup. These considerations become especially germane for aircraft that periodically traverse different SBAS service volumes. For instance, consider aircraft that regularly travel a transatlantic route. When outside a specific service volume, their devices will behave as shutdown receivers. Our selection described in Section 4.4 was to accommodate a short time to first fix. With a longer Hash Path, as flights travel across service volumes, the TFAF will more likely be bounded by receiver computational hash capability instead of the MT51 delivery schedule.

5 | TESLA TIME SYNCHRONIZATION

The HMAC keys authenticating all SBAS messages with HMACs are derived from the Hash Path delay-release on an assumed schedule. Therefore, it is critical to the Hash Path security that the Provider and the receiver are loosely time-synchronized. Loosely time-synchronized means that the receiver has sufficient externally-trusted time to reject HMACS after the release of the associated Hash Point. This poses a "Catch-22" for aircraft that use an isolated GNSS receiver, given that GNSS provides the time function. GNSS ranging signals allow receivers to derive an accurate, atomic-clock-synchronized time. This time measurement is certainly accurate enough to support the delay-release schedule. The prior art details many ways that receivers can establish trust in GNSS ranging signals (Fernandez-Hernandez et al., 2019; Psiaki & Humphreys, 2016); however, GNSS ranging signals are not yet rigorously authenticated with cryptography.

To damage SBAS security by breaking the loosely time-synchronized assumption, an adversary must spoof the receiver by delaying the receiver's time estimate. In our case, this delay is six seconds. If the receiver clock was six seconds behind the Provider's clock, an adversary listening to the Provider could identify the delay-released Hash Point, derive the keys, and generate forged HMACs. The longer the delay, the more HMAC forging can be accomplished by an adversary. While one could ensure that the receiver does not allow six-second time intervals; however, this strategy would not work with to combat a Creeping Replay Attack. The worst-case attack model will be examined in this work. In a Creeping Replay Attack, an adversary listens to and replays the GNSS and SBAS signals but introduces an incremental delay slowly enough to avoid detection by the receiver. There are several strategies that can be used to mitigate the Creeping Replay Attack.

The first mitigation strategy is to use the onboard clock to evaluate a spoofed-time hypothesis. The prior art has explored several mechanisms that might be used to examine clock tolerances and to generate and use tolerance bounds to assert clock trust (Fernandez-Hernandez et al., 2020). We have explored a few methods that might be used to fuse GNSS and the onboard clock to determine whether a receiver is actively involved in a Creeping Replay Attack, such as Pearson hypothesis tests and Kalman-filter-based hypotheses tests. Explicit hypothesis methods and their evaluation are left for future work. These strategies can detect a Creeping Replay Attack before the six-second breakage boundary, provided the delay rate is faster than the uncertainty bound of the onboard clock. For instance, a consumer quartz clock oscillator will gain or lose approximately 15 seconds each month. This means that any hypothesis that uses the onboard clock exclusively to estimate authentic time is information-bound to 15 seconds per month. Aircraft technicians and airport security officials could consider implementing procedures that guarantee that the aircraft receiver periodically receives trusted GNSS and SBAS signals, for example, while taxiing on the runway, where security officials ostensibly monitor for spoofed signals and jamming. Operators would need to communicate with the receiver that the current estimate can be trusted to reset the information-bound consideration. Receiver manufacturers could invest in more accurate clocks that tie time-trust events to periodic maintenance.

Another method is to compare to an external-to-GNSS, trusted time. Every day, billions of devices establish time via rigorous cryptographic authentication via the Internet. This system addresses tangential security concerns such as Internet-based banking. This typically works well because the Internet does not need to accommodate for an SBAS bandwidth limitation. A receiver could compare its own time to an Internet- or cellular-based time. This comparison need only be bound to the GNSS-based time. In other words, the receiver can still utilize the accurate time derived from the GNSS positioning regression, but it can check that this matches the Internet-based time within a boundary such as one second. This internet or a data-based time can be rigorously authenticated with cryptographically according to the standard of the medium. The original TESLA protocol provides a secure time-synchronization procedure. The Provider would establish an Internet server that would accept and operate on receiver-generated nonces that would work within the secure Hash Path (Perrig et al., 2005). Our choice to use the word "compare" is intentional in this case, given the need to address concerns about allowing external time inputs regarding to have access to an isolated SBAS receiver. However, this comparison can be achieved without exposing the receiver time to hacking strategies. Other strategies can be used to alert the pilot that the SBAS time is not trusted, such as incorporating the GNSS time estimate into the Air Traffic Control Radar Beacon System. If the beacon-broadcasted time was more than six seconds behind the trusted time on the ground, then ground services could alert the aircraft of the discrepancy.

