Cryptographic Ranging Authentication with TESLA, Rapid Re-keying, and a PRF

Jason Anderson, Sherman Lo, Todd Walter

Abstract

This work examines cryptographic design principles for next-generation GNSS signals that could provide a publicly authenticated ranging signal of comparable security to current encrypted signals. We discuss how any authentication of spreading codes must act as an effective bit-commitment authentication; therefore, we advocate and greedily apply Timed Efficient Stream Loss-tolerant Authentication ("TESLA") to cryptography-first GNSS design. Since any authentication acts as bit-commitment authentication, greedily using TESLA provides additional features regarding bandwidth efficiency and loss-tolerances relevant to GNSS. Using those design principles, we suggest a new method to generate secure spreading codes and distribute the required cryptographic seeds as a case study on how a cryptography-first design methodology would guide the design of a ranging signal. Moreover, we suggest an alternative publicly authenticated signal achievable by merely modifying the re-keying procedure of existing symmetrically encrypted signals (e.g., GPS's P(Y)-code, Galileo's E6B/C signal). This modification would maintain the current, real-time secure use of current encrypted ranging signals while providing a critical infrastructure needed by aviation and autonomous vehicle stakeholders. We compare this method to other publicly authenticated ranging signals and make the case that our suggestion would be easier and faster to achieve because it requires no changes to existing signals. Finally, we suggest that future GNSS systems modularly separate signal purposes. One signal, or signals, could provide the best possible real-time unauthenticated service, unencumbered by applying cryptography. One signal could provide the best possible delayed cryptographic spoofing detection service, unencumbered by existing requirements of real-time signals.

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.

GNSSs generally offer publically open and symmetrically secret signals, such as GPS C/A and P(Y), respectively. The spreading code of open signals is known and repeating, whereas with symmetrically secret signals, the known and repeating spreading code is encrypted symmetrically with a cipher. Only certain holders of the cipher's symmetric key ("Privileged Users") can use a confidential signal. Open signals currently offer no protection against forgeries of the spreading code, and a few newer signals provide authentication for the data channel, such as Galileo's OSNMA [1, 2, 3]. The secret symmetric signals offer some protection against forgery, but they cannot function as an open public service, and the symmetric design is vulnerable to key leakage. This work attempts to provide the protections afforded to Privileged Users to all other users and remove the current key-leakage vulnerability. Others have suggested augmenting public signals with unpredictable, pseudorandom watermarks to the signal [4, 5, 6], correlating with the unknown encrypted signals or distributing encrypted spreading code sections with a side-channel [7, 8] or adding unpredictable sections to the spreading code[9]. This work considers what a completely new, cryptography-first GNSS concept would look like. Starting from a clean slate allows a design that is efficient, flexible, and scalable with cryptography. Many of the principles of our design are inspired by previous work authenticating a Satellite-based Augmentation System ("SBAS")[10].

2. Limited Authentication Security

Modern cryptographic authentication certifies information as unmodified or forged while in transit from the authenticated provider to the receiver. The certification relies on arguments of computational work. If a cryptographic authentication scheme is secure, only the authenticated provider holding secret private keys could have generated the authenticated information. An adversary without the secret private keys would need inordinate computational resources to guess exhaustively the secret private keys to create forgeries. In GNSS, the information delivered includes navigation message information and ranging signals. Authenticating navigation messages with cryptography requires appending message authentication codes; in current systems, the main challenge is accommodating the additional bandwidth of the message authentication codes and associated rekeying maintenance. Authenticating ranging signals with cryptography is still an open area of research without a clear-cut solution.

The challenge of ranging authentication arises from how the signal arrival time at the receiver provides the navigation and timing solution. Signal arrival times are vulnerable to delay attacks even when augmented with cryptography. GNSS ranging signals are known, repeating spreading codes (encrypted with a symmetric key in closed signals). The selected repeating (sometimes encrypted) spreading codes are sufficiently pseudorandom for autocorrelation to determine pseudoranges. An adversary can forge that repeating unencrypted spreading codes because they are public knowledge. Assuming that provider and receiver are perfectly time-synchronized and that a receiver would reject late signals, an adversary can only forge the signal by predicting the cipher. However, GNSS provides the time for the receiver, creating a catch-22 dilemma for a GNSS-only user. Adversaries can manipulate the ranging signal by perturbing the ranging signal arrival times without modifying the content of the signal. For instance, an adversary can listen and rebroadcast the signal. Cryptography has not yet provided security against delay attacks that rely on computational work arguments[11, 12].

In this work, we consider a similar argument to computational work to establish ranging signal authentication. Since GNSS signals broadcast below the thermal noise floor, we consider authentication security arguments based on an adversary's signal processing ability. Like cryptography asserts authenticity based on the infeasible computational work required by an adversary, we consider authenticity assertions based on an adversary's ability to observe below-thermal-noise signals, manipulate, and rebroadcast them.

Consider the encrypted GPS P(Y) signal. Assuming the P(Y) cipher is secure, an adversary of concern to civil users cannot forge ranging signals unless they engage in a delay attack with a repeater. Without specialized expertise and exorbitant antennae hardware, and parallel computing resources, an adversary can only (1) delay all satellite signals simultaneously, causing a receiver's clock to delay or (2) fool a receiver to be at the repeater's position because the encrypted signals cannot be manipulated *individually* via signal processing. Ordinary adversaries of concern to civil contexts cannot individually delay satellite signals allowing forgery to an arbitrary receiver position. We allow the difficulty of individually processing and replaying encrypted, below-thermal noise spreading codes to form an argument of infeasible signal processing ability for an ordinary adversary. We will call this Repeater-limited Authentication Security because only listening and rebroadcasting the whole service together (without individual signals perturbed) are feasible by adversaries of concern to civil signals. Recognizing that GNSS signals provide timing, if the receiver has an externally secure, accurate time, the signal is also unforgeable under Repeater-limited Authentication Security because a receiver would expect each pseudorange signal at the correct time. This work hopes to extend this level of security from Privileged Users to everyone.

Consider the public GPS C/A signal. If the data channel were augmented with cryptographic pseudorandom bits, the signal is protected against forgery at a weaker level than Repeater-limited Authentication Security while those pseudorandom bits are broadcast. Since the spreading codes would be known, an ordinary adversary could engage a replay attack with the ability to *separate* the individual satellite signals allowing an adversary to manipulate a receiver position and time almost arbitrarily. An ordinary adversary engaged in a replay attack in this context would be sophisticated since they would need to listen and incorporate the pseudorandom bits in the spoofed ranging signal, so we again allow the difficulty of this attack to form an authentication security argument called Replay-limited Authentication Security. We call it Replay-limited Authentication Security since only an ordinary adversary engaged in a replay attack could forge the ranging signals.

