Efficient and Secure Use of Cryptography for Watermarked Signal Authentication

Jason Anderson, Sherman Lo, Todd Walter

Abstract

This work discusses and applies techniques to efficiently and securely use cryptography for signal authentication in civil (open) satellite navigation signals such as Chips-Message Robust Authentication (“Chimera”). By efficient and secure, we mean maintaining a cryptographic security strength requirement with minimum data bandwidth and watermark signal degradation. We exploit many strategies, including using Timed-Efficient Stream Loss-tolerant Authentication (“TESLA”), sectioned parallelized cryptographic secret distribution, and only one constant, independent spreading code watermark degradation. Our design allows multi-cadence (i.e., slow and fast) distribution even with only a single watermark degradation. Our design maintains standard 128-bit security with a single minimum bandwidth channel while allowing for authentication of many GNSS services.

I. INTRODUCTION

1. GNSS Authentication

GNSS allows receivers to deduce their position and time based on the Trilateration problem. The signal is composed of three layers. The base layer is the carrier wave that serves as the GNSS signal’s band. The spreading code exists on top of the carrier wave to allow multiple satellites to utilize the same carrier wave. The data channel exists on top of the spreading code.

A spreading code is a pseudorandom code unique to each broadcasting satellite. The set of spreading codes used must have low cross-correlation, such as the cross-correlation bound with the Gold Codes, or stochastically bounded assuming a pseudorandom function’s (“PRF”) unpredictability. A receiver autocorrelates a replica of each spreading code to its received raw signal to deduce the time-of-flight of each signal. The data channel provides ephemeris and other navigation data so that a receiver can compute the satellite’s position at the time of signal broadcast. The combination of signal time-of-flight information and navigation information provided by the data channel solves the Trilateration problem for receivers to deduce their position and time.

Concepts for authentication of civil GNSS signals, such as Chimera, incorporate unpredictable watermarks insertions into the GNSS spreading codes which later allows authentication after the insertions are disclosed[1]. The watermarks have a limited effect on the spreading code allowing legacy receivers to continue to use the signal without authentication. Receivers who authenticate their signal must store raw radio frequency data containing the watermarked spreading code. After the GNSS provider broadcasts the watermarked spreading codes, the GNSS provider distributes the cryptographic seed so that receivers can generate the watermarked replica. Receivers can then go back and re-auto-correlate the spreading codes. The receivers consider the signal cryptographically authentic if the replica-to-raw autocorrelation increases correctly after incorporating the watermarks into the replicas. Previous work has suggested a way to derive watermarks from cryptographic seeds [2], but some specific application cryptographic formulae have not been published to our knowledge. This work broadly conceptualizes how the schemes, such as Chimera[3], should best use cryptographic primitives inspired by previous TESLA customization for SBAS [4].

This work describes multiple strategies to improve vital parameters for an authentication design (e.g., reduce data bandwidth, faster time to the first authentication) to distribute cryptographic seeds. First, we suggest and define a single TESLA instance for all satellites. Therefore, at a particular time, all cryptographic information for the entire system derives from the same cryptographic seed. We maintain security deriving different keys via HMAC, ensuring cryptographic independence while deriving from a minimal distribution seed. Second, we suggest a multi-cadence distribution construction that allows for multiple, different frequency distributions with a single TESLA instance, thereby eliminating the need for any slow or fast channel to be independent and minimizing the necessary degradation to the spreading code. Third, we suggest that GNSS providers distribute sections of a particular seed in parallel over the available channels. Parallel section distribution allows for increased frequency of TESLA key changes, thereby decreasing the time to authentication. We consider how an entire GNSS constellation could implement this idea using multiple strategies.

We propose a watermarking function that provides valuable properties to the Chimera design and stakeholders. Specifically, our watermarking function degrades the spreading code the same amount each crypto period. Alternatives may degrade the spreading code stochastically with a uniform distribution. Our watermarking function achieves this feature by exploiting a shuffling operation that derives from a cryptographic seed with sufficient cryptographic entropy. By eliminating the need
for independent slow and fast channels, we minimize the necessary degradation of the spreading code and enforce that it is consistent.

2. TESLA

This section provides a basic introduction to Timed Efficient Stream Loss-tolerant Authentication (“TESLA”). We describe the protocol, its relationship to asymmetric protocols, and its advantages for the GNSS context. For concreteness, but without losing generality, when our protocol requires a secure cryptographic hash function, we use SHA-256. When our protocol requires a message authentication code, we use the key-hash message authentication codes (“HMAC”) with SHA-256 as its primitive. When our protocol requires an asymmetric authentication protocol, we use Elliptic Curve Discrete Signature Algorithm (“ECDSA”). Anywhere in the work where we use SHA-256, HMAC, or ECDSA, another analogous protocol could be used. For instance, another hash function could replace SHA-256, and another asymmetric protocol could replace ECDSA (e.g., EC Schnorr). We make our selection based on standardized, widely-used, and well-researched at the time of writing to aid with the scheme’s security. We use our selections explicitly to make this work more concrete, concise, and intelligible.

With ECDSA, a provider produces a message and a signature for a receiver to digest and authenticate immediately. Since the signature is cryptographically pseudorandom, it could serve as the cryptographic seed of the watermarks. Since the spreading code delivered by the GNSS ranging service exists below the thermal noise floor, the receiver must have a spreading code replica to deduce a pseudorange. Any cryptographic pseudorandom information added to the spreading code must be delivered to receivers separately from the spreading code itself because of the signal’s low received power. Therefore, even an asymmetric authentication of the spreading code acts as a bit-commitment, delay-release authentication scheme. We claim that in the context of a GNSS ranging service undetectable by a receiver without a replica, all authentication schemes act as bit-commitment-effective schemes. When considering ECDSA versus a bit-commitment scheme, the advantage of ECDSA in other non-GNSS contexts (e.g., internet authentication) is that the receiver can use authentication information to authenticate immediately, as opposed to the required delay for a bit-commitment scheme. Since the nature of GNSS invalidates this principal feature of ECDSA, it would be better to use a proper bit-commitment scheme, such as Timed Efficient Stream Loss-tolerant Authentication (“TESLA”), to exploit the additional features provided.