Because the ranging signal remains unauthenticated despite rigorous methods established with cryptography, SBAS must take a nuanced approach to assert the loosely time-synchronized assumption.

6 | FULL STACK SIMULATION AND KEY PERFORMANCE INDICATORS (KPIS)

To examine the efficacy of this scheme, we implemented a full-stack simulation of this scheme in MAAST. The MAAST Matlab implementation known as

the secure com.sun.security.auth java package was used to simulate actual SBAS message transmission with ECDSA and TESLA routines asserting authentic messages and test that to verify that forged messages could be detected. At the time of publication, the entire scheme was implemented except for (1) the hash path salt; (2) the PRN concatenation operation for multiple satellites; (3) the pre-stored encrypted level-1 public keys; and (4) testing for alert cases. These features do not materially affect the outcomes because we took special care to ensure that the simulated scheme had the same number of unique messages per Authentication Stack. Implementing these additional features will be left to future work after we receive feedback from the SBAS stakeholders.

Using MAAST, we modified the message scheduler to send MT50s at a rate of one in six messages and send MT51s whenever the schedule had an opening. For this work, MAAST implemented the Authenticate-Then-Use (ATU) framework for the receiver (Walter et al., 2021). Given the current use of the SBAS message schedule, MT51s were sent more frequently than the requirement specified above, at 1 in 17 messages. Therefore, among our simulations, TFAF occurred within two minutes, rather than the required five minutes. The TFAF that resulted was lower than that required as discussed in Section 4.4 because the simulated Provider sent