This work will consider cryptography-first GNSS design methodology to achieve Repeater-limited Authentication Security or Replay-Limited Authentication Security for open signals. Practically, the designs of this work allow encrypted signals to provide authentication to open signals at the same level as current encrypted signals to Privileged Users.

3. 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[13]. 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 [10].

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. Herein, we call that input Hash Point the preimage Hash Point of that 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.

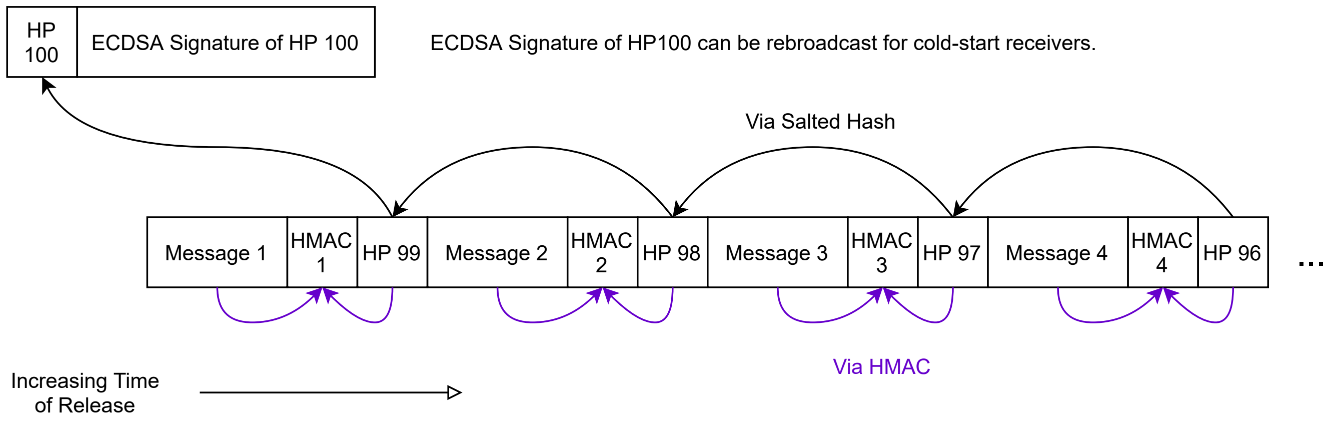
HP 100
ECDSA Signature of HP 100
ECDSA Signature of HP100 can be rebroadcast for cold-start receivers.
Via Salted Hash
Message 1
HMAC 1
HP 99
Message 2
HMAC 2
HP 98
Message 3
HMAC 3
HP 97
Message 4
HMAC 4
HP 96
...
Via HMAC
Increasing Time of Release

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[13]. 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.

II. TESLA AND SATELLITE NAVIGATION

Since GNSS signals contain three layers: carrier wave, spreading code, and data channel, one could augment the latter two with cryptography. This section considers how one could apply TESLA naively to those latter two. First, Section II.1 discusses the construction of Hash Paths relevant to satellite navigation systems. Second, Section II.2 discusses TESLA's application to the data channel. Third, Section II.3 discusses TESLA's application to spreading codes. Fourth, Section II.4 discusses how to scale TESLA to authenticate unbounded information. While the following sections represent the basic application of TESLA to consider security, Section III synthesizes them to build suggested signals.

Where we write AES-CTR, we mean AES in counter mode. AES-CTR will only be used as a pseudorandom function generator to provide a random spreading code for ranging, which is secure since it will not be used to send information confidentially. One could replace AES-CTR according to stakeholder feedback that considers receivers' speed and power efficiency. Where we write AES-GCM, we mean AES in Galois Counter Mode to provide authenticated encryption security on the applicable contents. One could replace AES-GCM, considering receivers' speed and power efficiency according to stakeholder feedback. We specify AES in this work because, at the time of writing, AES is standardized, widely used, and well-researched to aid with the scheme's security. Moreover, there are many efficient hardware chip accelerations in use. We use our selections explicitly to make this work more concrete, concise, and intelligible.

In the following sections we will describe a TESLA-based framework for all keys to be securely derived from a single distribution to aid the bandwidth-limited GNSS context. We took special care to ensure that all derived keys are cryptographically independent and related by a one-way function whose preimages are released after authenticated information's applicable expiration.

1. Hash Path Construction

In this section, we discuss the construction of Hash Paths relevant to satellite navigation. Let P be a particular Hash Path composed of many Hash Points p_i indexed by i. The Hash Path Start p_0^P 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 release of the i-th Hash

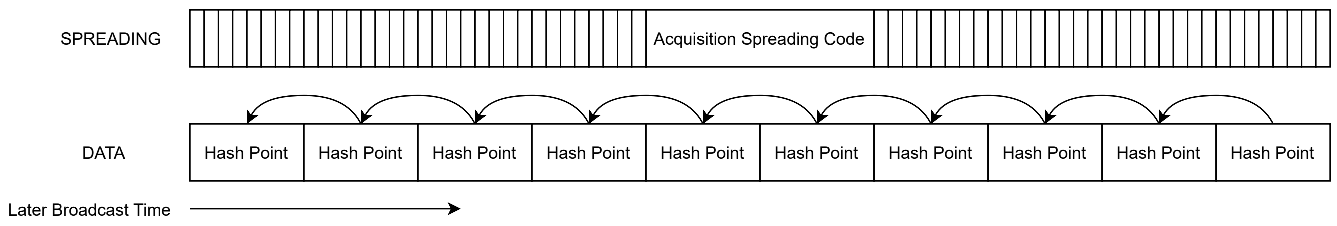

Figure 2: A conceptual diagram of signal applying TESLA to data channel. The diagram reads from left to right by increasing the time of release. The arrows to and from each Hash Path correspond to a hashing operation of Equation (1).

Point. With n as the desired security level and N the length of the Hash Path, we provide Equation (1) to define a Hash Path.

$$\begin{aligned} p_0^P &\xleftarrow{R} \{0,1\}^{\mathrm{n}} \\ p_{i+1}^P &= \mathrm{H}\left(p_i^P || S^P || t_i\right) \quad \forall i \in \{0,1,...,N-1\} \end{aligned} \tag{1}$$

Equation (1) concatenates 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 [14]. In summary, the Hash Path needs a counter and a salt.

