Authentication Security of Combinatorial Watermarking for GNSS Signal Authentication

Jason Anderson, Sherman Lo, Todd Walter

Abstract

Watermarking Signal Authentication is a technique where a GNSS provider cryptographically perturbs the spreading code to allow for limited cryptographic authentication of the signal. Several proposals and future studies exist or are underway to augment GNSS signals with this capability. This work reintroduces a generalized combinatorial watermarking function that affords a flexible pathway to cryptographically prove the authentication security of the signal with receiver observables under certain assumptions. The security levels can be on par with standard cryptographic security (e.g., 128-bit security) and require little or no additional use of the navigation data bandwidth. We show how our methods apply to signals of different designs and signal-to-noise ratios. From this work, one can design a Watermarking Signal Authentication scheme and the accompanying receiver to have high confidence in a signal's authenticity.

I. INTRODUCTION

Receivers can use Watermarking Signal Authentication to establish trust in the satellite navigation ranging signals [Scott, 2003]. Several proposals and studies exist to augment GNSS signals with this technique, including for GPS [Anderson et al., 2017, Hinks et al., 2021], and for WAAS [O'Hanlon et al., 2022]. These techniques work together with Navigation Message Authentication to establish trust in transmitted navigation data.

Watermarking Signal Authentication watermarks the spreading code by cryptographically pseudorandomly inverting or replacing sections of the signal spreading code. First, the GNSS provider (hereafter, the "provider") makes a cryptographic commitment based on cryptographic information the provider promises to hold secret for a specific time. Next, the provider perturbs the spreading code based on the commitment and broadcasts the perturbed spreading code within the ranging signal. The perturbation is small enough to avoid materially interfering with users ignoring authentication or attempting to acquire the signal. After receipt of the watermarked signal, the receiver stores raw intermediate frequency radio data (hereafter, "raw radio data") for the length of the applicable watermarked spreading code. After a specific time, the provider reveals the commitment used to generate the watermark. Then the receiver reexamines the raw radio data. If the receiver observes a correlation increase when using the watermarked replica (known only to the provider at broadcast), then the receiver has evidence of the authenticity of the signal. In this work, we address the question of how much increase relates to how much authentication security with mathematical proof arguments under adversarial and noise modeling assumptions.

Suppose the constellation's primary authentication framework is Timed Efficient Stream Loss-Tolerant Authentication ("TESLA"). In that case, the constellation can issue a watermarked signal with little or no additional bandwidth on the navigation data channel [Anderson et al., 2022a]. The Watermarking Signal Authentication must derive from a commitment via a secure, one-way key-derivation function (e.g., HMAC). This work reintroduces a function class to perform the watermarking perturbations on the spreading code [O'Hanlon et al., 2022, Anderson et al., 2022a]. We call this function class Combinatorial Watermarking Functions. Combinatorial Watermarking Functions will use a cryptographic commitment to derive a uniform combination of integers from a set. These integers determine which portions of a spreading code to invert. We are explicit and careful in our construction to ensure the necessary cryptographic one-way properties to prove authentication security.

Combinatorial Watermarking functions have several advantages over the suggestions in the prior art. First, Combinatorial Watermarking Functions are generalizable and flexible, allowing design transferability among navigation signals without materially affecting the security and operation analysis. Second, Combinatorial Watermarking Functions can be constructed to ensure uniform spreading code inversion and vast, exponentially-growing watermark spaces. This results from the judicious use of randomized construction and the exponential growth of the number-of-combinations function. The uniformity aids in simplifying mathematical analysis, and exponential growth eases applying standard authentication security (e.g., 128-bit security). Finally, but certainly not the last, the expected degradation is deterministic, further aiding mathematical analysis when considering the receiver observables.

This work considers the observables available to a receiver to determine whether the watermark is present and the signal is authentic. We present methods to compute a convenient statistic to determine authenticity up to missed detection and false alert probabilities. From our methods, one can design an authentication scheme and a receiver to determine the authenticity of a watermarked signal within the scope of authenticity provided by Watermarking Signal Authentication (e.g., limited by meaconing and other complex attacks).

In Section II, we use cryptographic, combinatoric, and sampling arguments to ensure (1) the selection of a watermark and inverted spreading code is uniform, (2) the selection of a watermark or inverted spreading code admits no efficient algorithms to predict future inverted spreading code selections, and (3) the space of available watermarks is sufficiently vast to deter spoofing. These properties lend to tractable mathematical analysis in later sections. In Section III, we show how the adversarial success probability follows the hypergeometric distribution and discuss how to relate and compute the security level of receiver observables without signal power, noise, and sampling considerations. In Section IV, we propose a filter and prove security bounds on the receiver statistics under standard noise assumptions, allowing one to design a receiver to meet specific security levels. Lastly, in Section V, we show experimental results from real signals to validate our models. Altogether, this work shows how to design a Watermarking Signal Authentication scheme and reciever so that an adversarial must exhaustively search or guess watermarks from a search space too vast for achievable computational resources to spoof the authentication receiver observables.

1. Scheme Scope and Notation

In this work, we will repeatedly use n, r, and s to refer to the navigation scheme parameters. Our methods can easily apply to any navigation signal. Here, we divide the navigation spreading code into n sections. The provider selects r sections uniformly and pseudorandomly for a watermark to be inverted. An adversary may elect to spoof a signal with s chip sections inverted by random guess or exhaustive search among the vast watermark space.

We discuss example, theoretical signals based on the GPS C/A and L1C signal designs. As a suggestion from [O'Hanlon et al., 2022], we use $n = 1023$ and $r = 52$ for both GPS C/A and L1C and $n = 10230$ and $r = 520$ for GPS L1C. Our selection of the inverted chip-section duty cycle comes from [O'Hanlon et al., 2022] to provide a representative comparison to Chimera's selected duty cycle [Technology, 2019]. We do not split the inversion duty cycle into fast and slow watermarks for simplicity, noting that it is possible to either (1) design a fast and slow authentication scheme to use the same watermark, as in [Anderson et al., 2022a], or (2) construct the watermark to retain the mathematical modeling used in future sections. Because we also showed how to use data bandwidth to distribute watermarking commitments efficiently, we have each spreading code sequence have its own watermark. For the C/A signal, each millisecond spreading code gets its own watermark where $r = 52$ of the chips are flipped among the total $n = 1023$. For the L1C signal, the spreading code is divided into $n = 1023$ 10-chip sections, of which $r = 52$ are flipped. The same methods can apply to an L1C signal where $r = 520$ of the L1C chips are flipped among the $n = 10230$ or anything similar. Moreover, one could reserve a number of chips as never flipped (noting the harm to our security arguments) or apply it to other spreading code designs.

Certain attack strategies are outside the security scope of Watermarked Signal Authentication. Using proper TESLA synchronization and an onboard clock, longer replay attacks can be prevented [Anderson et al., 2022b]. Security Code and Estimation Replay attacks are out of scope [Arizabaleta et al., 2019]. Therefore, claims about provable authenticity must be qualified with these considerations.