TESLA provides several advantages relevant to GNSS over exclusively using ECDSA. We refer to RFC 4082 for the details of the protocol. Three advantages relevant to GNSS are the following. First, a small cryptographic distribution can authenticate unlimited information, alleviating the GNSS data bandwidth constraint. Second, if a receiver misses a cryptographic distribution, a receiver can rederive that distribution from a later distribution, providing loss tolerance. Third, the cryptographic primitives can be more computationally efficient, saving receiver computation and power. TESLA assumes that the receiver is loosely time-synchronized to the provider. One must use TESLA in tandem with ECDSA. ECDSA’s use with TESLA effectively acts as the rekeying maintenance operation for the TESLA Hash Path, so it is treated as such in this work like in [T].

TESLA requires the construction of a one-way set of n-bit integers related via repeated application of a cryptographic hash function, where n is the security level of the protocol. Each n-bit integer among the set serves as an HMAC key at a particular time agreed to by the provider and receiver. Since the audience of this work are experts in navigation, we use geometric terms path and point instead of key chains and key and describe TESLA with the geometry of a one-way path. Later in this work, the path metaphor better expands when accommodating features relevant to GNSS. Let each n-bit integer be called a Hash Point, and let a collection of Hash Points consecutively related via the Hash Function be a Hash Path. Non-consecutive Hash Points further along the Hash Path relate via repeated application of the Hash Function. We call the first Hash Point along the path the Hash Path Start, and the Hash Path Start is just a random Hash Point, i.e., a random n-bit integer. We call the last Hash Point along the path the Hash Path End. Since the Hash Function is secure, there is no known efficient algorithm that can compute the input Hash Point to the Hash Function that yields a specific output Hash Point. In other words, it is trivially easy to compute the output Hash Point of the Hash Function given the preimage Hash Point, but one can only use exhaustive search to find any preimage Hash Point. It is a one-way path. The domain of n-bit integers, together with a randomized n-bit salt inclusion to the Hash Function and $n \geq 128$, make precomputation attacks (also called Rainbow Table Attacks) infeasible with modern supercomputers. The Hash Path End must be signed with ECDSA to complete the security.
Figure 1: A conceptual diagram of a basic stream of messages authenticated with TESLA. An authenticated provider will generate a Hash Path before using the stream and hold it secret. Provider releases the Hash Path End signed with ECDSA, then uses the secret preimages to sign the message stream. The provider releases the Hash Path backward to authenticate the message stream.

Figure 1 provides a conceptual diagram of a simple stream of messages authenticated with TESLA. A provider of an authenticated stream of messages follows the following protocol: First, the provider must generate a complete Hash Path and hold the entire Hash Path secret, except the Hash Path End. In Figure 1 the length of that Hash Path is 100 Hash Points with each Hash Point abbreviated “HP”. Second, the provider uses ECDSA to sign the Hash Path End (HP 100 in Figure 1) and distributes the Hash Path End with an ECDSA signature. Third, the provider sends a message with an HMAC derived from that message and the secret preimage of the Hash Path End (HP 99 in Figure 1). Fourth, the provider sends the preimage of the Hash Path End (HP 99 in Figure 1). Fifth, the provider returns to the third step to send a new message and HMAC with the secret second preimage of the Hash Path End (HP 98 in Figure 1), and so on. Once the provider runs out of preimage Hash Points to continue the stream, it must return to the first step.

A receiver follows the following protocol to authenticate a message: First, a receiver receives a message and its HMAC, noting the time of receipt. Second, following the agreed loosely time-synchronized schedule, a receiver will prepare to refuse messages and HMACs derived from a Hash Point after the Hash Point’s expected release. Third, a receiver receives the Hash Point (the “incoming Hash Point”) corresponding to the message and HMAC of the first step. A receiver authenticates the message via the following steps: A, B, and C. Step A, a receiver checks that the hash of incoming Hash Point is the previously received Hash Point in the stream. Step B, a receiver checks that repeated hashing of the incoming Hash Point eventually leads to the Hash Path End signed with a valid ECDSA signature. With the appropriate receiver internal caching. Step B only needs to be completed once per startup. Step C, a receiver checks that the message with the incoming Hash Point generates the HMAC.

The principal advantages of GNSS authentication are the following. An ECDSA is only required once per Hash Path, and multiple pieces of information can be authenticated per Hash Point, thus saving bandwidth and computation. We greedily exploit this property in this work, using HMAC as a secure key-generation function. We call HMAC’s use as a secure key-generation function Hash Path forking to continue the path metaphor. Hash Points missed can be derived from later Hash Points, thus providing loss-tolerant properties. The protocol is secure, provided the security of the hash function and adherence to the loosely time-synchronized schedule. Upon receiving a Hash Point via public broadcast, an adversary cannot forge messages because receivers know to reject messages signed with a released (and now expired) Hash Point. An adversary cannot know a preimage Hash Point along the Hash Path because the Hash Path remains secret except to the authenticated provider. An adversary does not have sufficient computational resources to invert a secure hash function in a reasonable time.