Equation (1) uses time as the counter to simplify the protocol and aid in detecting receiver and provider 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 of any signal, rounded down to the nearest integer, should be consistent since the signal flight time is less than one second. Our time counter saves data bandwidth by eliminating the need for the provider to send additional meta-data about the length, start, and end of Hash Paths. With the hashing property of Hash Points, upon receipt of a new Hash Point, a receiver can check whether the new Hash Point hashes to a new Hash Path End signed via ECDSA at constant time complexity, further reducing rekeying metadata needs[10]. Our time-counter aids with the security by making the scheme brittle to time synchronization errors. If the receiver clock is off by more than a second, the incoming Hash Points will not be verified, notifying the user.

2. TESLA and the Data Channel

We now consider TESLA application to the data channel. According to TESLA, data will be authenticated after its broadcast with the delayed release of Hash Points.

Providers provide navigation data within the data channel in the present GNSS system. So, for the data channel, one could use the message stream of Figure 1, where the messages deliver that navigation data. Providers need to include additional information in that message stream to complete the TESLA authentication protocol: (1) the Hash Path End with its ECDSA signature, and (2) any other rekeying maintenance information. TESLA's application to the data channel would provide authentication on the data itself only. We note that all proceeding information is authenticated upon receipt of the applicable Hash Point. This means that more information within the message stream increases the authentication delay of the data. Moreover, this scheme cannot provide authentication for the pseudoranges when the data channel is not broadcasting a Hash Point because there would be no cryptographic pseudo-random component to the signal.

While the data channel transmits a Hash Point, the spreading code is xor-ed with a PRF (the concatenated Hash Path is a PRF) that no adversary can efficiently predict. This means each pseudorange accompanies pseudorandom bits to provide Replay-limited Authentication Security. The navigation data is slowly varying: an adversary could predict the other sections of the data channel and then forge pseudoranges intermittently. Therefore, in our final consideration of applying TESLA to the data channel with Figure 2, we do not include any non-PRF data in the data channel to provide the best possible authentication security. Moreover, we do not weaken the scheme by allowing multiple satellites data channels to share the same Hash Path. Additional metadata may be needed to surround the Hash Points to allow a satellite to acquire the signal, which we refrain from discussing in this work. A receiver can partially mitigate replay attacks with an accurate clock by detecting spontaneous jumps in time. However, a sophisticated spoofer could still spoof the position without affecting time or engage in a creeping replay attack below the uncertainty of the onboard clock.

3. TESLA and GNSS Spreading Codes

We now consider TESLA application to the spreading codes. According to TESLA, signal acquisition and tracking can happen only after the broadcast with the release of Hash Points.

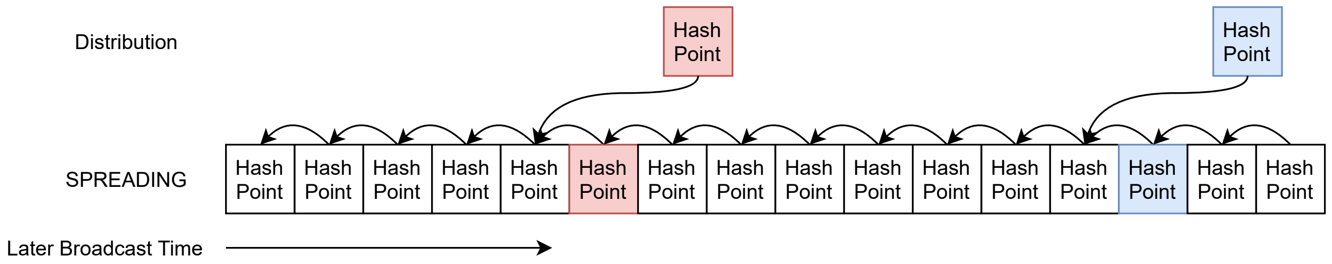

Figure 3: A conceptual diagram of a secure spreading code derived via TESLA. The spreading code is generated from a Hash Path. Some Hash Points are distributed out-of-band to users after their use. The user uses those distributions to derive the rest of the spreading code. If a receiver loses one of those out-of-band distributions, it can still derive the entire spreading code from a later distribution. In the diagram, the red and blue Hash Points are the same. Their distribution always comes after using the spreading code to maintain security.

Since one can construct a PRF from a hash function, and PRFs provide sufficient randomness to serve as spreading codes, one could naively use hashes as the spreading codes. A naive TESLA-based implementation would be to use the Hash Path as the spreading codes, as shown in Figure 3. The provider could broadcast a subset of the Hash Points for receivers to derive the rest of the spreading code. This design is loss-tolerant: if a receiver misses a Hash Point, it can still recover the spreading code from the next Hash Point. This design is somewhat efficient: the receiver receives a minimum cryptographic seed to generate swaths of spreading code. However, the spreading code generation is not parallelizable and may not be the most computational and receiver-power efficient. To achieve a better, more flexible design, we suggest having Hash Points serve as a PRF seed.

Let $S_{i,\text{SVN},s}$ be the spreading code derived from a particular Hash Point p_i^P, where s refers to the spreading code subsection index. We suggest Equation (2).

$$S_{i,\text{SVN},s} = \text{PRF}\left(p_i^P, t_i||\text{SVN}||s\right) \tag{2}$$

Equation (2) allows the minimal distribution of Hash Point p_i^P to seed an arbitrary amount of spreading code. Depending on stakeholder feedback that pertains to the computational and power efficiency, the PRF could be either HMAC or AES-CTR, as in Equations (3) and (4).

$$\text{PRF}(\cdot,\cdot) = \text{H}\left(\text{HMAC}(\cdot,\cdot)\right) \tag{3}$$
$$\text{PRF}(\cdot,\cdot) = \text{AES-CTR}(\cdot,\cdot) \tag{4}$$

With Equation (3), HMAC serves as a secure key derivation function, securely forking the Hash Path to provide additional secure keys, which we use again in Section II.4. The additional Hash operation is required to mitigate related-key attacks on HMAC. With Equation (4), AES-CTR turns AES into a PRF. Since we do not use AES-CTR as a stream cipher, just as a PRF for ranging, AES-CTR is sufficient. We specify counter mode to allow parallelization and best possible speed and computational efficiency.

