Time Synchronization for TESLA-based GNSS-enabled Systems

Jason Anderson, Sherman Lo, Todd Walter *Stanford University*

BIOGRAPHY

Jason Anderson is a PhD candidate at the GPS Laboratory at Stanford University.

Sherman Lo is a senior research engineer at the GPS laboratory at Stanford University.

Todd Walter is a Professor of Research and director of the GPS laboratory at Stanford University.

ABSTRACT

Timed Efficient Stream Loss-tolerant Authentication ("TESLA") is well-posed to serve as the primary cryptographic authentication algorithm for GNSS. TESLA provides excellent features, such as loss-tolerance and bandwidth efficiency, while also explicitly incorporating the need for time synchronization. This work discusses how to certify and re-synchronize the onboard clock to loosely time-synchronization requirements in the context of GNSS. Our certification and re-synchronization procedure is secure against a man-in-the-middle delaying adversary. By secure, we prevent forgeries from being accepted, but denial of service is still possible. Moreover, we discuss security considerations of the onboard clock regarding the disclosure of the onboard clock time.

I. INTRODUCTION

In a security context, Global Navigation Satellite System ("GNSS") signals include two layered components: (1) the spreading code and (2) navigation data. The spreading code layer serves as the ranging signal, allowing users to deduce their range to a satellite. The navigation data layer serves to provide users the satellite positions and other constellation data. From the ranges and satellite positions, users deduce their position and time. This work pertains to receiver time synchronization with the cryptographic methods establishing trust in the received GNSS signals between a GNSS provider and a receiver.

To protect the spreading code with authentication cryptography, two methods exist in the literature: (1) watermarking the spreading code with cryptographic pseudorandom punctures (e.g., Anderson et al. (2017, 2022b)) and (2) encrypting the signal and employing a delay-release schedule of the encryption keys (Anderson et al., 2022a). To protect the navigation data with authentication cryptography, one method is already in practice (Fernández-Hernández et al., 2016; Gal, 2021; Götzelmann et al., 2021). Challenges hinder the quick augmentation of GNSS with authentication cryptography, such as bandwidth limitation and maintaining backward compatibility. While the bandwidth limitation of today's signals inspires the current adoption of Timed-Efficient Stream Loss-tolerant Authentication ("TESLA") (Perrig et al., 2005), the nature of GNSS signals requires it (Anderson et al., 2022a).

TESLA boils down to the following scheme. The provider commits transmitted information by transmitting a keyed-cryptographic hash. The receiver notes the receipt time of that commitment. Provider releases the key used to make that commitment. The receiver authenticates the transmitted information provided (1) the delay-distributed key hashes down to a hash signed by an asymmetric method such as Elliptic Curve Digital Signature Algorithm ("ECDSA"), (2) the delay-distributed key and transmitted data generate the bit commitment, and (3) the commitment was received before the key was released publicly according to the schedule agreed to by provider and receiver and guaranteed by the loosely time-synchronized assumption. Underpinning TESLA is the assumption that provider and receiver are loosely time-synchronized.

TESLA presumes that the provider and receiver are loosely time-synchronized, posing a catch-22 since the GNSS signal provides the time. The loose-time synchronization requirement specifies a bound on how much a receiver clock *lags* the provider clock. Whereas previous work on this problem discussed methods and requirements assuming limited clock synchronization (Fernandez-Hernandez et al., 2020), this work discusses how one can certify a receiver as loosely time-synchronized and how to synchronize via a non-GNSS external connection. The description of TESLA provides a simple protocol to bootstrap loose-time synchronization (Perrig et al., 2005). This work applies the protocol to the GNSS context and discusses how to implement synchronization to prevent attacks.

1. Notation and Preliminary Assumptions

In this work, we will frequently describe the time of an event in two different clock frames: the provider clock and the receiver clock. Let a particular synchronization event be indexed by l, which will include several sub-events indexed by i. An event i during synchronization l, occurs at t_i^l in the provider clock frame and τ_i^l in the receiver clock frame, via the measurement Equation (1) where θ^l is the receiver clock offset during synchronization l.

$$\tau_i^l = t_i^l + \theta^l \tag{1}$$

For simplicity of notation, since this work examines a single synchronization event, we drop l in future equations.

We make several assumptions required for the bounds derived later to work. First, the internal clock oscillator of a receiver has not been tampered with by an adversary and produces an output that is monotonically increasing. Second, the clock offset during a synchronization event is constant, which is reasonable provided the synchronization event is short. Third, when a receiver connects to a server for the purpose of re synchronization, the server is not engaged in an attack against the receiver. This means that while an adversary might intercept and delay signals between receiver and provider, the provider (or server itself) is faithful (discussed further in Section I.4). Lastly, we assume that the provider clock does not drift. When GNSS is the provider, the clock will not drift in any meaningful way because GNSS is synchronized directly to atomic clocks.

2. TESLA Time Synchronization Bootstrap Protocol

For TESLA to be secure, the receiver's clock may not lag the provider's clock by more than a specified bound, which we will call Θ. Therefore, the receiver clock offset must satisfy Equation (2).

$$\theta > -\Theta \tag{2}$$

This Θ must be at least the length of time between a message's bit-commitment and the release of the key from which it was derived. A good Θ will conservatively also include the expected receiver clock drift during a transmission session and any other non-ideal clock considerations of the system. If the receiver's clock lags the provider's greater than Θ, the receiver will accept messages signed with previously distributed keys, which could be forged messages.

To bootstrap a safe Θ bound, the provider and receiver first must establish an authenticated message protocol (e.g., ECDSA). Then they performed the protocol of Algorithm 1 from (Perrig et al., 2005) and diagrammed by Figure 1, or another equivalent. The content of each message of this protocol is authenticated via cryptography (e.g., with ECDSA).