II. SECURE WATERMARK SELECTION

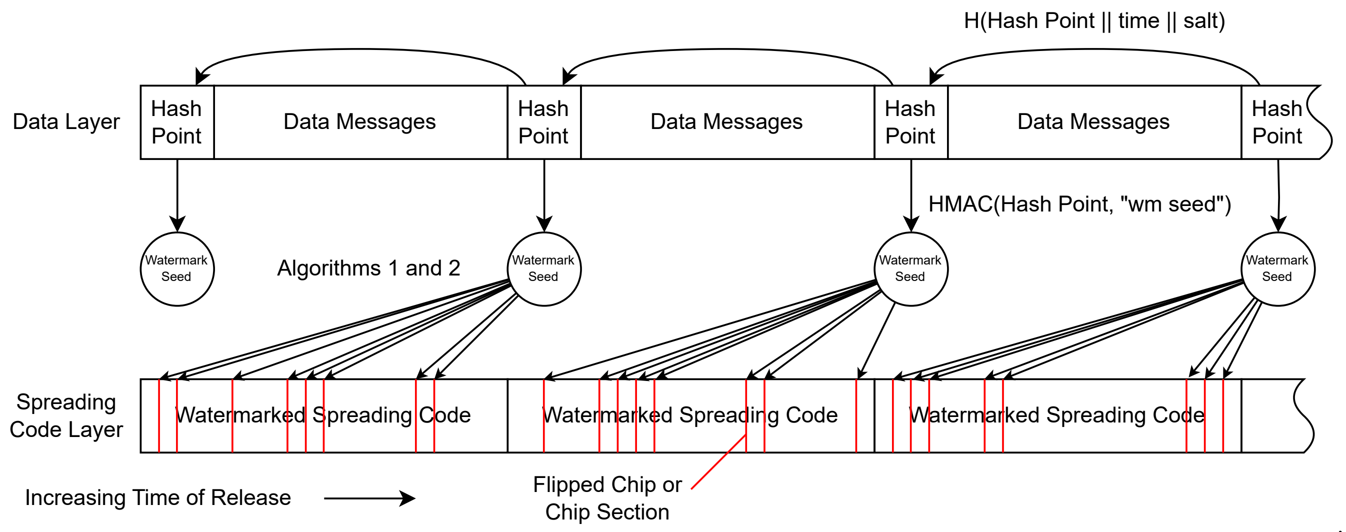
We propose that the provider pseudorandomly select a combination of integers to determine which spreading code sections to invert to create the watermarked spreading code. The selection of inverted spreading code must be uniform to ensure no efficient algorithm can guess a watermark. Section II.1 provides an unbiased combination selection function.

In the context of TESLA, the watermark will derive from the TESLA secret commitments. We must ensure that observation of the watermark lends no efficient algorithm to guess TESLA commitments presumed secret and held by the provider for the delay disclosure time. Otherwise, we break the authentication afforded by TESLA. Moreover, we must ensure that observation of specific chip-section inversions lends to no efficient algorithm to guess other chip-section inversions. We achieve these properties by having the pseudorandom integer function from Section II.1 derive from a cryptographic one-way function. Section II.2 provides a one-way random integer function based on HMAC.

With Sections II.1 and II.2, we construct our secure Combinatorial Watermarking Function. A Combinatorial Watermarking Function uses an input cryptographic commitment to compute a combination of integers that determines the spreading code inversions. A *secure* Combinatorial Watermarking Function (1) yields uniform watermark and chip-section selections, and (2) is cryptographically one-way. Figure 1 depicts an overview of how the properties of a secure Combinatorial Watermarking Function work within the TESLA context to provide authentication.

1. Combination Selection

Suppose we have n chip sections and desire to randomly select r of them to invert to compose a watermark. The provider could draw a random combination in several ways assuming access to a one-way random integer function (Section II.2). A preliminary approach would be to repeatedly draw one-way random integers from 1 to n until r unique elements are drawn. While this



H(Hash Point || time || salt)
Data Layer
Hash Point
Data Messages
HMAC(Hash Point, "wm seed")
Watermark Seed
Algorithms 1 and 2
Spreading Code Layer
Watermarked Spreading Code
Increasing Time of Release
Flipped Chip or Chip Section

Figure 1: A diagram that relates the release time of specific information in the context of TESLA. Each arrow represents a one-way operation, admitting no efficient algorithm to guess the pre-image of each arrow. The top of the diagram is the TESLA chain used for all other authentication, including navigation message authentication. The red lines indicate inverted sections of the spreading code among the total n available. The red lines correspond to pseudorandom integers derived via Algorithms 1 and 2. HP stands for "Hash Point", referring to our terminology in [Anderson et al., 2022a] where the provider uses TESLA secrets for many different authentication aspects (e.g., Navigation Message Authentication).

approach would work, it can be computationally wasteful, especially in the context of a scheme that requires many integers drawn. Floyd provides a better approach with a deterministic number of draws [Bentley, 1987]. Floyd's algorithm provides a way to select r unique elements with exactly r calls to a random integer function. We provide pseudocode with Algorithm 1.

a). Proof of Floyd's Algorithm

For the reader's convenience, herein we provide a proof that Algorithm 1 selects a combination where the probability of the selection of any element is the same, inspired by [Bentley, 1987, Blatter,]. To begin, suppose we have n total elements from which we wish to draw a uniform combination of r elements. Within Algorithm 1, the for-loop iterates j from $n-r+1$ to n (inclusive). Let $P_j(\neg i)$ be the probability that element i is not selected with the j-th for-loop iteration, and let Prob$(\neg i)$ be the probability that element i was never selected after the last iteration of the for-loop. We want to show that Prob$(\neg i)$ is the same for all i. To do this, we must split Prob$(\neg i)$ into two cases that correspond to whether the if conditional within Algorithm 1 could activate: (A) $1 \leq i \leq n-r+1$ and (B) $n-r+1 < i \leq n$.

For Case A, we start with the very first draw, corresponding to when $j = n-r+1$, $P_{n-r+1}(\neg i) = \frac{n-r}{n-r+1}$. Thereafter, $P_j(\neg i) = \frac{j-1}{j}$. Therefore,

$$\begin{aligned}\text{Prob}(\neg i) &= \prod_{j=n-r+1}^{n} P_j(\neg i) \\ &= \frac{n-r}{n-r+1} \cdot \frac{n-r+1}{n-r+2} \cdots \frac{n-1}{n} \\ \text{Prob}(\neg i) &= \frac{n-r}{n}\end{aligned}$$

For Case B, i is not in the range of the random integer drawn until $j = i$. Therefore, $P_j(\neg i) = 1 \quad \forall j < i$. On the $j = i$ draw, following through the for-loop iteration, either the i-th element is selected via the random integer function or the i-th element is selected because the random integer function provided an element previously drawn. Among the i possible elements to draw, $i-(n-r+1)+1$ outcomes lead to drawing i, and $n-r$ outcomes lead to not drawing i. Therefore, $P_i(\neg i) = \frac{n-r}{i}$. Thereafter,

Algorithm 1: A function that selects a pseudorandom combination of integers uniformly from a pseudorandom seed.

```
// n:  The total number of elements that can form any combination.
// r:  The number of elements selected for a particular combination.
// seed:  The pseudorandom seed from which the uniform combination will derive.
// salt:  Salt to deter a pre-computation attack.
function pseudorandom_combination(n, r, seed, salt)
    combination = Set() // Initialize set data structure
    hmac_rand_int_cache // Initialize empty hmac_hmac_int cache.
    for j = n-r+1, ..., n do
        // Derive random integer from 1 to j in a cryptographic one-way manner.
        rand_int, hmac_rand_int_cache = hmac_rand_int(1, j, seed, salt, *hmac_rand_int_cache)
        // Make unbiased selection.
        if rand_int not in combination then
            combination.add(rand_int)
        else
            combination.add(j)
        end
    end
    return combination
```

like with case A, $P_j(\neg i) = \frac{j-1}{j}$. Therefore,

$$\begin{aligned}\text{Prob}(\neg i) &= \prod_{j=n-r+1}^{n} P_j(\neg i)\\ &= \prod_{j=i}^{n} P_j(\neg i)\\ &= \frac{n-r}{i} \cdot \frac{i}{i+1} \frac{i+1}{i+2} \cdots \frac{n-1}{n}\\ \text{Prob}(\neg i) &= \frac{n-r}{n}\end{aligned}$$

Since both Case A and Case B yield the same Prob$(\neg i)$, each element i is selected uniformly among the combinations of r elements.

2. One-way Pseudorandom Integers

Floyd's Algorithm discussed in Section II.1 requires a random integer function. In Algorithm 1, we use an HMAC-based pseudorandom function with Algorithm 2. Other pseudorandom functions could provide the random integers required for Algorithm 1; however, Algorithm 2 meets several desired properties.

One such property is that Algorithm 2 is a one-way pseudorandom function. A one-way function admits no efficient algorithm to determine the function pre-image to a function output. HMAC provides cryptographic security on its one-way property. Therefore, provided the integers derive from HMAC, the integers have the one-way property. Algorithm 2 admits no efficient algorithm to determine the input seed from the output integers.

An adversary can observe the watermark with a radio. Upon observing an inverted chip section in the spreading code, the adversary can deduce the integer derived from our watermarking function. By ensuring that the integer derives from a one-way function, we limit how an adversary can predict future inverted chip sections in the remainder of the watermark and derive or guess the watermarking seed. This also allows the integers in Figure 1 to come from a one-way arrow.