Both HMAC and AES-CTR allow parallelizing the spreading code generation. Using p_i^P, a receiver can generate all the derived sections of spreading code, indexed by s, in parallel. This is particularly important if the satellite constellation shares a single Hash Path, like as conceptually depicted in Figure 4 with blue arrows. If a constellation shares a Hash Path, the contextual field of the PRF must include satellite information to ensure all the derived spreading codes are unique, as done in Equation (2) with SVN, which stands for Satellite Vehicle Number. We extend the security argument of Section II.1 to include t_i in Equation (2).

4. Scaling TESLA

TESLA's efficient bandwidth use for authentication is the most relevant property to satellite navigation systems. This property can be greedily exploited to authenticate an unbounded amount of information with loss-tolerance. To enact this, we fork the Hash Path with a secure key derivation function, such as HMAC. We provide a conceptual diagram with Figure 5. In Figure 5, we show how to minimize the cryptographic distribution to a single Hash Point while authenticating navigation data, distributing decryption keys for encrypted navigation data, and distributing seeds for spreading code generation for multiple satellites. Additional information can be authenticated to a TESLA scheme by additionally forking the Hash Path.

Data channels on top of PRF-spreading codes must be encrypted, so all layers behave as PRFs for security. A secure key derivation function should be used to encrypt sections of data separately to aid with the loss tolerance of the distribution of encrypted navigation data. If a particular section of encrypted data is lost, it will not affect other sections' decryption (and authentication).

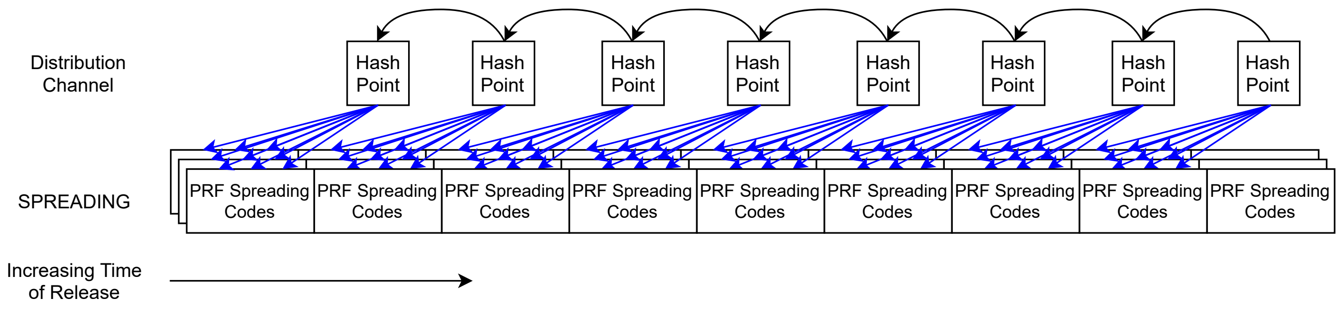

Figure 4: A conceptual diagram of signal of using TESLAd-derived seeds for a PRF for spreading codes. The diagram reads from left to right by increasing the time of release. The black arrows represent a normal hashing operation of Equation (1). The blue arrows from a Hash Point to the spreading code represent Equation (2). The spreading code must be broadcast before the Hash Point generator is released to maintain Repeater-limited Authentication Security. The Hash Path can be distributed securely via an internet-based side channel or in the data layer of any other signal (e.g., in-quadrature or acquisition signal).

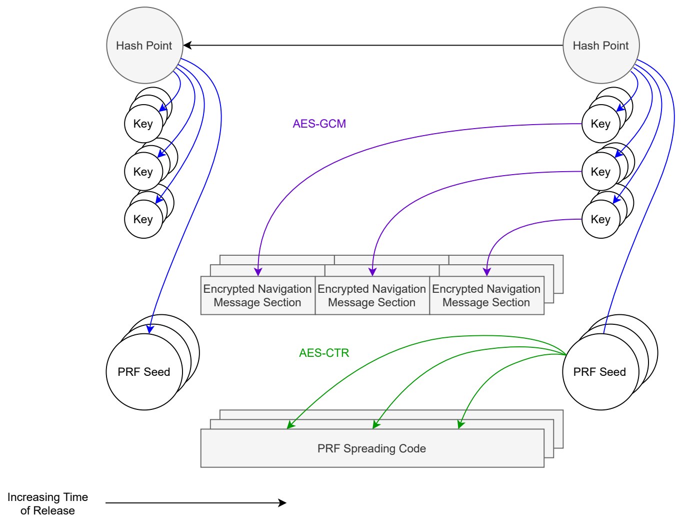
Hash Point
Hash Point
Key
Key
Key
Key
Key
Key
AES-GCM
Encrypted Navigation Message Section
Encrypted Navigation Message Section
Encrypted Navigation Message Section
AES-CTR
PRF Seed
PRF Seed
PRF Spreading Code
Increasing Time of Release

Figure 5: A conceptual diagram of how to use HMAC as a key-derivation function to authenticate an unbounded amount of information with a minimal Hash Point distribution. The diagram reads from left to right by increasing the time of release. The diagram objects in grey are broadcast via the navigation system: either the data channel or spreading code; the derived keys and PRF seeds from the Hash Point are not. The black arrow represents a regular hashing operation, such as Equation (1); the blue arrows, an HMAC operation as a secure key derivation function; the green arrows, deriving spreading codes, such as Equations (3) and (4); the purple arrows, secure encryption by a cipher that provides authentication security. Some objects (i.e., everything except Hash Points) depict overlaid repetitions, representing that a single Hash Point distribution could derive all cryptography data needed for multiple satellites.