TESLA assumes a loose time-synchronization and provides a procedure to establish that assumption[5]. In summary, the start-up receiver clock delay to the provider clock and expected receiver clock drift must be bounded so that the receiver does not accept information signed with already-released Hash Points. While this explicit time-synchronization can add complexity, an ECDSA-only scheme suffers from the same consideration behind the scenes. A network query can ensure that this bound is satisfied in the general TESLA case. In the GNSS-only case where GNSS provides the time, a receiver must assume the first GNSS timing estimate is within that bound. Using an onboard clock that checks for spontaneous clock jumps from spoofing and implementing procedures (e.g., recurring maintenance), one can mitigate clock spoofing risks at receiver start-up and allow an accurate clock to deter creeping delay attacks during receiver operation. This is discussed further in Section III.
II. EFFICIENT GNSS WATERMARKED SIGNAL AUTHENTICATION WITH TESLA

Within this Section [1] we structure our proposed TESLA application in the following order. First, we provide concrete mathematical definitions to our proposed scheme with Equations [1] through [8]: Section [1] Second, we describe the general design with conceptual diagrams: Sections [1,2] [1,3] [1,4] and [1,5]. Third, we describe, in detail, why particular operations of Equations [1] through [8] are required: Section [1,6]. Broadly, to customize TESLA for a GNSS watermarked signal authentication, we use a sophisticated Hash Path with recurring forks. All cryptographic authentication information derives from a minimal distribution to best accommodate the GNSS bandwidth restrictions.

1. Definitions

Let $P$ be a particular Hash Path composed of many Hash Points $p_i$ indexed by $i$. The Hash Path Start $p_{0}$ is just a random n-bit integer. Let $S'P$ be another random n-bit integer of the particular Hash Path $P$, called the salt. Let $t_i$ be the integer time of the $i$-th Hash Point release. Equations [1] and [2] completely define the slow-channel Hash Path. They are labelled “slow” because their distribution will be over the slow channel: the GNSS data channel.

$$
\text{slow } p_0 \leftarrow \{0, 1, \ldots, 2^n - 1\} \\
\text{slow } p_{i+1} = H (\text{slow } p_i || S' || t_i)
$$

At each slow-channel Hash Point, we create a forked Hash Path with HMAC. Let each Hash Point along the forked slow-channel Hash Point $p_i$ be $p_{i,j}$, indexed by $j$. Let $t_{i,j}$ be the integer time of the release of the $i, j$-th Hash Point. Equations [3] and [4] completely define the required recurring forked Hash Paths, which make up the fast-channel Hash Paths in our scheme. They are labeled “fast” because their distribution will be over a side-channel that supports a fast, high bandwidth distribution. The text “Fast Channel Hash Path Start” in Equation [3], analogously in future equations, serves only as a salt to ensure each Hash Path Fork is a unique.

$$
\text{fast } p_{i,0} = \text{HMAC} (\text{slow } p_i, \text{“Fast Channel Hash Path Start”} || t_i) \\
\text{fast } p_{i,j+1} = H (\text{fast } p_{i,j} || S' || t_{i,j})
$$

From the fast-channel Hash Points, we fork again (as necessary) with HMAC to derive HMAC keys to authenticate ephemeris and other navigation message information with Equations [5] and [6]. All navigation message information will be signed by HMACs derived from fast-channel Hash Points released after broadcast of the authenticated information, so fast-channel users can authenticated slow-channel navigation messages data at the fast-channel cadence.

$$
\text{Ephemeris Signature Key}_{i,j} = \text{HMAC} (\text{fast } p_{i,j-1}, \text{“Ephemeris Signature Key”} || SVN || t_j) \\
\text{Ephemeris Signature}_{i,j} = \text{HMAC} (\text{Ephemeris Signature Key}_{i,j}, \text{Ephemeris}_{i,j})
$$

From the fast-channel Hash Points, we fork again with HMAC to derive seeds that determine the spreading code authentication watermarks with Equations [7] and [8]. Again, the watermarks will derive from HMACs from fast-channel Hash Points released after the broadcast of the watermarked spreading code. The $F$ of Equation [8] is discussed further in Section [1].

$$
\text{Water Marked Spreading Seed}_{i,j} = H (\text{fast } p_{i,j-1}, \text{“Water Mark Spreading Code Seed”} || SVN || t_j) \\
\text{Water Marked Spreading Code}_{i,j} = F (\text{Water Marked Spreading Seed}_{i,j}, \text{Spreading Code}_{i,j})
$$

We note the following to provide intuitive context. Due to the collision-resistance feature of the hash function, we do not expect Hash Path forks to collide or reconverge. Every forked fast-channel Hash Path must lead to a Hash Path End signed by ECDSA for fast-channel users only. If a receiver obtains a slow channel Hash Point, it can derive the entire fast-channel Hash Path fork and the navigation message signatures and watermarks. The entire Hash Path derives deterministically from a single random n-bit integer $p_{0}$, which is held secret. The Hash Path is released backward, meaning that later-released Hash Points allow a receiver to derive previously released Hash Points allowing loss tolerance. To authenticate additional information, one needs to create additional HMAC forks using unique HMAC message fields.

2. Multi-cadence distribution with a Single Watermark Degradation

We claim that a multi-cadence authentication is achievable with a single watermark. In the case of Chimera, the independent slow- and fast-channel watermark degradations can be combined, reducing the required spreading code degradation. To maintain
A conceptual diagram of how to create a multi-cadence distribution using only one set of watermarks. Each black arrow is a regular hash operation, with some concatenated information to maintain entropy to deter pre-computation attacks. The black arrows for the slow channel, Equation [9]; for the fast channel, Equation [4]. The green arrows use HMAC to fork the Hash Path: Equation [3]. Each arrow is a one-way cryptographic operation, meaning the derivable information is one-way. The diagram reads from left to right with increasing release time by provider. The more left an object is, the earlier it is broadcast by an authenticated provider. The main Hash Path is distributed via the slow channel. Recurring Hash Path forks compose the fast channel, which derives the watermark via the blue arrow operations. The use of ECDSA is not shown in this diagram, deferring to Figure 4.