Another property we desire is that the function is efficient with computational resources. One naive way to derive the integers required is to produce an HMAC for each derived integer. HMAC allows the pseudorandomness to be derived in parallel (as opposed to via repeated hash application). Unfortunately, there is no way to sample an arbitrary random integer with a deterministic runtime unless the range of integers is a power of 2. For an arbitrary random integer range, we must sample with rejection to ensure no bias in the pseudorandom integer function output. For instance, the output of HMAC cast to be an integer can be rejected if outside the desired integer range in favor of new HMACs until the output is within the desired range. If the output of HMAC is 256 bits, but the range might be considerably smaller (e.g., 1 to 1023), the function would reject most outputs. To derive an integer i, we should use the smallest number of bits required to delineate the range of positive integers up to the smallest power of 2 larger than i. Therefore, it is desirable to sparingly use the output bits from HMAC so that multiple random integers can be derived from a single call to HMAC.

Lastly, we desire to deter pre-computation attacks against the function. This can be done by introducing salt into the computation of random integers. This salt can be the same salt used for the rest of TESLA, so no additional data bandwidth is required for its distribution.

Algorithm 2 has all the above properties. Because the random bits used to generate the random integers derive from an HMAC derived from the inputs with Lines 3 and 10, we satisfy the one-way property. We cache the output of HMAC to ensure the minimum number of HMAC calls. Line 6 ensure the minimum required number of bits is used for generating each integer. We use a cache containing a counter and start bit to ensure non-periodicity with Lines 3 and 10. Salt is incorporated into the HMAC call to deter an adversary from generating a rainbow table of the seeds to output integers. The recursive structure achieves sample rejection with a concise program.

Algorithm 2: An one-way random integer function based on HMAC.

```
   // a:  The minimum output integer.
   // b:  The maximum output integer.
   // seed:  The input to the HMAC key field while generating random bits.
   // salt:  HMAC message field input to deter pre-computation attack.
   // counter:  HMAC message field input to ensure non-periodicity of HMAC pseudorandom output.
   // start_bit:  The first bit of hmac cache not yet used to generate a random integer.
   // hmac_cache:  A cache of saved random bits from the last call to HMAC.
1  function hmac_rand_int(a, b, seed, salt, counter=0, start_bit=0, hmac_cache="")
       // Generate new pseudorandom bits for the cache.
2      if hmac_cache="" then
3          hmac_cache = HMAC(seed, salt || counter) // Concatenate salt and counter.
4      end
       // the bit length of the output of HMAC (e.g., 256 for HMAC-SHA-256)
5      max_bit = HMAC.length
       // Compute the number of pseudorandom bits to commence random draw
6      bits_needed = FLOOR(LOG2(b-a))+1
       // If not enough unused bits of HMAC cache, then recursively redraw with updated counter
7      if bits_needed > max_bit - start_bit then
8          return hmac_rand_int(a, b, seed, salt, counter+1, 0, "")
9      end
       // Use some random bits and cast as integer
10     random_int = INT(hmac_cache[start_bit:start_bit + bits_needed])
11     if random_int > (b-a) then
           // Reject and redraw random integer if outside the desired range
12         return hmac_rand_int(a, b, seed, salt, counter, start_bit+bits_needed, hmac_cache)
13     else
           // Return unbiased one-way random integer and cache required to generate additional
              random integers at next hmac_rand_int call.
14         return a + random_int, (counter, start_bit+bits_needed, hmac_cache)
15     end
```

III. WATERMARK SECURITY

The requirements of Section II.1 and II.2 and the construction of Algorithms 1 and 2 ensure a *secure* Combinatorial Watermarking Function. Because a secure Combinatorial Watermarking Function admits no efficient algorithm to predict a watermark, we can model adversarial attacks on the watermark as uniform guesses, simplifying mathematical analysis. Moreover, the combinatorial nature of the watermark allows easy exponential expansion of the watermark space enabling standard cryptographic security.

To determine the authenticity of a pseudorange measurement, the receiver records the signal raw radio data and notes the watermarked signal's correlation with the original spreading code. Then, after revealing a commitment according to the TESLA schedule, the receiver recomputes the correlation with the watermarked spreading replica. An adversary may randomly guess a watermark to spoof a signal, which would reflect in the relative correlation measurement. In Section III.1, we show that the distribution of an adversary selecting correct chip-section inversions is the hypergeometric distribution. In Section III.2, we relate the hypergeometric distribution to the receiver correlation observation without power, noise, and sampling considerations.

1. Hypergeometric Distribution of Guessing Chip Sections

To begin, we note that the adversary's ability to guess a watermark is limited to an exhaustive search or randomly guessing from previous sections. Suppose we have a watermark with n chip sections, of which the provider will flip r. A chip section is a section of chips inverted together. For instance, a provider could take a 1023-chip sequence and flip 52 of them to make a watermark. Or, a provider could take a 10230-chip sequence and flip 52 ten-chip sections to make a watermark. The adversary may take the original spreading code replica, infuse its own guessed watermark, and broadcast an unauthentic watermarked signal. The adversary may elect to invert as many chip sections as they want, increasing the complexity of our game; let the number of chip sections inverted by an adversary in an attack be s. There are $\binom{n}{r}$ watermarks for the provider to choose and $\binom{n}{s}$ watermarks for the adversary to choose, and the product is the total number of scenarios.

Now suppose that the adversary guesses h chip sections correctly. Among the r authentic chip-section selections, there are $\binom{r}{h}$ ways for the adversary to guess h correctly and then $\binom{n-r}{s-h}$ ways for the adversary to guess the remaining incorrectly in the same attempt. Therefore we arrive at Equation (1) for the distribution of correct adversarial chip-section guesses, the hypergeometric distribution.

$$\begin{aligned}\text{Prob}(h \text{ correct given } s \text{ adversary strategy}) &= \frac{\binom{n}{r}\binom{r}{h}\binom{n-r}{s-h}}{\binom{n}{r}\binom{n}{s}} \\ \mathcal{H}(h \mid n, r, s) &= \frac{\binom{r}{h}\binom{n-r}{s-h}}{\binom{n}{s}} \end{aligned} \tag{1}$$

2. Authenticity with Basic Correlation Measurements

A receiver will use a correlator to determine authenticity. In this section, we start relating the hypergeometric distribution to the output of a correlator. As a starting point, this section derives probability bounds on the security of the output of a correlator *without* considering (1) signal power, (2) noise, (3) the receiver sampling rate, and (4) the carrier wave. We revisit these effects in Section IV. In this section, we assume single-chip chip sections, one sample per chip, and that the signal is just the spreading code. This allows our derivations to be simpler for the moment so that this section can focus on combinatoric arguments.

To deduce a pseudorange measurement, a receiver correlates a replica of the spreading code with its raw radio data with a Matched Filter [Haykin, 1988]. Let $S_j \in \{-1, 1\}^n$ be the receiver-measured signal with no noise, where n is the number of j-indexed samples over an interval matching the n chips for a single watermark. For instance, a theoretical GPS C/A watermarked signal could flip $r = 52$ chips among the $n = 1023$ so that the loss of replica correlation is about 10%. S can either be authentic or spoofed, for which we denote S^{auth} and S^{spoof}, respectively. Let $R_j, R^w_j \in \{-1, 1\}^n$ be the replica and watermarked replica, respectively, with the same n and j-index convention. Let the Match function be defined as the following, noting that the Match function is equivalent to a convolution via a filter whose kernel is the reversed replica denoted R_-. We utilize the *valid* convolution without padding within a Match. The valid convolution of a signal and the reversed replica of the same sample length is a scalar: the unnormalized correlation.

$$\text{Match}(R, S) = \sum_{j=1}^{n} R_j \cdot S_j = R_{-j} * S_j = R_- * S$$

The receiver will use two replicas (R and R^w), and the signal may be authentic or spoofed; therefore, there are four convolutions useful to this discussion. When S is authentic, $S^{\text{auth}} = R^w$ because only the provider knew R^w at the time of broadcast. Each term of the convolution of Equation (2) aligns, yielding the total number of elements. For the terms of the convolution of