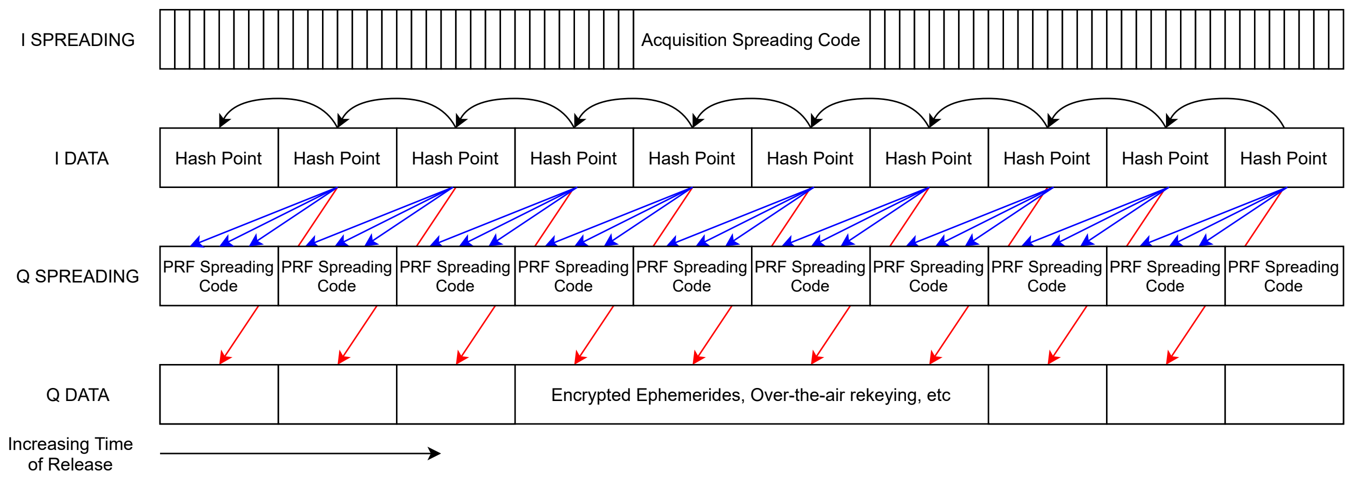
I SPREADING
Acquisition Spreading Code
I DATA
Hash Point
Q SPREADING
PRF Spreading Code
Q DATA
Encrypted Ephemerides, Over-the-air rekeying, etc
Increasing Time of Release

Figure 6: A conceptual diagram of a dual-channel service; for instance, in quadrature (in-phase is "I" and in-quadrature is "Q"). The diagram reads from left to right with increasing release time by provider. Top to bottom is in order of use by a receiver. First, a receiver uses the known spreading code of the I channel as an acquisition signal. The receiver records raw frequency radio data for the length of the Hash Point cadence. The I data channel provides the delayed cryptography seeds for a receiver to generate the Q spreading code replica for Repeater-Limited-Authentication-Secure ranging with the recorded raw frequency. A receiver will use the I channel real-time positioning and timing and use the Q channel for authenticated, delayed positioning and timing as required by its security requirements. The black arrows refer to a normal hash operation, like Equation (1), the blue arrows refer to the parallelizable HMAC/PRF operation, like Equation (2), and the red arrows refer to the encryption operation on navigation data.

For all services, the time to authentication is bounded below by the cadence of the Hash Path distribution. At the time of this work, navigation systems are extremely data bandwidth-limited, hence the motivation to apply TESLA. In a future system, bandwidth may not be of concern, but this methodology should still be applied to maximize the cadence of the distribution. The faster the Hash Points are rekeyed in the system, the faster the authenticated signal can be used for authenticated navigation solutions. So a new system should attempt to rekey itself as fast as possible to minimize the time to authentication, provided the receiver clocks are accurate enough to maintain the loose-time synchronization assumption.

III. SUGGESTED NEW SIGNALS

Whereas Section II delineated several design ideas and considerations for applying TESLA to navigation systems, this Section III synthesizes those principles and considerations to suggest new signals. Section III.1 considers a two-phase signal in quadrature, inspired by the C/A and P(Y) signal provided by GPS today. Section III.2 considers a single phase signal. Section III.3 considers how to modify the rekeying procedure of current encrypted services for Privileged Users to provide a public authenticated service without modifying the signal.

1. Dual-channel Service

We proposed a two-phase signal in quadrature, inspired by the C/A and P(Y) signal provided by GPS today, with Figure 6. The signal contains an In-phase ("I") signal and an In-quadrature ("Q") signal, although the signal does not necessarily have to utilize the same frequency. The I signal uses a spreading code designed like modern public GPS C/A as an *acquisition* code. The I data channel exclusively contains a Hash Path (with any necessary repeated metadata for synchronization purposes). The I channel provides *real-time* pseudoranges under Replay-limited Authentication Security. The Q signal uses PRF spreading codes derived from the I data channel Hash Points, which are always released after expiry. The Q data channel exclusively contains all navigation and rekeying data encrypted by a key derived from the I data channel Hash Points. The Q channel provides *delayed* pseudoranges under Repeater-limited Authentication Security, and the navigation data is protected with standard authenticated encryption.

A receiver does the following to acquire a full navigation solution. First, using the publicly known replica of the acquisition spreading code, the receiver acquires and tracks the ranging signal provided by the I signal. The receiver must begin to store raw radio data of the Q signal. Second, after receiving a complete Hash Point from the I data channel, a receiver derives the PRF-spreading codes and processes the stored raw radio data of the Q signal. The receiver can process Q signals in the past, computing Repeater-limited Authenticated pseudoranges and decrypting the navigation data of the Q channel. Third, the receiver performs an ECDSA authentication of the Hash Path End to anchor the TESLA Hash Path security. Fourth, the receiver now

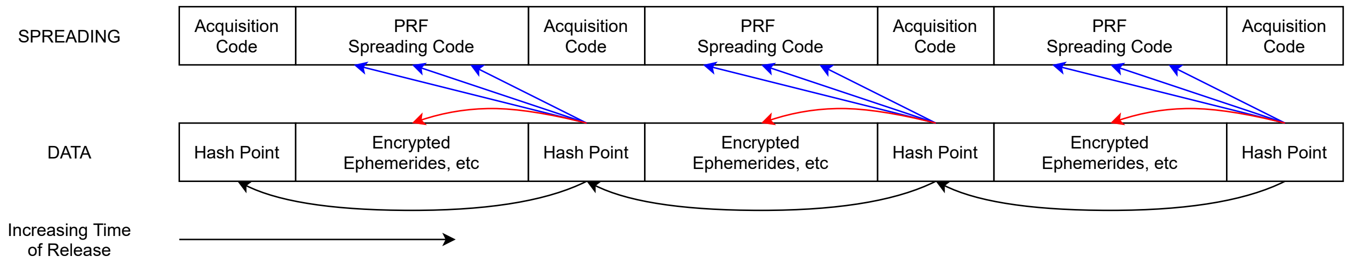

Figure 7: A conceptual diagram of a single channel service. The diagram reads from left to right with increasing release time by provider. First, a receiver uses a known spreading code as an acquisition signal that occurs at a lower frequency, which is authenticated via TESLA on the data channel to Replay-limited Authenticated Security. A receiver that would like to either (1) have a Delay-limited Authentication Secured pseudorange or (2) receive authenticated navigation data would need to record raw frequency data for the length of the delay. The data channel provides the delayed cryptography seeds for a receiver to generate the remaining replica for PRF ranging with the recorded raw frequency. A receiver will use the lower frequency ranging for the course, real-time positioning, and timing solution and generate the rest of the spreading code for a delayed, precision positioning and timing solution as required by its safety-of-life standards. The black arrows refer to a regular hash operation, like Equation (1), and the blow arrows refer to the parallelizable HMAC operation, like Equation (2).

uses the TESLA Hash Path to authenticate the navigation data under standard cryptographic security. The I signal pseudoranges provide Replay-limited Authentication Security, and Q signal pseudoranges provide Repeater-limited Authentication Security to achieve an authenticated navigation solution.