A fast-channel-capable receiver completes the following procedure at a fast cadence to authenticate ranges. First, the receiver records raw radio data and ascribes that data at a time assumed compliant with the TESLA loose time-synchronization requirement. Second, the receiver receives a Hash Point from the fast channel. Third, the receiver authenticates that Hash Point via the typical TESLA procedure via a fast-channel Hash Path End, see Figure 4. Last, the receiver computes the watermarked spreading code and checks for the increase in autocorrelation. The required radio frequency storage is around the standard period of a single fast-channel Hash Point applicability.

For a receiver listening only to the GNSS signal (i.e., a slow channel only user), the receiver completes the following procedure at a slow cadence. First, the receiver records raw radio data and ascribes that data at a time assumed compliant with the TESLA loose time-synchronization requirement. Second, the receiver receives a Hash Point from the navigation message (i.e., the slow channel). Third, the receiver authenticates that Hash Point via the typical TESLA procedure via a slow-channel Hash Path End. Fourth, the receiver derives the entire forked fast-channel Hash Path, noting that each of those Hash Points has already expired. Last, the receiver computes the watermarked spreading code and checks for the increase in autocorrelation, and the receiver derives the navigation message HMACs to authenticate the data (explained further in Section II.3). If a receiver has constrained computation and power resources, the receiver can check the autocorrelation increase of a subset of the fast-channel watermark periods.

This scheme provides several advantages. First, this scheme needs only one watermark degradation. The alternative, where slow and fast channels each have their own independent watermarks, requires either (1) double the amount of watermark degradation or (2) half the autocorrelation bump to show an authenticated signal. Second, the scheme provides more flexibility to slow channel receivers. Rather than record raw radio data for the entire slow-channel epoch, it can listen to random subsections (since each subsection has fast-channel-level cryptographic watermarking density), thereby alleviating receiver hardware requirements. Moreover, authenticating additional information does not require additional data bandwidth distributed over the slow channel, the discussion of Section II.3


A principal advantage of TESLA is its scalability feature. A fixed TESLA Hash Point distribution can authenticate any amount of information distributed prior to the Hash Point distribution. We propose greedily exploiting this feature to authenticate all navigation message data. Figure 3 provides a conceptual diagram that portrays the combined slow- and fast-channel feature with navigation message authentication. Several diagram objects portray overlapping repetitions (e.g., watermarked spreading code, ephemerides). This overlapping repetition signifies the uniqueness of each satellite, frequency, and signal. The slow-channel and fast-channel Hash Paths do not have overlapping repetitions, meaning that the entire constellation or GNSS system...
Figure 3: A conceptual diagram on producing TESLA Hash Path forks to accommodate authenticating navigation messages and generating watermarks. The solid black arrows represent a regular hash, as in Equation 2 and 3. The red, green, and blue arrows represent the use of HMAC as a secure key derivation function, with the HMAC-message field necessarily unique. Green is Equation 3; Red, Equation 5; Blue, Equation 7. The purple arrows are the regular HMAC operation to sign data with a key and message field. The yellow arrows refer to the watermarking function in Section V. With the diagram, all items except the slow channel and fast channel Hash Path have overlayed copies. Overlayed copies represent the different instances among satellite vehicles. The entire constellation shares the same slow-and fast-channel Hash Paths for efficiency, specifically to aid Section III. The use of ECDSA is not shown in this diagram, deferring to Figure 4.

shares them. This choice aids in minimizing the slow-channel bandwidth requirement, thereby minimizing the slow channel time-to-authentication, as described in Section III.

The navigation message authentication HMAC keys must fork off the fast-channel hash path to allow fast-channel receivers to authenticate that information at the fast-channel cadence. The security of all the forks hinges on the one-way nature of the hash function and HMAC. With any use of HMAC, the message field must necessarily be contextually unique to ensure no repeated keys.

4. TESLA’s Required use of ECDSA and Rekeying

To achieve complete authentication security, each forked TESLA Hash Path must arrive at a Hash Path End authenticated via ECDSA. The bandwidth and loss-tolerance advantages arise when this is not required often. Any authenticating cryptographic scheme generally uses multiple hierarchical levels of ECDSA keys with varying strength and applicability time. Since the provider need not transmit ECDSA signatures of any Hash Path Ends and higher-level ECDSA signatures of lower-level ECDSA keys among the strict TESLA message stream according to the agreed schedule, Hash Path Ends, ECDSA keys and over-the-air rekeying, and ECDSA signatures are better handled within (and discussion deferred to) the rekeying framework. In other words, since the scheme must already provide a mechanism for rekeying the multiple, hierarchical levels of ECDSA keys, we assert that it is apropos to have that system distribute Hash Path End ECDSA signatures just like within a TESLA SBAS proposal [4].

Figure 4 provides a conceptual diagram of how each Hash Path Fork must hash to a Hash Path End signed via ECDSA. The slow channel Hash Path End ECDSA signatures can be distributed among the required rekeying information in the slow channel. Distribution can occur at a low frequency, depending on the desired time-of-first-lix time for a slow-channel receiver. Since a slow channel receiver will derive each fast-channel Hash Path fork from a slow channel Hash Point after complete expiration of the entire fast channel Hash Path fork, a slow channel receiver does not need to check the ECDSA signatures of fast channel Hash Path Ends. A slow-channel receiver receives an efficient distribution of the entire fast channel Hash Path after its expiry according to the bandwidth constraint of the slow channel. We assume that bandwidth constraints are not a concern for the fast channel, meaning the receiver can use the side channel to receive the ECDSA signatures for the fast channel Hash Path End.
5. SBAS as a Side Channel

The previous sections exploit the TESLA scaling feature to efficiently use bandwidth to authenticate information. We showed how to complete multi-cadence distribution and expand authentication to suit different receiver needs. To minimize the required authentication bandwidth of a GNSS scheme, it would be apt for the entire GNSS to use a single TESLA Hash Path, forking as needed for specialized applications. There are already proposals for SBAS authentication with TESLA [4], and Galileo’s Open Service Navigation Message (“OSNMA”) Authentication already uses TESLA [5][6][8]. Therefore, to minimize authentication bandwidth among GNSS, it would be apt to use TESLA Hash Paths for navigation message authentication and spreading code watermarking. Alternatively, one could fork the OSNMA Hash Path to generate watermarks for a Galileo watermarke, spreading code authentication concept. For the SBAS case, SBAS could serve as a fast channel for civilian aircraft. Sharing Hash Paths to save bandwidth does not introduce vulnerabilities in the scheme security, provided any Hash Point broadcast distribution occurs after a provider expires its use. Ideally, this would require tight authentication coordination among the managing entities (e.g., GPS, WAAS). One entity will have to generate the required Hash Path(s), and the other entities will either (1) need privileged access of the secret preimage Hash Points of the Hash Path and ensure no leakage or (2) act as a repeater distributing already expired Hash Points.