Equation (3), r of them disalign. Since those r sum terms change from 1 to -1 for each misalignment, the difference is twice r.

$$R_{-}^{w} * S^{\text{auth}} = R_{-}^{w} * R^{w} = n \tag{2}$$

$$R_{-} * S^{\text{auth}} = R_{-} * R^{w} = n - 2r \tag{3}$$

In the spoofing case, the adversary may elect to flip s chips, so the convolution with the original replica is always n less twice s, as in Equation (4). For the convolution with R^w, in the case where the adversary guesses every chip incorrectly, the difference results from the r watermarked chips *and* each of the s incorrect chips. This means, in the worst case, the convolution is $n-2r-2s$. For each chip guessed correctly, the misalignment from one of the r provider chips and one of the s adversary chips simultaneously aligns, meaning the convolution increases by 4. Therefore, if the adversary guesses h chips correctly among s, the convolution becomes Equation (5). h is the number of chips that the provider and adversary select together, which follows the hypergeometric distribution from Section III.1 because chip selection is uniform from Sections II.1 and II.2.

$$R_{-} * S^{\text{spoof}} = n - 2s \tag{4}$$

$$R_{-}^{w} * S^{\text{spoof}} = n - 2r - 2s + 4h \tag{5}$$

For our simplified case, we suggest subtracted correlation statistic X of Equation (6), the normalized Match using R^w less the normalized Match using R, to determine the authenticity of S.

$$X = \frac{1}{n}\left(\text{Match}(R^{w}, S) - \text{Match}(R, S)\right) \tag{6}$$

$$= \frac{1}{n}\left(R_{-}^{w} * S - R_{-} * S\right)$$

$$X = \frac{1}{n}(R_{-}^{w} - R_{-}) * S \tag{7}$$

Subsituting Equations (4) and (5) into Equation (6), we derive how our statistic X relates to the hypergemetric distribution $\mathcal{H}$ via f of Equation (8).

$$x = f(h \mid n, r, s) = \frac{1}{n}(4h - 2r) \tag{8}$$

$$X^{\text{spoof}} \sim f(\mathcal{H}) \mid n, r, s$$

Suppose we desire to enforce an l-bit security on the watermark authenticity discrimination statistic X. This means that provided the cryptographic properties on the functions used in the construction of the watermark, our decision based on X admits no efficient adversarial algorithm advantage beyond exhaustive search over at least a 2^l-sized search space. Therefore, we desire to bound the probability of guessing a specific watermark to spoof a statistic on X to be at most 2^{-l}.

To show how to apply a security level on X, we introduce a security parameter b_h within the support of $\mathcal{H}$, and its associated $b_x = f(b_h \mid n, r)$ within the support of X to enforce l-bit security. According to the TESLA schedule, the receiver will make an authentication determination based on a Match increase from using R^w over R after a watermark seed is released. Therefore, we desire l-bit security on an adversary guessing a watermarked spreading code that will sufficiently spoof a statistic $X^{\text{spoof}} \geq b_x$ for every attack strategy s. We must consider all s since an adversary can elect to flip as many chip sections as they wish.

$$\text{Prob}\left(X \geq b_x | n, r, s\right) < 2^{-l} \quad \forall s$$

$$\text{Prob}\left(f(\mathcal{H}) \geq b_x | n, r, s\right) < 2^{-l} \quad \forall s$$

$$\text{Prob}(\mathcal{H}(h \mid n, r, s) \geq b_h) < 2^{-l} \quad \forall s$$

$$\sum_{h=b_h}^{s} \mathcal{H}(h \mid n, r, s) < 2^{-l} \quad \forall s \tag{9}$$

The upper tail of the hypergeometric distribution tail of the form of Equation (9) allows us to compute the probability of an adversary sufficiently guessing a watermark to spoof a statistic based on the subtracted normalized Match X. The problem for

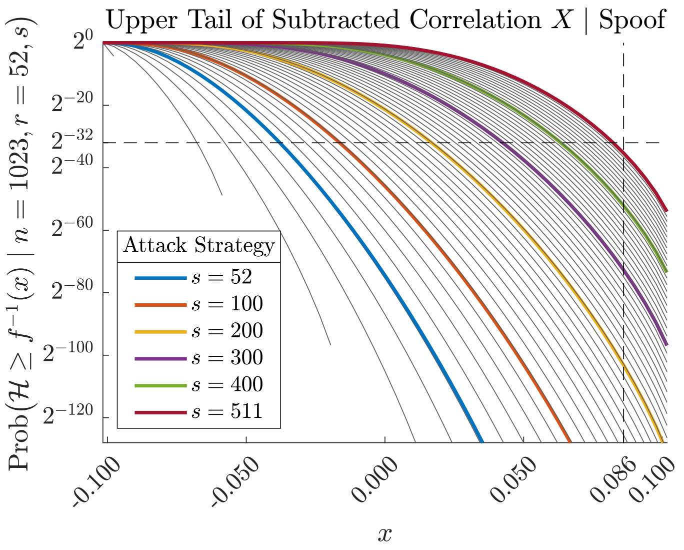


Figure 2: A plot of the upper-tail probability of the distribution of X under spoofing conditions of a single watermark with the system parameters of our example GPS C/A signal of Section III.2 a). The adversary guesses random watermarks. Each line corresponds to a specific attack strategy s, the number of chips the adversary elects to invert. We plot only one-tenth of all s in gray for a clean figure. Some are colored to show trends. The plot also marks the appropriate bound for a 32-bit security level.

designing a watermarking signal authentication is now the selection of signal parameters n, r, and b to provide a desired security level l.

The Hypergeometric cumulative probability density function does not have a convenient form, frustrating further simplification of Equation (9). However, it is possible to continue the analysis of a watermarking design computationally. We provide an example of a theoretical GPS C/A signal in Section III.2 a).

a). Example with GPS C/A

In this section, we provide an example analysis of a theoretical GPS C/A signal. We note that we provide an additional example for GPS L1C similar to what is proposed by Chimera [Technology, 2019] that considered additional effects in Section V. Let us consider the GPS C/A with 1023 chips per millisecond. To achieve about 10% degradation of the signal, we flip 52 of the chips. Therefore, in the nomenclature herein, $n = 1023$ and $r = 52$. Because each flip inverts the autocorrelation power, the expected correlation of this watermark signal will be $\frac{1023-2\cdot 52}{1023} = 0.90$ with the exclusionary noise and power assumptions from Section III.2. The total number of watermarks is very large: $\binom{1023}{52} > 2^{292}$, and the selection of a combinatorial watermark admits no efficient algorithm beyond exhaustive search. Now we proceed to compute the necessary security bound to enforce security at the desired levels with Figure 2.

Figure 2 shows the upper-tail probability an adversary could spoof the subtracted Match statistic X before the distribution of the delayed TESLA seed via Equation (9). The x-axis denotes subtracted normalized Match X. The y-axis denotes the upper-tail probability Prob $\left(\mathcal{H} \geq f^{-1}(x) \mid n = 1023, r = 52, s\right)$ which is the sum from Equation (9) for a representative selection of s. Figure 2 shows one-tenth of the possible s from 0 to 511, with some colored to show trends. When an adversary elects to flip half of the chips, the signal degrades to a pure pseudorandom sequence, and the hypergeometric distribution centers at $X = 0$, providing a worst-case. We note that we expect that the receiver would lose track of the signal long before s approached 511; however, the situation forms our worst-cast suitable for conservative analysis. If the adversary elects to flip all of the chips, then the signal might appear to have an inverted data stream or a phase change, forming a mirrored situation that is corrected in the tracking loop. One could compute probability bounds after first limiting to a more reasonable s (i.e., before a receiver would lose tracking), but interestingly, we find that we can bound 32-bit security even in the $s = 511$ case. Moreover, this is 32-bit security over a 1ms watermark, deferring analysis on longer integration times to later sections.

Many GNSS systems will incorporate navigation message authentication with shorter HMACS [Anderson et al., 2021]. The 32-bit security is satisfactory due to the short time the HMACs must secure the message between broadcast and the delayed disclosure of the TESLA secret. If these systems already find 32-bit security acceptable for navigation message authentication, then they might also find 32-bit security acceptable for the watermarks. From Figure 2, this would mean $X \geq 0.086$ would meet the acceptable security level for *every* attack strategy s and assert authenticity to 32-bit security on X. However, depending on assumptions of the signal processing capabilities, it may be possible to have less stringent bounds on X to assert authenticity.