Since the navigation data slowly varies and a Hash Path rotates at a slow cadence, the receiver does not have to continue decrypting Q data. Using the navigation data acquired during the first position fix, the receiver can ignore the Q signal entirely to maintain Replay-limited Authentication Security. However, the receiver need not continue processing the entire Q signal to maintain Repeater-limited Authentication Security. Instead, a receiver could only check small sections of the PRF spreading code, decreasing the required continuous raw radio processing. A receiver can access the Q signal according to the requirements of their use case context.

By separating the intentions of each signal (e.g., acquisition and Hash Point distribution versus secure ranging and navigation data distribution), the design is flexible in its construction to stakeholder design considerations and uses. For instance, the time-to-authentication is bounded below by the cadence that the PRF spreading code changes. The maximum raw radio storage needed also depends on the cadence of the PRF spreading code changes. Depending on that cadence and periods between PRF-seed change, a receiver could select to observe subsections or at a lower frequency and still maintain Repeater-limited Authentication. Moreover, a signal of this construction could authenticate other ranging services (see Section IV).

2. Single Channel Service

We propose a signal that combines the dual service of Section III.1 into a single signal with Figure 7. Essentially, we take the in-quadrature signals from Section III.1 and interleave them. The spreading code alternates between a publicly known *acquisition* code and a PRF spreading code. When the spreading code is in the acquisition section, the data channel exclusively contains a Hash Point. When the spreading code is in the PRF section, the data channel exclusively contains all navigation and rekeying data encrypted by a key derived by a later-released Hash Point. The seeds for the PRF spreading code sections derive from the Hash Points intermittently dispersed during the acquisition spreading code sections. The seeds are always distributed after expiry after the PRF code has been broadcast. Since we call the publicly known spreading code sections *acquisition* codes, we suggest that most of the spreading code is a delay-released PRF, leaving only small sections at a slower cadence to allow receiver acquisition.

To acquire a full navigation solution, a receiver does the following. First, using the publicly known replica of the Acquisition Spreading Code, the receiver acquires and tracks the signal at a slow cadence. The receiver must begin to store raw radio data of the signal. Second, after receiving a complete Hash Point, a receiver derives the PRF-spreading codes and processes the stored raw radio data. The receiver is able to process PRF spreading code from the past, computing temporarily insecure pseudoranges, and decrypting the encrypted data channel. Third, the receiver performs an ECDSA authentication of the Hash Path End to anchor the TESLA Hash Path security. Fourth, the receiver now uses the TESLA Hash Path to authenticate the navigation data to normal cryptographic authentication security. The acquisition pseudoranges provide Replay-limited Authentication Security, and the PRF pseudoranges provide Repeater-limited Authentication Security to achieve an authenticated navigation solution.

Since the navigation data is slowly varying and Hash Paths rotates at a slow cadence, the receiver does not have to continue processing the PRF spreading codes and encrypted data channel. However, the receiver will compute a solution at the acquisition

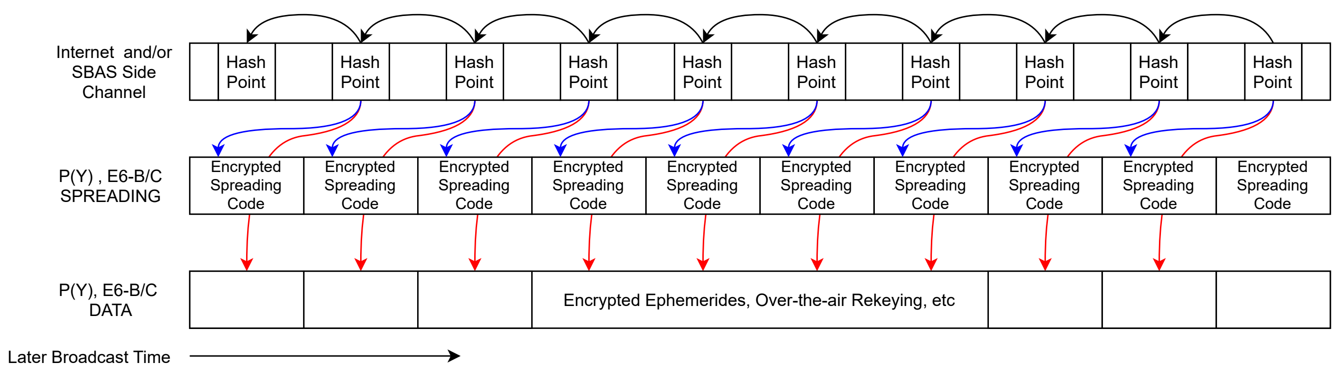

Figure 8: Diagram of how to apply TESLA to current encrypted spreading code signal to provide an open Repeater-limited-Authentication secure ranging service. The diagram reads from left to right with increasing release time by provider. Rather than have independent keys, the key relate via the Hash Path Equation (1), represented by the arrows to and from Hash Paths. The arrows from the Hash Point to the encrypted spreading codes represent the encrypted spreading code with the applicable, ephemeral Hash Point key. Restricted real-time access is preserved for Privileged Users secure distribution and storage of the Hash Path Start. Delayed-time access is allowable with a side-channel (e.g., internet, SBAS). A key derivation function, such as Equation (5), allows for multiple satellites to use the same seed.

cadence. To compute position solutions at the designed cadence, the receiver will need to process the raw radio data to process the PRF pseudoranges.

This signal has many flexibility traits relevant to different stakeholders as discussed in Section III.1.

3. Modifying GPS P(Y) and Galileo E6-B/C

While this paper focuses on how a new GNSS system could be designed via a cryptography-first methodology, we suggest a comparable service with current signals such as the GPS P(Y) and Galileo E6-B/C or any other encrypted signal that serves Privileged Users, as depicted with Figure 8. We propose a minor modification where Privileged Users retain their secure, real-time encrypted use but civilian stakeholders gain a delayed, secure-ranging signal. The modification only pertains to key management. The cipher nor the ranging algorithms need change, ensuring complete backward compatibility, provided old receiver key management software can be updated.