For instance, if TESLA is applied to SBAS[4], then a GNSS authentication scheme could fork those Hash Points for authenticating GNSS navigation data and producing watermarks. Or vice versa, the GNSS system manages the Hash Path, and SBAS acts as a repeater and authenticates SBAS information by forking the GNSS Hash Path. Effectively, the slow-channel distribution could provide a global slow channel distribution, and then SBAS could provide a fast-channel for all aircraft and other systems that must listen to SBAS already within the SBAS service volume. Considering the geometry of a Hash Path and the distribution schemes already in use, a short time-to-authentication for SBAS listeners could be provided without materially modifying plans for SBAS authentication.

6. Security Considerations Of Definitions

This section explains the security considerations in the construction of Equations (1) through (8). To begin, we note that several of Equations (1) through (8) use a concatenation operation ||. This is secure, assuming that the total HMAC message field input is less than the input block size of the hash function. For SHA-256, this is 512-bits. All of the definitions have fields less than 512-bits. If this were not the case, then Equations (1) through (8) would need to be modified to prevent extension attacks on the hash function.

Equations (2) and (4) each concatenate a Hash Point with a salt and time. We refer to other work that has considered in detail how these operations ensure the required amount of entropy to deter precomputation attacks on the Hash Path [9]. In summary, the Hash Path needs a counter and a salt. We use time as the counter to simplify the protocol and aid in detecting provider and receiver clock mismatch. By time, we mean the integer time (e.g., GPS time: week and time of week) of sending and receiving the Hash Point. Sending and receiving time rounded down to the nearest integer is the same for a ranging signal since the fight time is less than one second. Moreover, if we use time as the counter, we save bandwidth since we do not have to transmit metadata informing the receiver when a Hash Path Starts or Ends. In fact, given the hashing property of Hash Points, a receiver can check for a Hash Path switch at a constant time complexity [4]. To save bandwidth with the distribution of the salt, the salt can derive from the ECDSA signature on a Hash Path End, as suggested for SBAS [4].

In Equations (5), (6), and (7), HMAC serves as a secure key derivation function. The intermediate HMAC Hash Path forking
III. EFFICIENT SLOW-CHANNEL HASH POINT DISTRIBUTION

An important feature of the TESLA Hash Path designs of Section II is that all authentication information can derive from a minimal slow-channel Hash Point distribution. This minimal slow-channel Hash Point is shared among the entire GNSS constellation in Section I to aid the scheme presented in this Section III. The time-to-authentication is bounded below by a slow channel Hash Point period, herein the “Slow-channel Epoch”. The faster the cadence of slow-channel Hash Points distribution and expiry, the shorter the Slow-channel Epoch, the shorter the time-to-authentication. In other words, the faster the TESLA scheme rekeys itself by moving up a Hash Path, the faster the time-to-authentication. The bandwidth constraint of the slow channel wholly limits this criterion. This section proposes to minimize the time-to-authentication by exploiting how a receiver receives data from multiple satellites simultaneously.

A receiver must track at least four GNSS satellites to use the ranging service to deduce its position and time. Usually, the receiver tracks more than four GNSS satellites. We have specified that the entire constellation shares a slow-channel Hash Point to derive all authentication information. It would be redundant for each satellite to distribute that same slow-channel Hash Point identically. Therefore, we propose two strategies to utilize the distribution best. Figures 5 and 6 provide conceptual diagrams of the naive and the proposed schemes, respectively.

Figure 5 depicts a naive slow-channel hash point distribution scheme; whereas, Figure 6 depicts two more efficient suggestions: Interleaved and Sectioned Schemes. Both exploit how a single slow-channel hash point applies to the entire GNSS constellation and that receivers listen to multiple satellites that could provide parallel bandwidth data channels. By sharing the bandwidth, the cadence can increase, decreasing authentication time. Each diagram among Figures 5 and 6 each depict the distribution of nine Hash Points; however, the time-to-authentication in Figure 6 is shorter. For the sake of argument, and as provided in the conceptual diagrams of Figures 5 and Figure 6, consider a 128-bit-security TESLA Hash Path and a GNSS system where the satellites are split into three groups. We will consider more or less than three groups later. A slow-channel Hash Point is a 128-bit integer. In the naive case of Figure 5, each of the satellites has its own Hash Path; the time-to-authentication is the Slow-channel Epoch. Figure 5 depicts the slow-channel Hash Point distributions among the three groups out of phase to aid in an IMU strategy, as suggested [10].