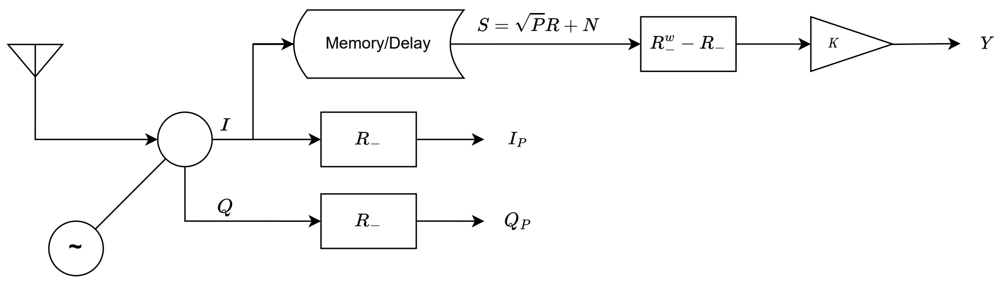


Figure 3: A diagram of our proposed modified Match Filter that produces statistic Y for the purpose of determining the authenticity of a cryptographically watermarked signal. R_- and R^w_- are the reversed signal replicas (including carrier, sampling effects, and the length of the coherent integration), not-watermarked and watermarked, respectively. $R_- - R^w_-$ forms the filter kernel. K is a gain selected later for the mathematical conciseness of the derived probability distributions. While a real receiver may compute the product sum, we are explicit with our use of the valid convolution to aid with propagating noise. We include the Y filter's relation to a typical GNSS tracking loop producing prompt in-phase and quadrature correlation values and the required memory buffer.

Finally, Figure 2 shows that the best chance an adversary has to spoof X is to submit a random sequence to the receiver.

IV. RECEIVER OBSERVABLE SECURITY

With Section III.2, our derivations are without the signal power, noise, and sampling considerations to allow the reader to understand the combinatorics derivation separately from effects induced by actual receivers. By utilizing the properties of the Match Filter and convolution, we now include signal power, noise, and sampling considerations.

Our received signal, S_j, is now the signal immediately preceding the spreading-code correlators that form the tracking loop (after other filtering and amplifications before correlation). S_j is composed of the replica broadcast R_j (including the carrier and spreading code) with power P and noise N_j, as denoted in Equation (10). In Section V, we provide a way to determine $\sqrt{P}$ and σ^2. We model the noise as the typical additive white Gaussian noise, specifying the noise power immediately preceding correlation as $\mathbb{E}[N^2] = \sigma^2$, which is also assumed constant over an authenticity determination. In the case of C/A and L1C, σ^2 only includes in-phase channel noise.

$$S_j = \sqrt{P}R_j + N_j \tag{10}$$

The receiver feeds S_j into a set of correlators to form a tracking loop. We assume the receiver has already acquired the signal, and our security statistics will come from a functioning tracking loop. To incorporate the cryptographic security above, we invoke the properties of the Matched Filter. In our derivation, we utilize the *valid* convolution with no padding. The valid convolution of a signal and the reversed replica of the same sample length results in a scalar. The tracking loop already maximizes this scalar.

A naive approach might be to use the prompt correlator value from the tracking loop and another prompt correlator that executes after receipt of the watermark seed. However, we suggest a different approach where the receiver will have a separate filter for authentication. The separate filter will better account for noise considerations and efficient memory use, which is discussed in Section IV.2. We suggest that the receiver constructs a filter whose kernel is $R^w_- - R_-$. We remind that the Match Filter convolves the reversed replica to maximize the peak indicating correlation. Each R now contains the carrier wave (whether in-phase or quadrature) and spreading code information with the receiver sampling rate and with a coherent integration time over the single spreading code (or more than one in later sections). We note that we discuss how to aggregate over multiple watermarks in Section IV.1. By specifying the filter this way, we are agnostic to the filter sampling rate and whether the samples evenly divide the chip sections. We include a gain K for mathematical conciseness later, arriving at statistic Y in Figure 3.

We feed the signal S through our modified Match Filter to arrive at Equation (11). Since our filter is linear-time-invariant, we can consider each term separately.

$$\begin{aligned} Y &= K \cdot (R^w_- - R_-) * S \\ &= K \cdot (R^w_- - R_-) * (\sqrt{P}R + N) \\ Y &= K\sqrt{P} \cdot (R^w_- - R_-) * R + K \cdot (R^w_- - R_-) * N \end{aligned} \tag{11}$$

First, we derive a good value for K. Section III.2 derives the case without power, noise, and sampling considerations to show that $\frac{1}{n}(R^w_- - R_-) * S^{\text{spoof}} \sim f(\mathcal{H}) \mid n, r, s$. We suggest that the K of Equation (12) be used to map the case *with* power, noise, and sampling considerations to the case *without* to aid mathematical conciseness when deriving safety bounds. This will ensure that the spoofing hypothesis distribution is a simple sum of (1) the hypergeometric distribution $\mathcal{H}$ and (2) the normal noise distribution.

$$K = \frac{1}{2}\frac{1}{\sqrt{P}}\frac{1}{||R||_1} \tag{12}$$

Equation (12) enforces that the output of Y is normalized to the fraction of spreading code inverted.

Like with all of our hypotheses, K is a function of P. We provide suggestions on how to regress P and σ^2 in Section V. A different value of K can be used with the appropriate adjustments to the derivations below; however, our choice removes many constants from our derivations, allows for easier comparison among different scheme designs, and allows us to nakedly cite the derivations from Section III.2 for mathematical conciseness.

Second, we propagate the noise in Equation (11). We compute the propagated noise variance via the Fourier Transform $\mathcal{F}$, noting that the expectation of N is 0 and that the variance is invariant under $\mathcal{F}$ from Parsaval's Theorem. We derive Equation (13) as the variance of the zero-mean filtered noise.

$$\begin{aligned}
\mathbb{E}\left[\left|\left|K \cdot (R^w_- - R_-) * N\right|\right|^2\right] &= K^2 \cdot \mathbb{E}\left[\left|\left|(R^w_- - R_-) * N\right|\right|^2\right] \\
&= K^2 \cdot \mathbb{E}\left[\left|\left|\mathcal{F}\left\{(R^w_- - R_-) * N\right\}\right|\right|^2\right] \\
&= K^2 \cdot \mathbb{E}\left[\left|\left|\mathcal{F}\left\{R^w_- - R_-\right\} \cdot \mathcal{F}\left\{N\right\}\right|\right|^2\right] \\
&= K^2 \cdot \mathbb{E}\left[\left|\left|\mathcal{F}\left\{R^w_- - R_-\right\}\right|\right|^2 \cdot ||\mathcal{F}\left\{N\right\}||^2\right] \\
&= K^2 \cdot \mathbb{E}\left[\left|\left|R^w_- - R_-\right|\right|^2 \cdot ||N||^2\right] \\
&= K^2 \cdot \left|\left|R^w_- - R_-\right|\right|^2 \cdot \mathbb{E}\left[||N||^2\right] \\
\mathbb{E}\left[\left|\left|K \cdot (R^w_- - R_-) * N\right|\right|^2\right] &= K^2 \cdot \left|\left|R^w_- - R_-\right|\right|^2 \cdot \sigma^2
\end{aligned} \tag{13}$$

Last, we consider two hypotheses: (1) the signal is authentic, and (2) the signal is spoofed. In the authentic case, $S = \sqrt{P}R^w + N$, even though the receiver does not know R^w until later. This translates to $R = R^w$ in Equation (11). We arrive at our authentic hypothesis model Equation (14): a simple normal distribution.