Rather than use independent keys in encrypted signals, the keys could relate via a Hash Path like Equation (1). Defense, utility, and other restricted-use stakeholders would retain secure real-time encrypted use by receiving the Hash Path Start as Privileged Users. After a specific key is used, that key is widely broadcast, like any other TESLA scheme. Given the security of the Hash Path as it is released backward, released Hash Points do not compromise security for the Privileged Users. Therefore, this modification would allow civilian users a cryptographically authenticated signal at comparative security to the real-time, confidential service for Privileged Users. For this service to be useful, the seed change frequency must be fast since the seed change frequency determines the time to authentication, like with the signals from the above sections.

We desire to minimize the required civilian side-channel. Equation (1) provides a n-bit security pathway that requires only N bits per distribution. Equation (5), together with Equation (1), generates as many keys as required. As written, Equation (5) provides a secure key-generation function that can accommodate any time t_i and any SVN. Then, Equation (6) can encrypt the original spreading code to generate a new one with the original cipher.

$$k_{i,\text{SVN}} = \text{HMAC}\left(p_i^P, t_i||\text{SVN}\right) \tag{5}$$
$$S_{i,\text{SVN}} = \text{ENCRYPT}\left(k_{i,\text{SVN}}, S_{i,\text{SVN}}\right) \tag{6}$$

The minimal seed can be delivered in multiple ways. An internet-based service would be useful for autonomous driving systems. An SBAS based service would be useful for civilian aircraft, provided close coordination between the GNSS and SBAS provider is achievable. In each case, a receiver would need to store raw radio frequency data from the transmission time until the release of the corresponding Hash Point. Depending on the provider of this side-channel service, the release could occur immediately after sending the Hash Point. This bounds detection time between Hash Point changes in the derived spreading code.

We previously mentioned that the Privileged Users would retain secure, real-time access. However, if those stakeholders discontinued their current key distribution practice and adopted the aforementioned side-channel access, the encrypted signal loses its vulnerability to key leakage. For instance, the United States military could retain its current practice of key distribution for its M-code Service and allow its deprecated P(Y) service to follow the procedure here. The P(Y) signal could serve to secure the M-code against key leakage. In either case, without knowledge of its current key distribution, using this procedure, the required key distribution is only 128-bits per change to cover all satellites.

This signal has many flexibility traits relevant to different stakeholders as discussed in Section III.1.

a). Comparison to Chimera

Comparing to Chimera, the schemes of this work essentially discard the PRF-watermark insertions in favor of an entire spreading code that is a PRF. Since Chimera is an augmentation of existing signal designs, it would be best to compare Chimera to the P(Y)-Galileo-E6-B/C augmentation of Section III.3.

For the scheme in Section III.3 to work with the satellites in orbit at the time of this writing, several modifications must come together in ways that may not be feasible. First, the satellites must be able to rekey themselves quickly. Specifically, the rekey rate determines the time-to-authentication, so ideally, the rekey cadence would be on the order of seconds. Second, the receivers currently in operation must modify their rekeying procedure to accommodate the rapid rekeying. Since Chimera aims to augment the signal in future satellites and signals, requiring changes to the spreading code design, it would not be fair to compare Chimera to Section III.3 applied to existing satellites. Rather, we should compare Chimera to a modified P(Y)-Galileo-E6-B/C of a future system.

Chimera provides backward compatibility with existing receivers since receivers can ignore the existence of watermarks. The Chimera signal provides the ability to listen to a signal choose whether to authenticate it. For the scheme of Section III.3, observing the signal already ensures authenticity; there is no way to observe and choose not to authenticate it under its security. Since non-Privileged Users cannot observe the Section-III.3 scheme currently, providing it as a publicly authenticated signal would not remove from the current public signals from use. With Chimera, the watermark signal degradation will decrease receiver signal power unless future satellites have a larger power.

For the scheme of Section III.3, the signal itself remains unchanged. There is no signal degradation; only the rekeying procedure is changed. The entire spreading code of Section III.3 is a PRF; whereas, for Chimera, a PRF is inserted into a known spreading code. This means that the density of PRF-derived information is higher for the signal of Section III.3. This ultimately means that the theoretical minimum time-to-authentication would be lower with the scheme of Section III.3.

The choice to employ watermarked signal authentication or to employ Section III.3 signal that is completely a PRF, is a political one. Would it be better to augment an existing signal with watermarks or modify an existing signal's rekeying procedure? Leaving more detailed analysis for later work, we claim that the latter would be easier. From a technical perspective, modifying the rekeying procedure would not require redesigning the signal and radio receivers. Existing technical know-how in the sectors of Priveldeged Users (e.g., defense, utilities) would immediately be available for the open case, noting that radio memory would need to be augmented to accommodate the raw radio data buffer requirements. Satellites transmitter know-how would also be immediately applicable since only the frequency of rekeying would change. Power requirements would not need to change, and spectrum would not need to be transferred away from Privileged User use. Whereas, all of these conveniences are not the case with Chimera.

IV. MODULAR SIGNALS AND SOLUTION AUTHENTICATION

With the signals of Sections III.1 III.2 and III.3, the pseudoranges fall under the Repeater-limited Authentication Secure. Those Repeater-limited-Authentication-Secure pseudoranges could provide a positioning and timing solution. Then what is the use of other non-secure signals? Before, we discussed how any authenticated ranging must exist under a delayed authentication framework resulting from the need to have a spreading code replica, exposing a fundamental limitation. Authenticated signals can only provide delayed authentication.

Unless the externally secure precise timing can be provided and the Repeater-limited-Authentication-Secure signal's rekeying cadence is fast, non-secure signals still provide receivers the necessary service. Non-secure signals can provide real-time ranging; secure signals can provide delayed ranging. This motivates our suggestion to separate the ranging authentication problem from real-time ranging modularly. Focused resources can separately provide several real-time, insecure-ranging services and a single delayed, secure-ranging service. The single delayed, secure-ranging service can be used to authenticate all other real-time, insecure-ranging services since all pseudoranges must be consistent. A modular methodology avoids augmenting severely bandwidth-limited signals with cryptography that would provide an inflexible, super-delayed authentication. Rather, it suggests authoring a single delayed, form-fitted, and cryptography-first signal to authenticate all other signals.