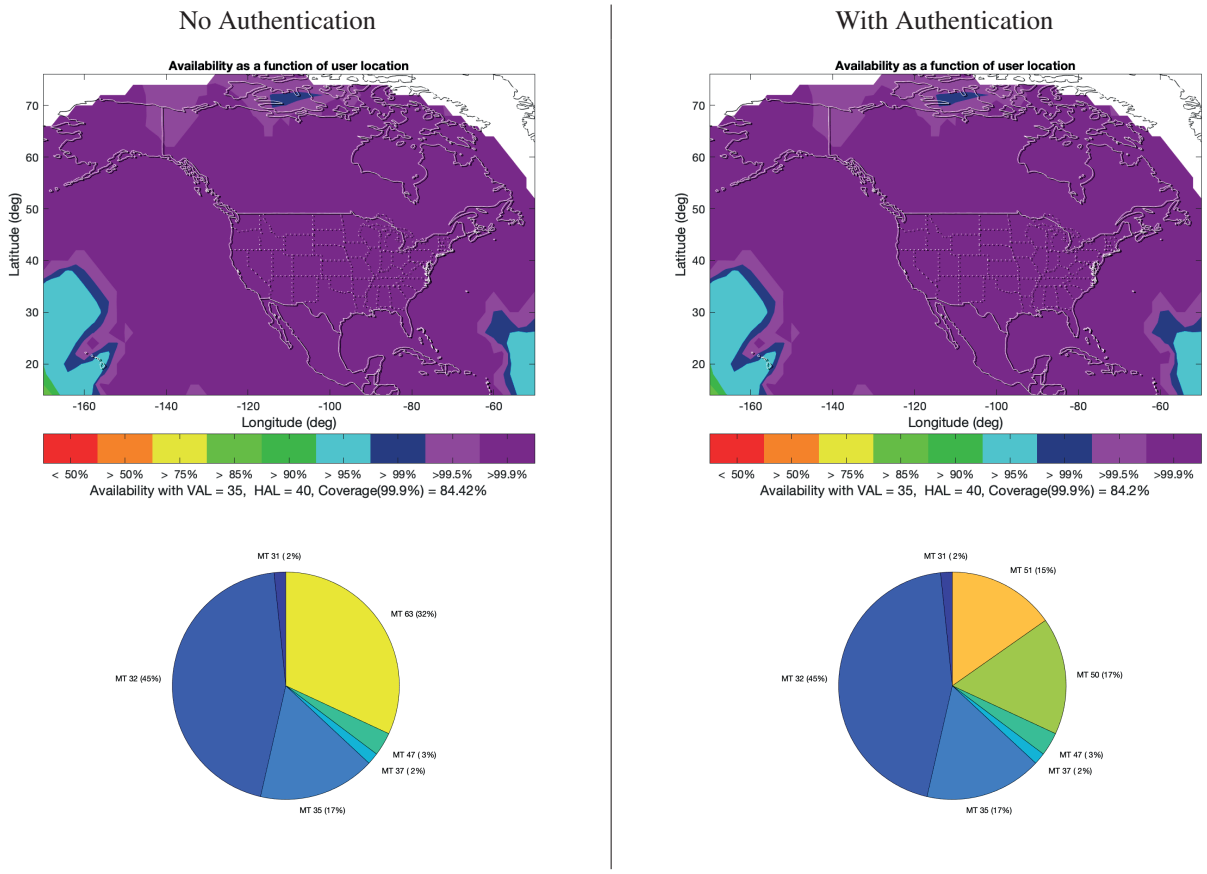

FIGURE 5 Availability of contour maps from the output of MAAST
No authentication is shown on the left and with authentication is shown on the right. In the authentication case, messages were accepted by the receiver only after authentication. The pie graphs indicate the distribution of the different messages with no authentication (left) and with authentication (right). With authentication, MT51s were sent in 1 of every 6 messages. While our requirement specifies MT51 is to be sent in 1 of every 17 messages (i.e., about 6%), the simulation schedule replaces the remaining MT63s with MT51s. There are some small coverage differences between the figures, including north of Canada at approximately $(-130^\circ, 72^\circ)$.

the 16 unique messages within open slots of the SBAS schedule. Therefore, our simulation indicates that SBAS availability and continuity are not affected by this authentication scheme. Figure 5 displays availability maps that compare two cases, including one with no authentication and another with faithful authentication via MT50 and MT51 with message information accepted only after authentication. While the availability maps appear slightly different from one another, most noticeably so at the service volume boundaries over Alaska and Canada, given the reasonable fidelity of MAAST, we consider the results indistinguishable. The pie graphs below indicate the distribution of the different messages with no authentication (left) and with authentication (right). In the authentication case, MT51s are sent in 1 of every 6 messages. While our requirement specifies that MT51 is to sent at a rate of 1 of every 17 messages (i.e., about 6%), this simulation schedule replaces the remaining MT63s with MT51s. Given the nearly identical results under the reasonable fidelity of MAAST, we find these results demonstrate the viabililty of the scheme presented in this work.

7 | CONCLUSION

This works delineates a complete TESLA-based SBAS authentication scheme that includes OTAR. The scheme relies on three levels of security: (1) 128-bit-security TESLA; (2) 128-bit-security ECDSA; and (3) 256-bit-security ECDSA. Using this scheme, the two message types are appended to the schedule. It does not require removal of power from the I-channel to support a Q-channel strategy. This strategy is immediately backward compatible because older receivers can ignore the new message types. It is also flexible and can be expanded according to the needs of and additional feedback from SBAS Stakeholders. The flexibility of the scheme derives from the observation that a single message can be used for all cryptographic maintenance. Given the reasonable use of onboard clocks, external clocks, and maintenance patterns, we assert the existence of reasonable, nuanced strategies that might be used to mitigate attacks against the loose time-synchronized assumption required by TESLA.

We tested the scheme with a faithful, full-stack simulation that includes full encoding and decoding of messages and use of an appropriate cryptographic library. Since the appended information fits within the unused slots in the message schedule, our simulation revealed only negligible differences in performance of the simulated receivers. Moreover, since the scheme makes use of space within the I-channel and not the Q-channel, no associated power decrease or loss of service at volume boundaries was observed. Therefore, we find this scheme, or one that is substantially similar, will be acceptable for use in SBAS authentication.

ACKNOWLEDGMENTS

We gratefully acknowledge the support of the FAA Satellite Navigation Team for funding this work under Memorandum of Agreement #693KA8-19-N-00015.

REFERENCES

Anderson, J., Lo, S., Neish, A., & Walter, T. (2021). On SBAS authentication with over-the-air rekeying schemes. *Proc. of the 34th International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS+ 2021)*, St. Louis, MO. 4288–4304. https://doi.org/10.33012/2021.18132

Anderson, J., Lo, S., & Walter, T. (2022). Efficient and secure use of cryptography for watermarked signal authentication. *Proc. of the International Technical Meeting of the Institute of Navigation (ITM 2022),* Long Beach, CA. 68–82. https://doi.org/10.33012/2022.18228

Ardizzon, F., Laurenti, N., Sarto, C., & Gamba, G. (2022). It's Galileo time: Options for crystal oscillators in OSNMA-enabled receivers. GPS World. https://www.gpsworld.com/its-galileo-time-options-for-crystal-oscillators-in-osnma-enabled-receivers/

Boneh, D., & Shoup, V. (2017). A graduate course in applied cryptography. Retrieved from: https://crypto.stanford.edu/~dabo/cryptobook/BonehShoup_0_4.pdf

Cancela, S., Calle, J. D., & Fernández-Hernández, I. (2019). CPU consumption analysis of TESLA-based navigation message authentication. *Proc. of the European Navigation Conference (ENC 2019)*, Warsaw, Poland. 1–6. https://doi.org/10.1109/EURONAV.2019.8714171