In the Interleaved case in Figure 6, the time-to-authentication decreases to one-third of the Slow-channel Epoch time. The entire constellation moves along the slow-channel Hash Path at thrice the naive rate, and therefore, the slow-channel Hash Points expire at thrice the naive rate. However, the distribution burden is shared among three groups of satellites: satellites take turns distributing the slow-channel Hash Point to save bandwidth in the navigation channel. Regardless of which satellite a receiver receives a Hash Point, the receiver can deduce the watermarks for the entire constellation, provided it observes at least one
Figure 6: A conceptual diagram of two strategies for efficient slow channel hash point distribution scheme. Compare with Figure 5. Each circle represents a Hash Point with the arrows depicting the one-way Hash Path relation. On the right, the circles are smaller, depicting how a Hash Point is split into smaller sections and sent in parallel among different satellites split into groups. In these cases, the entire constellation uses a single Hash Path and coordinates distribution across satellites as parallel channels for efficiency to decrease the time to authentication. The bandwidth savings allow the cadence to increase, decreasing time to authentication.

satellite that broadcasts the slow channel Hash Point at that time. In the Sectioned case in Figure 6, the time-to-authentication also decreases to one-third of the Slow-channel Epoch time. However, all satellites distribute a section of a slow-channel Hash Point simultaneously. Split three ways, including 2-bits of metadata bits identifying the particular section distributed by a particular satellite’s data channel, rounding up to the nearest bit, each section will require 45-bits per distribution. The time-to-authentication in both cases decreases because the slow channel Hash Path is shared for the entire constellation.

Careful consideration is required to ensure that a receiver will receive the slow-channel Hash Point at the designed cadence globally. The conceptual diagrams assume three groups of satellites. The greater the number of groups, the smaller the Slow-channel Epoch, the faster time-to-authentication. However, the greater the number of groups, the harder it is to guarantee that a receiver will receive the required distributions at the designed cadence globally. For an Interleaved or a Sectioned distribution scheme to work, the GNSS system must provide a reasonable guarantee that a receiver anywhere will receive all the required distributions to meet the (possibly stochastic) time-to-authentication specification. We propose two strategies for separating the satellites into groups: randomly and geometrically. The random selection below should serve as a worst case, provided a suitable scheme is authored for the geometric case. The following sections discuss the probability of receiver receipt for an Interleaved and Sectioned strategy. Since the Interleaved case provides higher probabilities of receipt than the Sectioned case, we first include the Interleaved case. However, the modeling assumptions (e.g., distributions as independent events) are weaker for the Interleaved case than the Sectioned case. We also note that, given the loss-tolerant feature of TESLA, an unlucky receiver can recover missed Hash Point distributions from later Hash Point distributions.

1. Interleaved Random Distribution
To save bandwidth within the slow channel distribution of a slow channel Hash Point, the constellation coordinates the distribution of the slow-channel Hash Path. For the Interleaved case, satellites periodically take turns distributing the slow channel Hash Point out of phase. Each distribution authenticates proceeding data for the entire constellation.

We consider two models for random selection: replacement and no replacement. The design of the GNSS data channel will likely dictate that each satellite will broadcast information in a fixed, repeating order. For instance, ephemeris information, rekeying information, then slow-channel Hash Point, repeat. Therefore, a particular satellite’s phase does not change in time (unlike in the case of Section III.2). Therefore, the no-replacement case is the more physical case. However, we provide results for the replacement and no replacement cases.

To compute the probability of receiver receipt, we model the problem as independent of Earth geometry, recognizing that this assumption is non-physical if satellite phases do not change in time. A receiver receives the distribution only if it observes at least one satellite in the slow-channel Hash Point distribution phase at that time. Equations (9) and (10) provide the receipt probabilities where $\phi$ is the number of phases, $N$ is the number of satellites in the constellation, and $V$ is the number of satellites in view of the receiver, assuming that at $\phi$ evenly divides $N$. We derive this by inverting the probability that a receiver does not
Figure 7: Comparison of slow-channel Hash Point receipt probabilities at a particular epoch (top numbers) and mean authentication epoch time (bottom numbers) given a variety specified number of phases and assumed satellites in view. By epoch time, we mean the number of epochs. In the left case, we assume that each satellite observed has a random phase without regard to the total number of satellites (replacement case). In the right case, we assume an equal number of satellites in each phase (no replacement case). The right case assumes a RTM satellite constellation. For both cases, the non-physical assumptions include that phases observed in view are independent of earth geometry and that satellites can switch phases spontaneously.

![Insert Figure 7 here]

observe a single satellite in the current Hash Point distribution phase.

\[
P(\text{complete receipt}|\text{interleaved, replacement}) = 1 - \left( \frac{\phi - 1}{\phi} \right)^V
\]

\[
P(\text{complete receipt}|\text{interleaved, no replacement}) = 1 - \left( \frac{N\phi - 1}{\phi} \right) \left( N\phi - 1 \right) \left( \ldots \right) \left( N\phi - 1 - V + 1 \right)
\]

\[
E[\text{Slow Channel Epoch Time of Receipt}] = \frac{1}{\phi} \frac{1}{P(\text{complete receipt})}
\]

Figure 7 provides a table of probabilities of receipt over a single epoch and the mean number of epochs before receiving an entire distribution (“Mean Epoch Time of Receipt”). We note that, given the loss-tolerant nature of the scheme, if an unlucky receiver were not to observe a satellite in the slow channel Hash Point distribution phase, it can still authenticate that epoch at the next epoch when the observed satellites should then be the slow channel Hash Point distribution phase. We will refrain from commenting on the best number of phases used for this work since it will depend on the system requirements.

2. Sectioned Random Distribution

To save bandwidth within the slow channel distribution of a slow channel Hash Point, the constellation coordinates the distribution of the slow-channel Hash Path. For the 128-bit-security-3-satellite-sectioned case, satellites must periodically broadcast 45-bits simultaneously within their data channel. Among the three 45-bit sections of a slow channel, the constellation specifies that a particular satellite will randomly pick its section to broadcast. There are two methods for random selection: replacement or no replacement. Each satellite draws a random section independently of the constellation in the replacement case. In the no-replacement case, the constellation specifies that a fixed number of satellites among the entire constellation will broadcast each group, guaranteed by random shuffling operation. For instance, in a 24-satellite constellation with three groups, at any one time, eight satellites broadcast each of the three groups at random.