$$\begin{aligned}
Y \mid \text{authentic} &\sim K\sqrt{P} \cdot (R^w_- - R_-) * R^w + \mathcal{N}\left(0, K^2\left|\left|R^w_- - R_-\right|\right|^2 \cdot \sigma^2\right) \\
&\sim \mathcal{N}\left(K\sqrt{P} \cdot (R^w_- - R_-) * R^w, K^2\left|\left|R^w_- - R_-\right|\right|^2 \cdot \sigma^2\right) \\
Y \mid \text{authentic} &\sim \mathcal{N}\left(\frac{1}{2||R||_1} \cdot (R^w_- - R_-) * R^w, \frac{1}{4||R||_1^2}\left|\left|R^w_- - R_-\right|\right|^2 \cdot \frac{\sigma^2}{P}\right)
\end{aligned} \tag{14}$$

In the spoofing case, $S = \sqrt{P}R^{\text{spoof}} + N$. In Section III.2, we showed how $\frac{1}{n}(R^w_- - R_-) * S^{\text{spoof}} \sim f(\mathcal{H}) \mid n, r, s$ under the assumption of one sample per single-chip chip-section, no noise and unitary power. Now we recall our choice of K from Equation (12) and derive the transformation g to make Equation (16): a sum of the hypergeometric distribution and a simple normal distribution. The transformation g relies on the uniformity of chip-section inversions from our combinatorial watermark.

$$y = g(h \mid n, r, s) = \frac{1}{n}(2h - r) \tag{15}$$

$$\begin{aligned}
Y \mid \text{spoofed} &\sim \frac{1}{2}\frac{1}{||R||_1} \cdot (R^w_- - R_-) * R^{\text{spoof}} + \mathcal{N}\left(0, \frac{1}{4}\frac{1}{||R||_1^2}\left|\left|R^w_- - R_-\right|\right|^2 \cdot \frac{\sigma^2}{P}\right) \\
Y \mid \text{spoofed} &\sim g(\mathcal{H}) \mid n, r, s + \mathcal{N}\left(0, \frac{1}{4}\frac{1}{||R||_1^2}\left|\left|R^w_- - R_-\right|\right|^2 \cdot \frac{\sigma^2}{P}\right)
\end{aligned} \tag{16}$$

While Equations (14) and (16) account for any signal power, noise, and sampling rate, to make this more derivation concrete, suppose a receiver has a fixed sampling rate F for the signal S over a coherent integration time T that also evenly divides the number of chip sections n of which r is inverted. This means the following.

$$
\begin{aligned}
||R||_1 &= F \cdot T \\
(R^w_- - R_-) * R^w &= 2\frac{r}{n}FT \\
||R^w_- - R_-||_1^2 &= 4\frac{r}{n}FT \\
Y \mid \text{authentic} &\sim \mathcal{N}\left(\frac{r}{n}, \frac{r}{n}\frac{\sigma^2}{P}\frac{1}{FT}\right) \qquad (17) \\
Y \mid \text{spoof} &\sim g(\mathcal{H}) \mid n, r, s + \mathcal{N}\left(0, \frac{r}{n}\frac{\sigma^2}{P}\frac{1}{FT}\right) \qquad (18)
\end{aligned}
$$

From Equations (17) and (18), we observe the trade off that a receiver must make between authentication time, authentication uncertainty, and choice of hardware. Given our models, we need now hope to find a decision boundary b_y with an acceptable probability of false alarms and missed detection, Equations (19) and (20) respectively.

$$
\begin{aligned}
\text{Prob}(Y < b_y \mid \text{authentic}) &< \text{Allowable False Alarm Rate} \qquad (19) \\
\text{Prob}(Y \geq b_y \mid \text{spoofed}) &< 2^{-l} \qquad (20)
\end{aligned}
$$

Equation (19) lends to a trivial false alert probability bound via the Gaussian cumulative probability density function. Equation (20) is more complicated, and we discuss it in Section IV.1.

1. Missed Detection and False Alert Probability Bounds

Now that we have identified the distributions of the two relevant hypotheses, we now discuss how to compute missed detection and false alert probabilities given a decision boundary on Y. We have a simple normal distribution for the authentic hypothesis, lending to trivially using the normal cumulative probability density function to compute the false alert probability. We suggest using convolution to compute the probability density function of Y for the spoofed hypothesis.

Starting with Equation(18), we note the structure is that of a simple sum. The probability density function of a sum of two probability distributions is the convolution of the two probability distribution functions, provided their supports are the same, as in Equation (21).

$$
\mathcal{Y} \mid \text{spoof} = \text{Prob}\left(Y = y \mid \text{spoof}\right) = g(\mathcal{H}(y) \mid n, r, s) * \mathcal{N}\left(y \middle| 0, \frac{r}{n}\frac{\sigma^2}{P}\frac{1}{FT}\right) \qquad (21)
$$

We can extend this when observing multiple watermarks. Now suppose Y now includes H individual watermarks. For instance, an L1C receiver with a spreading code length of 10 ms might perform 100 coherent integrations over one second with $H = 100$ watermarks. To account for this, we first note in our formulation K compensates by a factor of $\frac{1}{H}$ so that Y^{auth} is still centered at $\frac{r}{n}$. Individual Y are independent because individual watermarks are cryptographically independent, and the noise is assumed independent. The distribution of $\mathcal{Y}$ with H watermarks is just $\mathcal{Y}$ convolved with itself H times. For abuse of notation, we note H-repeated convolution with $(\cdot)^{*H}$.

$$
\begin{aligned}
\mathcal{Y} \mid \text{spoof}, H &= \frac{1}{H}\mathcal{Y}^{*H} \\
&= \frac{1}{H}\left(g(\mathcal{H} \mid n, r, s) * \mathcal{N}\left(0, \frac{r}{n}\frac{\sigma^2}{P}\frac{1}{FT}\right)\right)^{*H} \\
&= \frac{1}{H}g(\mathcal{H} \mid n, r, s)^{*H} * \mathcal{N}\left(0, \frac{r}{n}\frac{\sigma^2}{P}\frac{1}{FT}\right)^{*H} \\
\mathcal{Y} \mid \text{spoof}, H &= \frac{1}{H}g\left(\mathcal{H}(n, r, s)^{*H}\right) * \mathcal{N}\left(0, \frac{r}{n}\frac{\sigma^2}{P}\frac{1}{FTH}\right) \qquad (22)
\end{aligned}
$$

In our derivation of Equation (22), we exploit the properties of convolution and deliberately separate $\mathcal{H}$ and $\mathcal{N}$ to save with computation. The repeated convolution of $\mathcal{N}^{*H}$ can be done analytically as the sum of identically distributed normal random

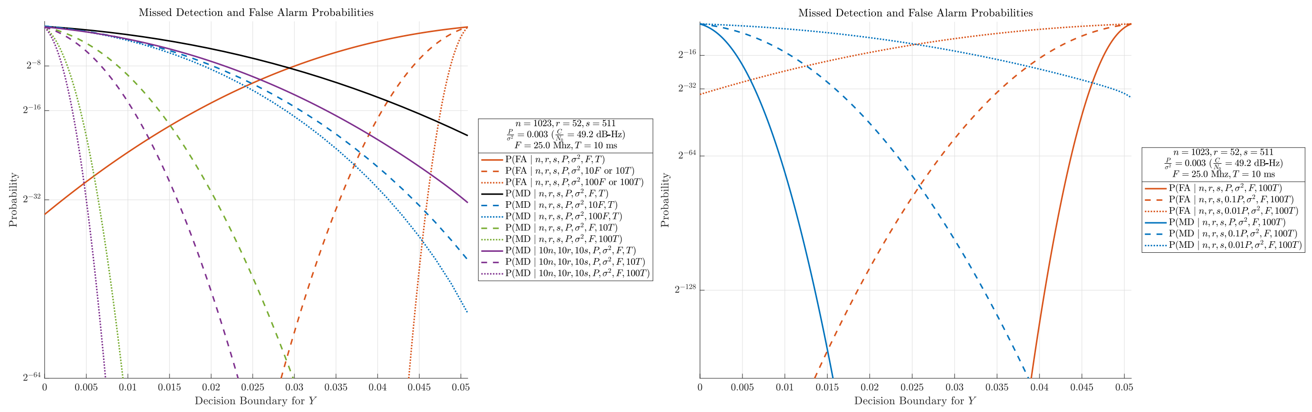


Figure 4: For a given decision boundary on Y, the false alert ("FA") and missed detection ("MD") probabilities of various scheme designs and situations. The parameters are based on a theoretical L1C watermarked signal, and the signal power is based on observations of L1C signals from Section V. For the (n, r) L1C scheme, the 10230-chip sequence is divided into 1023 10-chip sections, and individual chip sections are inverted. For the $(10n, 10r)$ L1C scheme, 520 of the 10230 chips are inverted in a watermark. For situations where the integration time is longer than a single spreading code (e.g., $10T$, $100T$), the curve is computed via repeated convolution as described in Section IV.1. The line types (i.e., whether solid, dashed, or dotted) indicate which of the MD and FA probabilities correspond. For instance, each dashed MD line should be used with the red dashed FA line.