We discuss two detection paradigms, as conceptually depicted in Figure 9. The first is a delayed-detection paradigm. A receiver reasons in the past about the authentication of all real-time signals. Provided the rekeying cadence is short, upon observing a mismatch of real-time and delayed pseudoranges, a receiver should be notified reasonably quickly not to trust the real-time signals. The second is an IMU-based real-time detection paradigm. A receiver uses IMU data to propagate

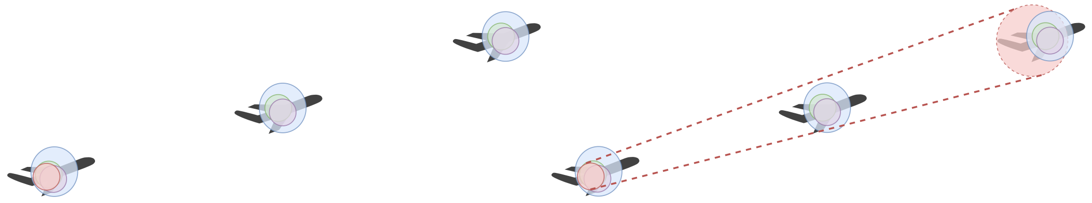

Figure 9: A conceptual diagram of two detection paradigms with modular authentication signals. Left: a receiver compares the real-time signal pseudoranges with the delayed authenticated ones in the past. A receiver will be able to detect false signals after the authentication delay. Right: a receiver uses IMU and other sensors to create a current position estimate compared to real-time pseudoranges. Within both sides of the diagram, the blue estimate represents the position and timing solution ("PNT") from a course signal, and the green and purple estimates represent PNT from public precision signals from different GNSS systems. The red represents the PNT from a delayed authenticated signal (assumed precision-level solution accuracy since smaller ships are harder to estimate for replay). If any of the signals do not match the authenticated one, a receiver detects spoofed signals.

the authenticated solution in the past to the current real-time solutions. The tightest detectable mismatch would occur if the delayed, secure pseudoranges originated from the same satellites as the real-time signals. Assuming a normal distribution on the pseudorange, one could check the normal z-score deviation since the two signals act as one measurement. In the case of a delayed, secure pseudoranges authenticated another constellation's pseudoranges; one can form statistical tests based on the Receiver Autonomous Integrity Monitoring ("RAIM") framework. In the same way, RAIM detects malfunctioning satellites, a RAIM strategy that observed consistent real-time insecure pseudoranges that were inconsistent with consistent delayed, and secure pseudoranges would detect forgery under Repeater-limited-Authentication-Security.

V. CONCLUSION

This work examines what a start-from-scratch, cryptography-first GNSS system would look like. Since we start from scratch, this design is unencumbered by the bandwidth limitations of current systems. We observe how a cryptography-first design provides additional flexibility to the system design requirements, such as how the rekeying cadence determines the time-to-authentication. The use of TESLA adds features relevant to GNSS. We suggest that modifying the rekeying procedure of current signals could provide a publicly authenticated ranging service. Since any publicly authenticated ranging service must provide a delayed authentication, we also suggest that authentication efforts be modularly separated. New GNSS systems should consider separate signals: an optimized real-time insecure signal and an optimized delayed secure signal since the secure signal can authenticate the others.

REFERENCES

[1] I. Fernández-Hernández, V. Rijmen, G. Seco-Granados, J. Simon, I. Rodríguez, and J. D. Calle, "A navigation message authentication proposal for the galileo open service," *NAVIGATION*, vol. 63, no. 1, pp. 85–102, 2016. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/navi.125

[2] "Galileo open service navigation message authentication (osnma) user icd for the test phase," vol. 1, 11 2021.

[3] M. Götzelmann, E. Köller, I. V. Semper, D. Oskam, E. Gkougkas, J. Simon, and A. de Latour, "Galileo open service navigation message authentication: Preparation phase and drivers for future service provision," 9 2021, pp. 385–401.

[4] J. M. Anderson, K. L. Carroll, N. P. DeVilbiss, J. T. Gillis, J. C. Hinks, B. W. O'Hanlon, J. J. Rushanan, L. Scott, and R. A. Yazdi, "Chips-message robust authentication (chimera) for gps civilian signals," 2017, pp. 2388 – 2416.

[5] A. Poltronieri, G. Caparra, and N. Laurenti, "Analysis of the chimera time-binding scheme for authenticating gps l1c," in *2018 9th ESA Workshop on Satellite NavigationTechnologies and European Workshop on GNSS Signals and Signal Processing (NAVITEC)*, 2018, pp. 1–6.

[6] J. Hinks, J. T. Gillis, P. Loveridge, Shawn, G. Myer, J. J. Rushanan, and S. Stoyanov, "Signal and data authentication experiments on nts-3," 9 2021, pp. 3621–3641.

[7] O. Pozzobon, L. Canzian, M. Danieletto, and A. D. Chiara, "Anti-spoofing and open gnss signal authentication with signal authentication sequences," in *2010 5th ESA Workshop on Satellite Navigation Technologies and European Workshop on GNSS Signals and Signal Processing (NAVITEC)*, 2010, pp. 1–6.

[8] M. L. Psiaki, B. W. O'Hanlon, J. A. Bhatti, D. P. Shepard, and T. E. Humphreys, "Gps spoofing detection via dual-receiver

correlation of military signals," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 49, no. 4, pp. 2250–2267, 2013.

[9] L. Scott, "Anti-spoofing & authenticated signal architectures for civil navigation systems," in *Proceedings of the 16th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GPS/GNSS 2003)*, 2003, pp. 1543 – 1552.

[10] J. Anderson, S. Lo, A. Neish, and T. Walter, "On sbas authentication with over-the-air rekeying schemes," *ION GNSS+*, 9 2021.

[11] T. Humphreys, "Limitations of signal-side gnss signal authentication," 09 2014, pp. 1351–1363.

[12] R. Annessi, J. Fabini, F. Iglesias, and T. Zseby, "Encryption is futile: Delay attacks on high-precision clock synchronization," *CoRR*, vol. abs/1811.08569, 2018. [Online]. Available: http://arxiv.org/abs/1811.08569

[13] A. Perrig, D. Song, R. Canetti, J. Tygar, and B. Briscoe, "Timed efficient stream loss-tolerant authentication (tesla): Multicast source authentication transform introduction," *Request For Comments (RFC)*, vol. 4082, 2005.

[14] G. Caparra, S. Sturaro, N. Laurenti, and C. Wullems, "Evaluating the security of one-way key chains in tesla-based gnss navigation message authentication schemes," in *2016 International Conference on Localization and GNSS (ICL-GNSS)*, 2016, pp. 1–6.

ACKNOWLEDGEMENTS

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