To compute the probability of receiver receipt, we model the problem as independent of Earth geometry. A receiver receives the distribution only if each unique section is received. We use the inclusion-exclusion principle to derive Equations (11) and (12), where \( S \) is the number of sections, \( N \) is the number of satellites in the constellations, and \( V \) is the number of satellites in
Figure 8: Comparison of complete receipt probabilities at a particular epoch (top numbers) and mean authentication epoch time (bottom numbers) given a variety of specified sections and assumed satellites in view. In the left case, each satellite draws a random section to distribute (replacement case). In the right case, the constellation draws a random shuffle to guarantee that an equal number of satellites are broadcasting each section (no replacement case). The right case assumes a RTM satellite constellation view of the receiver, assuming that $S$ evenly divides $N$.

$$P(\text{complete receipt}) = 0 \quad \text{if } S > V$$

$$P(\text{complete receipt}|\text{sectioned, replacement}) = 1 + \sum_{i=1}^{S-1} (-1)^i \cdot \left( \frac{S}{S-i} \right) \cdot \left( \frac{S-i}{S} \right)^V$$

$$P(\text{complete receipt}|\text{sectioned, no replacement}) = 1 + \sum_{i=1}^{S-1} (-1)^i \cdot \left( \frac{S}{S-i} \right) \cdot \left( \frac{N \frac{S-i}{S} \cdot (N \frac{S-i}{S} - 1) \cdot \ldots \cdot (N \frac{S-i}{S} - V + 1)}{(N)(N-1)\ldots(N-V+1)} \right)$$

$$E[\text{Slow Channel Epoch Time of Receipt}] = \frac{1}{S} \cdot \frac{1}{P(\text{complete receipt})}$$

Figure 8 provides a table of probabilities of receipt over a single epoch and the mean number of epochs before receiving an entire distribution (“Mean Epoch Time of Receipt”). We note that, given the loss-tolerant nature of the scheme, if an unlucky receiver were not to receive each unique section, it can still authenticate that epoch later once it receives a complete distribution at the next epoch. We will refrain from commenting on the best number of sections that should be used for this work since it will depend on the system requirements.

3. Geometric Selection

Rather than allow satellites to distribute hash points randomly, one could develop a method based on orbital geometry. GPS satellites are approximately divided into six orbital planes. If the satellites were divided into three groups according to opposing orbital planes, like in Figure 9, then one may be able to guarantee global coverage of at least one satellite of each group in view. The solution of Figure 9 would work well for receivers at the poles where the orbital planes form a hexagonal shape with the same groups on opposite sides; however, it may not work well for certain areas in the equator where same groups intersect. One could modify Figure 9’s group-to-plane allocation at lower latitudes to accommodate those areas of the globe. For instance, satellites could change groups each time they pass through an intersection of another plane, or satellites could follow a different group-to-plane allocation function at lower latitudes. Or satellites could rotate their groups each epoch to ensure that all receivers authenticated at three times the nominal cadence. For the real GPS constellation, only simulation would verify global coverage of at least one satellite in the view from each group globally.

4. Loose-time Synchronization Considerations

A critical security assumption for TESLA is loose-time synchronization. If a receiver clock is delayed, a receiver may accept messages signed with hash points after the provider has delay-released them. If that were the case, then a receiver could accept
spoofed messages. Therefore, the entire protocol assumes that a receiver’s clock delay is below a threshold agreed to by the scheme. We note that these considerations and vulnerability would be present in any ECDSA-only scheme and claim that it is better to use TESLA since the requirements are explicitly addressed in a TESLA scheme, and TESLA provides the additional features described in previous sections of this work.

Computers use Network Time Protocol to synchronize clocks, which has now been extended to include security features (“NTS”)[11]. Many GNSS users must remain isolated from the internet, posing a catch-22 to this protocol because the signal itself provides the time. Any NTS and GNSS system remain vulnerable to a delay attack where a spoofer delays the signals to delay a receiver clock into breaking TESLA security. By incorporating an accurate receiver clock, this risk can be partially mitigated. For this discussion’s concreteness, let us consider the civilian aircraft stakeholder, whose GNSS receivers currently are isolated from internet connection. An aircraft can observe the GNSS-deduced timing measurement and the timing provided by an onboard (assumed untampered) clock. If the GNSS time estimate spontaneously delays, the aircraft should not trust the GNSS signal and inspect its clock. This mitigation does not work for a creeping delay attack that spoofs the time slowly undetected under the drift rate of the onboard clock. However, these concerns can also be mitigated with routine aircraft maintenance or time-synchronization in a safe zone [12]. For instance, a periodic maintenance schedule or advanced spoofing detection methods implemented at airports could certify that an airplane’s clock is not delayed at take-off. Receiver manufacturers can ensure the clock drift rate is low enough to provide security until the next synchronization check. Therefore, the scheme hinges on the clock drift rate for a time-isolated flight. We defer further discussion on the ability to provide time synchronization to other work and continue to assume the loose time-synchronization assumption in writing this scheme.

The conceptual diagrams of this work show the relative release schedule of information: watermarked-spreading code and navigation message, and then Hash Point. However, if receiver clock drift rates are not ideal, Hash Point distributions can be delayed further. Delaying Hash Point distributions increases the required time and effort for a spoofer to creepily delay a receiver time, making the attack more difficult. For the design of a TESLA scheme for SBAS [4][13]. Hash Points are not released immediately after use; rather, they are released one extra epoch later. This makes the required delay 6 seconds before the SBAS TESLA scheme breaks. We now consider how to incorporate this reasoning in this scheme.