variables. The repeated convolution of $\mathcal{H}^{*H}$ can be done with the limited 0 to r domain first and then up-sampled to the desired granularity for the final convolution with $\mathcal{N}^{*H}$. The repeated convolution can also be done with Exponentiation by Squaring and via the Fast Fourier Transform. While the computation of $\mathcal{Y} \mid$ spoof, H can be done once offline, as H becomes larger to accommodate non-ideal receivers, ignoring these computational considerations and techniques becomes costly. By implementing these techniques, we could compute $Y \mid$ spoof, $H = 10000$ in less than 20 seconds with laptop hardware.

In Figure 4, we use Equation (22) to compute upper tails as a function of decision boundary selection in Y for the spoof hypothesis. The upper tails are the probabilities of missed detection. We use Equation (17) to compute lower tails for the probability of false alarms as a function of decision boundary selection in Y. We consider an L1C signal where the 10230 chips are divided into 1023 10-chip sections, of which 52 are inverted per watermark (i.e., the n, r case) and the case where 520 of the 10230 chips are inverted (i.e., the $10n, 10r$ case). Our default signal characteristics comes from the equipment and experimental results from Section V. This includes the $T = 10$ ms coherent integration time per watermark from the L1C spreading code, F = 25 MHz from the equipment, and the predicted pre-correlation signal-to-noise-ratio from Section V.

From Figure 4 (left), we observe the following trends. With the blue lines, increasing F decreases the probability of missed detection. With the green lines, increasing the integration time to include multiple watermarks dramatically decreases the probability of missed detection. This results from the independence among individual watermarks. With the purple lines, increasing the scheme parameters decreases the probability of missed detection. Among those three trends, it appears that integrating over multiple watermarks is the most effective way to decrease missed detection and false alert probabilities. From Figure 4 (right), we observe that we can use longer integration on the order of seconds to compensate for weaker signals.

2. Receiver Memory

Our suggestion to use a separate filter from the track loop allows for a more flexible receiver design and more efficient memory use. From Figure 4, we expect that lower-cost receivers may require longer integration times. However, the required memory can be reduced if the TESLA commitments are released more frequently. For instance, suppose the TESLA commitments were released every six seconds, but the receiver's sampling rate, operating conditions, and protection levels required observing Ys over a few minutes. After the distribution of the TESLA commitment after six seconds, the memory required for the raw radio data for those six seconds can be dumped in favor of just the smaller number of integrated samples of Y where $R^w - R \neq 0$. Therefore, the memory required (e.g., a ring buffer) for a receiver corresponds to $F \cdot T_{\text{TESLA}}$, where T_{TESLA} is the maximum delay before a watermark's secret commitment is released according to the TESLA schedule. And, higher protection levels are available without substantially increasing receiver memory in low-cost receivers.

V. EXPERIMENTAL VALIDATION

While rudimentary simulations simulating signal watermarks confirm our method's ability to predict the distributions of Y, we found an opportunity to demonstrate efficacy with actual data by modifying an existing software design radio [cus, 2023]. We

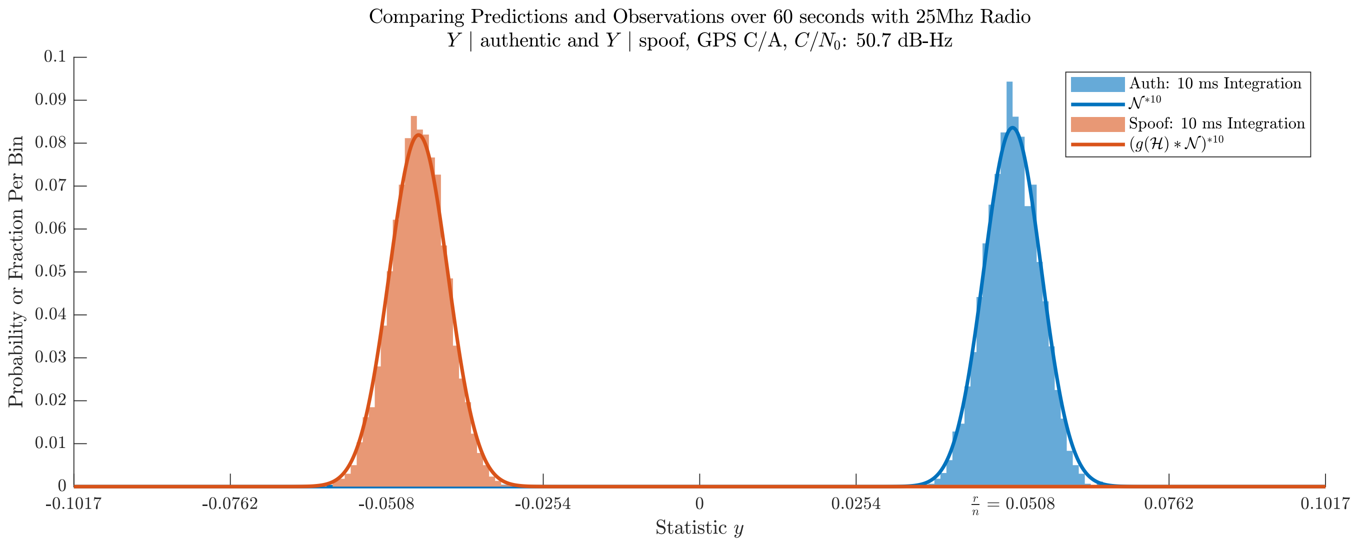


Figure 5: A comparison of our predicted distributions with post-process GPS C/A observations made with a 25 Mhz software-defined radio (based on a mirrored problem described in Section V). Over a 60-second time period, we measure 6000 of the proposed statistic Y, each computed over a 10 ms integration within a converged tracking loop. Each millisecond contains its own mirrored watermark, so each Y is related to 10 independent watermarks, and we computed $Y \mid$ spoof via repeated convolution. In our scenario, the adversary has elected to randomly guess $s = 52$ chips to flip per ms against $r = 52$ chips flipped by the provider.

collected data with a Universal Software Radio Peripheral providing raw 16-bit I and Q samples at 25 Mhz. Figure 5 and 6 validates our methods on GPS C/A data and L1C data, respectively.

Since no watermarked signals were broadcast at the time of writing, we tested our methods on a mirrored situation. In a real Watermarking Signal Authentication scheme, the receiver correlator tracks a watermarked signal using the original spreading code. Instead, to simulate the authentic situation, we had the receiver correlate with a watermarked replica to track the real original signal. The net result is the negation of statistic Y. To simulate the spoofed situation, we had the Y filter use two watermarked replicas. The first was one random watermarking function application of the original spreading code. The second was an additional random watermarking function application of the first watermarked replica. Provided both watermarks were the same number of flips (i.e., $r = s$), we have the inverted Y for the spoofing situation. In our results with both scenarios, we compensate for the mirrored situation by negating K within the receiver.