Caparra, G., Sturaro, S., Laurenti, N., & Wullems, C. (2016). Evaluating the security of one-way key chains in TESLA-based GNSS navigation message authentication schemes. *Proc. of the International Conference on Localization and GNSS (ICL-GNSS 2016)*, Barcelona, Spain. 1–6. https://doi.org/10.1109/ICL-GNSS.2016.7533685

Fernandez-Hernandez, I., Calle, D., Cancela, S., Fernández, A., Martínez, R., Seco-Granados, G., & Walker, P. (2017). Fountain codes for GNSS. *Proc. of the 30th International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS+ 2017),* Portland, OR. 1496–1507. https://doi.org/10.33012/2017.15368

Fernandez-Hernandez, I., Walter, T., Alexander, K., Clark, B., Châtre, E., Hegarty, C., Appel, M., & Meurer, M. (2019). Increasing international civil aviation resilience: A proposal for nomenclature, categorization and treatment of new interference threats. *Proc. of the International Technical Meeting of the Institute of Navigation (ITM 2019)*, Reston, VA. 389–407. https://doi.org/10.33012/2019.16699

Fernandez-Hernandez, I., Walter, T., Neish, A., Anderson, J., Mabilleau, M., Vecchione, G., & Châtre, E. (2021). SBAS message authentication: A review of protocols, figures of merit and standardization plans. *Proc. of the International Technical Meeting of the Institute of Navigation (ITM 2021)*. 111–124. https://doi.org/10.33012/2021.17829

Fernandez-Hernandez, I., Walter, T., Neish, A., & O'Driscoll, C. (2020). Independent time synchronization for resilient GNSS receivers. *Proc. of the International Technical Meeting of the Institute of Navigation (ITM 2020)*, San Diego, CA. 964–978. https://doi.org/10.33012/2020.17190

Fernández-Hernández, I., Rijmen, V., Seco-Granados, G., Simon, J., Rodríguez, I., & Calle, J. D. (2016). A navigation message authentication proposal for the galileo open service. *NAVIGATION*, 63(1), 85–102. https://doi.org/10.1002/navi.125

Jan, S.-S., Chan, W., Walter, T., & Enge, P. (2001). Matlab simulation toolset for SBAS availability analysis. https://web.stanford.edu/group/scpnt/gpslab/pubs/papers/Jan_IONGPS_2001.pdf

Neish, A. (2020). Establishing trust through authentication in satellite based augmentation systems [Doctoral dissertation, Stanford University, Stanford, CA, USA]. https://web.stanford.edu/group/scpnt/gpslab/pubs/theses/Neish-Thesis-Final.pdf

Neish, A., Walter, T., & Enge, P. (2019). Quantum-resistant authentication algorithms for satellite-based augmentation systems. *NAVIGATION, 66*(1), 199–209. https://doi.org/10.1002/navi.287

Neish, A., Walter, T., & Powell, J. D. (2019). Design and analysis of a public key infrastructure for SBAS data authentication. *NAVIGATION, 66*(4), 831–844. https://doi.org/10.1002/navi.338

O'Hanlon, B., Rushanan, J. J., Hegarty, C., Anderson, J., Walter, T., & Lo, S. (2022). SBAS signal authentication. *Proc. of the 35th International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS+ 2022)*, Denver, CO. 3369–3377. https://doi.org/10.33012/2022.18443

Perrig, A., Song, D., Canetti, R., Tygar, J., & Briscoe, B. (2005). Timed efficient stream loss-tolerant authentication (TESLA): Multicast source authentication transform introduction. Request For Comments, 4082. https://www.rfc-editor.org/rfc/rfc4082

Peyrin, T., Sasaki, Y., & Wang, L. (2012). Generic related-key attacks for HMAC. In X. Wang & K. Sako (Eds.), Advances in cryptology – asiacrypt 2012. 580–597. Springer Berlin Heidelberg. https://eprint.iacr.org/2012/684.pdf

Psiaki, M. L., & Humphreys, T. E. (2016). GNSS spoofing and detection. *Proc. of the IEEE, 104*(6), 1258–1270. https://doi.org/10.1109/JPROC.2016.2526658

Various. (2021). Galileo open service navigation message authentication (OSNMA) user ICD for the test phase. In (Vol. 1). https://www.gsc-europa.eu/sites/default/files/sites/all/files/Galileo_OSNMA_User_ICD_for_Test_Phase_v1.0.pdf

Walter, T., Anderson, J., & Lo, S. (2021). SBAS message schemes to support inline message authentication. *Proc. of the 34th International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS+ 2021)*, St. Louis, MO. 474–484. https://doi.org/10.33012/2021.17908

How to cite this article: Anderson, J., Lo, S., Neish, A., & Walter, T. (2023). Authentication of satellite-based augmentation systems with over-the-air rekeying schemes. *NAVIGATION, 70*(3). https://doi.org/10.33012/navi.595