Since we suggest combining the slow and fast channels, the time synchronization delay-bound requirements must be the same for slow and fast channel receivers, albeit a fast channel receiver can have assisted access with a network. To accommodate the expected clock drift rates of all receivers, one need only delay both slow and fast channel distributions simultaneously, as conceptually provided in Figure[10]. For the sake of clarity, to make this change to Figure[5] all of the Hash Points of the slow and fast channel would shift right while all other objects remained in the same place, increasing the length of all the arrows. We note that this accommodation does not affect the cadence of slow and fast channel users but increases the time-to-authentication delay for all users. A spoofer cannot listen to the fast channel distribution, derive the watermarks, and spoof a slow channel.
All slow channel receivers should have already received the forgeable information before receipt of the fast-channel Hash Point by a spoof. We believe that it is reasonable to expect that slow-channel receivers have hardware clocks at least as good as (but more likely better) fast channel receivers since commodity hardware provides clocks with low drift rates for this application.

IV. OUT-OF-PHASE REKEYING DISTRIBUTIONS

For the rekeying problem, strict adherence to the TESLA loose time-synchronization schedule is not required. The rekeying problem determines the time-of-first-authenticated fix for a cold start receiver. A cold start receiver needs to receive all the levels of ECDSA keys and the associated signatures and the TESLA Hash Path End and the associated ECDSA signature. As suggested for SBAS by geostationary satellite and frequency band [T], distributing sections of rekeying information for GNSS out-of-phase among the parallel satellite data channels will decrease the time-of-first-authenticated-fix for cold start receivers according to the number of phases.

V. SIMPLE AND EFFICIENT WATERMARKING FUNCTION

In this section, we suggest a simple and efficient function for Equation (8), the “Watermarking Function”. By simple, we mean that the function is simple in its construction. By efficient, we mean that it has linear complexity while used in its context. The Watermarking Function takes a cryptographic seed and deterministically watermarks the spreading code. As depicted in Figure 5, the seed is a Hash Point forked from the fast-channel Hash Path, meaning that the cryptographic seed is just a hash point, an n-bit integer.

We require that any watermark degrade the spreading code the same amount. Requiring the same degradation aids with security since a receiver should expect the same (not-stochastic) autocorrelation bump after receiving every watermark seed. We suggest that the Watermarking Function flip a set number of chips within the spreading code to implement this. Therefore, the Watermarking function takes a Hash Point and then deterministically flips a fixed number of chips of the spreading code. To enforce that the same number of chips are flipped for each watermark, we have the Watermarking Function deterministically select a combination (with the same number of selections) of chips from a Hash Point.

Let the particular combination derive from a cryptographic random integer, according to Algorithm [1]. Let the spreading code be divided into \( n \) sections, each with an equal number of chips. Let the number of sections to flip per watermark be \( m \). The degradation will be twice \( \frac{m}{n} \). We must assert that \( \log_2 C(n, m) > 128 \), where \( C \) is the combination function; otherwise, there will be too few unique watermarks to ensure 128-bit security. To compute the specific chip sections to flip, we use Algorithm [1]. For example, we suggest the following parameters to provide about 10% degradation for the L1C and C/A signals. For L1C with 10230 chips, one could divide the spreading code into 465 sections of 22 chips each, then randomly select 23 of them according to Algorithm [1]. This provides sufficient entropy with \( \log_2 C(465, 23) > 128 \) and about 10% degradation. For C/A with 1023 chips, discard 223 chips as never going to be modified by a watermark, divide the remaining spreading code into 400 sections of 2 chips each, then randomly select 25 of them according to Algorithm [1]. This provides sufficient entropy with \( \log_2 C(400, 25) > 128 \) and about 10% degradation. If a particular section is selected, the spreading code section is flipped.
Algorithm 1 is inspired by a clever trick suggested by Robert Floyd [1T]. Robert Floyd provided an efficient algorithm to select a random combination uniformly. Algorithm 1 modifies Robert Floyd’s suggestion by replacing the random selection with a semantically random selection that is deterministic from a Hash Point. We replace the randomness of Floyd’s algorithm with several modulo of the cryptographic input seed, making the selection deterministic. Since the input Hash Point should be distributed uniformly among the Hash Point domain, given the security of the hash function, all the modulo should also be uniformly random. Moreover, since Robert Floyd’s algorithm selects a combination uniformly, sections will be flipped uniformly. Algorithm 1 is simple because of its simple definition involving a repeated modulo operation and is efficient because of its linear time complexity. We add a random salt \( s \) to the operation to make inverting the watermarking function more difficult.

**Algorithm 1:** Derive a random spreading code watermark from a cryptographic seed.

Let \( k \in [0, 1, \ldots, 2^{128} - 1] \) be a semantic random number that derives from a secure cryptographic seed, such as an unreleased TESLA Hash Point.

Let \( n \) be the total number of spreading code sections that can serve in the watermark.

Let \( m \) be the total number of spreading code sections that must be selected for all watermarks.

We assert that \( \log_2 C(n, m) > 128 \) to ensure minimum required entropy for security.

Let \( s \in [0, 1, \ldots, 2^{128} - 1] \) be a salt.

Let \( S \) be a set data structure.

\[
\text{for } j = [n - m + 1, \ldots, n] \text{ do} \\
\quad t = (k \oplus s) \mod j \\
\quad \text{if } t \notin S \text{ then} \\
\quad \quad S \leftarrow t \\
\quad \text{else} \\
\quad \quad S \leftarrow j \\
\text{end}
\]

Flip the spreading code sections indexed by those in \( S \) to watermark the spreading code.

VI. CONCLUSION

With this work, we provide suggestions on how to greedily exploit the advantages of TESLA for GNSS watermarked spreading authentication schemes, such as Chimera. The additional features discussed include the following. First, a multi cadence watermark distribution with a single watermark degradation. Second, we minimize the slow-channel distribution while allowing navigation message authentication and spreading code watermarking. Third, we discuss how to exploit parallel data distribution among the constellation. Last, we discuss a watermarking function. With the design concepts of this work, we could minimize the bandwidth burden for authenticating GNSS signals via watermarked spreading codes.

**REFERENCES**


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