Our methods in Section IV.1 presume knowledge of P and σ^2, the pre-correlation signal power and noise, respectively. Receivers will not be sensitive enough to make an authentication determination based on a 1 ms or a 10 ms integration, meaning that the decision should incorporate longer integration. This naturally leads to a simple way to regress an accurate P from those individual integrations. We suggest dividing a mean prompt correlation divided by the expected gain from the correlator, as in Equation (23). In Equation (23), we use a watermarked prompt correlation, and we divide by $R_{-} * R^w$ because the signal would be watermarked in a real scheme.

$$\widehat{\sqrt{P}} = \frac{\text{mean}(I_p)}{R_{-} * R^w} = \frac{\text{mean}(I_p)}{(1 - 2\frac{r}{n})FT} \tag{23}$$

We found regressing σ^2 trickier but ultimately backtracked the receiver-estimated $\frac{C}{N_0}$ through the correlator using derivations from [Falletti et al., 2011] with Equation (24).

$$\hat{\sigma}^2 = \frac{1}{2}\frac{N_0}{C} B_{\text{eq}} P ||R||_1 = \frac{1}{2}\frac{N_0}{C} FP; \tag{24}$$

In Equation(24), we take the $\frac{C}{N_0}$ back through the correlation by dividing by $||R||_1$, correct for the equivalent bandwidth B_{eq}, and multiply by $\frac{1}{2}$ since our pre-correlation S has been stripped of the quadrature components, leaving a noise-to-signal ratio that is multiplied by our regressed P. Alternative ways could be more effective, including those that directly regress or bootstrap the ratio. We defer to future work to ensure that the regression is resistant (or provably immune) to manipulation by an adversary. However, we suspect that adversarial attacks will always increase σ^2, which will be appropriately accounted for in the receiver's confidence in authenticity.

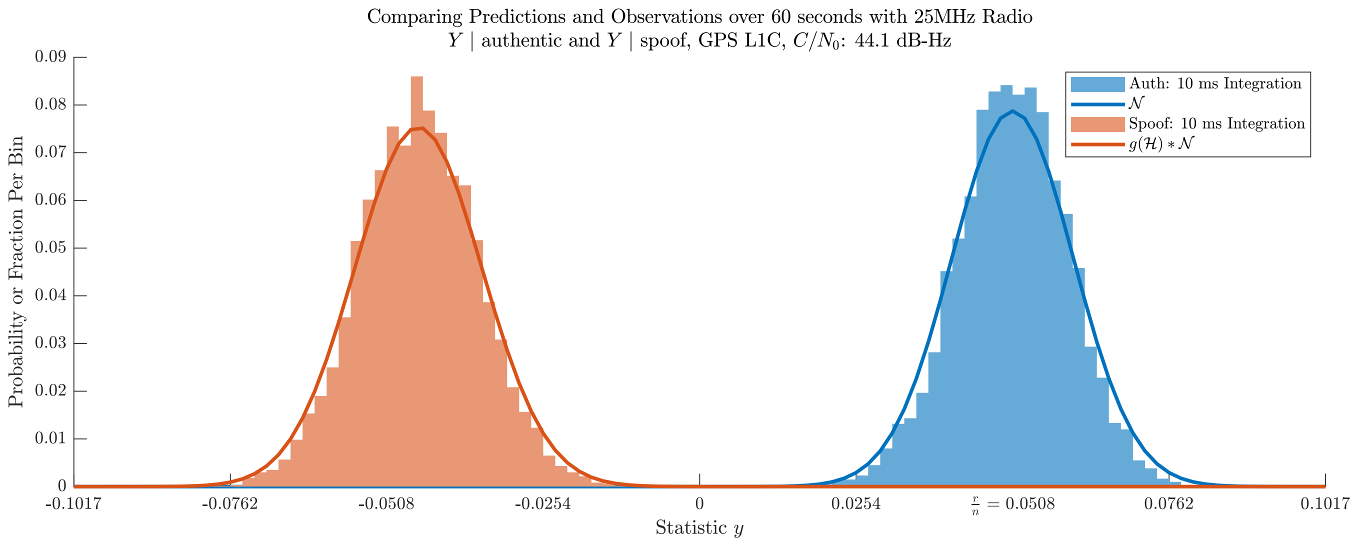


Figure 6: A comparison of our predicted distributions with post-process GPS L1C observations made with a 25 Mhz software-defined radio (based on a mirrored problem described in Section V). Over a 60-second time period, we measure 6000 of the proposed statistic Y, each computed over a 10 ms integration within a converged tracking loop. Each 10 ms contains its own mirrored watermark where sections of 10 chips each were flipped together. In our scenario, the adversary has elected to randomly guess $s = 52$ chip sections to flip per ms against $r = 52$ chip sections flipped by the provider. Since there are 52 chip sections of 10 chips flipped, a total of 520 chips are inverted per watermark.

In Figure 5, the receiver replica is watermarked by inverting 52 chips from 1023 chips per millisecond. Each one-millisecond spreading code received its own watermark. The statistic Y was computed on ten consecutive watermarks and individual coherent integrations in the converged tracking loop. We chose these parameters to match with L1C and Figure 6 and demonstrate the efficacy of using convolution to predict the distribution. In Figure 6, the watermark is generated by dividing the L1C spreading code into 1023 10-chip sections. Among the 1023 10-chip sections, 52 of the 10-chip sections were flipped every ten milliseconds.

In Figure 5 and 6, we validate our methods with real data. We observe that the spoof distribution closely resembles a normal distribution, an effect from both the concentration of the hypergeometric distribution and the Central Limit Theorem. While one could approximate both hypotheses with normal distributions in post-processing, our methods predict the center and spread of the spoofed condition distribution. Given an estimation of the signal $\frac{C}{N_0}$, one can predict the distribution of $Y \mid$ spoof in the worst-case adversarial attack (i.e., $s = \frac{n}{2}$) to compute security bounds on Y for real-time processing (e.g., Figure 4). Furthermore, we validate our method's ability to predict radio requirements.

VI. CONCLUSIONS

In this work, we reintroduce Combinatorial Watermarking Functions. Our methods present several advantages over the state-of-the-art. These include a flexible and transferable security scheme that presents a mathematical pathway to prove security bounds. We show that, given model assumptions on receiver noise, the receiver response to these watermarks can be mathematically modeled, and we can derive security bounds on the level of typical security levels. Through our derivation, we find the tools required to design a scheme and corresponding radio capable of determining the authenticity of a signal. We validate our mathematical models with radio data collected with a software-defined radio.

REFERENCES

[cus, 2023] (2023). Cu-sdr-collection. `https://github.com/gnsscusdr/CU-SDR-Collection`.

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

[Anderson et al., 2022a] Anderson, J., Lo, S., and Walter, T. (2022a). Efficient and secure use of cryptography for watermarked signal authentication. In *Proceedings of the 2022 International Technical Meeting of The Institute of Navigation*, pages 68–82.

[Anderson et al., 2022b] Anderson, J., Lo, S., and Walter, T. (2022b). Time synchronization for tesla-based gnss-enabled systems. In *Proceedings of the 35th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2022)*, pages 3408–3417.

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

[Arizabaleta et al., 2019] Arizabaleta, M., Gkougkas, E., and Pany, T. (2019). A feasibility study and risk assessment of security code estimation and replay (scer) attacks. In *Proceedings of the 32nd International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2019)*, pages 1039–1050.

[Bentley, 1987] Bentley, J. L. (1987). A sample of brilliance. *Commun. ACM*, 30(9):754–757.

[Blatter,] Blatter, C. What's the proof of correctness for robert floyd's algorithm for selecting a single, random combination of values? Mathematics Stack Exchange.

[Falletti et al., 2011] Falletti, E., Pini, M., and Presti, L. L. (2011). Low complexity carrier-to-noise ratio estimators for gnss digital receivers. *IEEE Transactions on Aerospace and Electronic Systems*, 47(1):420–437.

[Haykin, 1988] Haykin, S. (1988). *Digital Communications*. John Wiley & Sons, Inc., New York.

[Hinks et al., 2021] Hinks, J., Gillis, J. T., Loveridge, P., Miller, S., Myer, G., Rushanan, J. J., and Stoyanov, S. (2021). Signal and data authentication experiments on nts-3. pages 3621–3641.

[O'Hanlon et al., 2022] O'Hanlon, B., Rushanan, J. J., Hegarty, C., Anderson, J., Walter, T., and Lo, S. (2022). Sbas signal authentication. In *Proceedings of the 35th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2022)*, pages 3369–3377.

[Scott, 2003] Scott, L. (2003). 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)*, pages 1543 – 1552.

[Technology, 2019] Technology, A. F. R. L. S. V. D. A. G. (2019). Is-agt-100: Chips message robust authentication (chimera) enhancement for the l1c signal: Space segment/user segment interface.