Algorithm 1 TESLA Time Synchronization Bootstrap Protocol. Diagrammed with Figure 1.

```
1: Receiver computes a nonce $k$.
2: Receiver sends a secure message $m_1 = (k, \tau_1, s_1^{\text{receiver}})$ where $\tau_1$ is the time receiver recorded at the moment of sending $m_1$
   and $s_1^{\text{receiver}}$ is an authentication signature on $(k, \tau_1)$.
3: Provider records $t_2$, the time of receipt of the message $m_1$.
4: Provider sends a secure message $m_2 = (k, \tau_1, t_2, s_2^{\text{receiver}})$ back to receiver, where $s_2^{\text{receiver}}$ is an authentication signature on
   $(k, \tau_1, t_2)$.
5: $\Theta = t_2 - \tau_1 + D$, where $D$ is an estimated bound of the clock drift during the TESLA session. $\Theta$ will be how long a
   provider delays sending a TESLA key for a message bit-commitment.
6: Provider must release a TESLA key at least $\Theta$ after the last instance it was used to commit a message.
```

To show the security of Algorithm 1, we first make the reasonable assumption of tamper-proof clock hardware so that the clock measurements themselves are not manipulated. The content of m_1 and m_2 is protected with cryptographic authentication security; however, if an adversary were to intercept m_1, they could maliciously increase the t_2 measurement. Concretely, if an adversary in the middle intercepts m_1, they could not manipulate the nonce k nor the value of τ_1 transmitted, but they could delay the arrival of m_1 thereby increasing t_2. Equation (3) relates events 1 and 2 in the provider's clock frame, and event 1 proceeds event 2, regardless of which frame.

$$\begin{aligned} t_2 - t_1 &> 0 \quad (3) \\ t_2 - (\tau_1 - \theta) &> 0 \\ \theta &> -(t_2 - \tau_1) \quad (4) \end{aligned}$$

After substituting the measurement function, we arrive at Equation (4), which places a lower bound on the actual receiver clock offset.

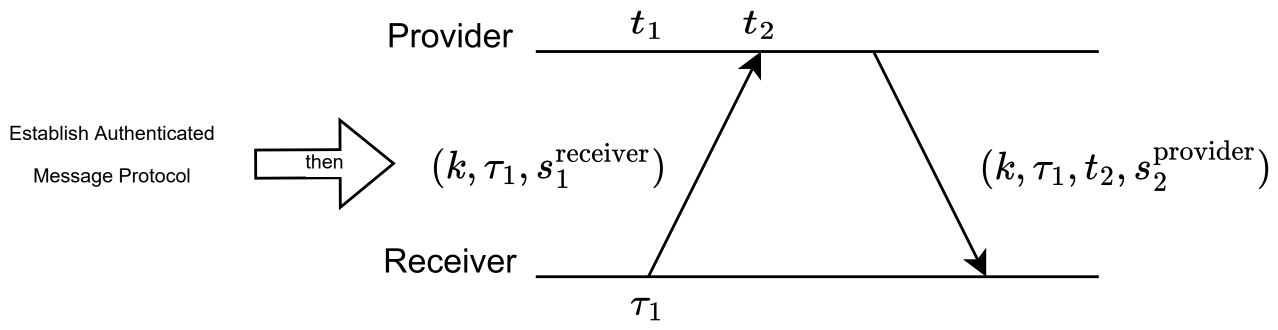
Provider
t_1
t_2
Establish Authenticated Message Protocol
then
$(k, \tau_1, s_1^{\text{receiver}})$
$(k, \tau_1, t_2, s_2^{\text{provider}})$
Receiver
τ_1

Figure 1: A conceptual diagram of Algorithm 1. The diagram depicts increasing time from left to right in the provider and receiver clock frames, and the messages shared between them. The protocol presumes provider and receiver have already established a cryptographic authentication instance to protect the *content* of the messages transmitted. The protocol is secure against adversary-induced delays in message transmission but vulnerable to denial of service.

Since we began with Equation (3) to derive Equation (4), we certify that, at the time provider and receiver underwent Algorithm 1, the receiver clock lag to the provider clock was no more than the bound in Equation (4). If an adversary delays m_1, the computed Θ increases with the adversary's delay, meaning a provider would increase the length of the delay to release keys. If an adversary delays the message transmission too long, it can only deny receiver service since the provider will further delay the release of keys for authentication. The D in Algorithm 1 accounts for estimated drift between applications of the protocol by simply making the bound more conservative.

Nowhere in this protocol is the receiver's clock estimate modified. In the unfortunate case where a receiver's clock is substantially delayed, the computed Θ will be large, delaying authentication substantially to ensure authentication security.

Algorithm 1 could be augmented with concepts from Precision Time Protocol ("PTP") (Shankarkumar et al., 2017). For instance, the receiver and provider could be augmented with specialized hardware and resend a better τ_1 and t_2, respectively, after their first transmission. This would serve to shrink Θ slightly, making the time-to-authentication smaller. However, as long as the stakeholders find a Θ reasonable, PTP is not necessary.

Algorithm 1 allows an individual provider and receiver to set Θ. However, with GNSS, this is not possible because GNSS serves everyone broadcast-only. This work includes discussing the correct application of this bootstrapping algorithm to GNSS.

3. Network Time Protocol and Network Time Security

Network Time Protocol ("NTP") from Martin et al. (2010) provides a simple synchronization protocol between two connected clocks. Network Time Security ("NTS") from Franke et al. (2020) is NTP augmented with cryptography authentication security so that the *content* of the messages is protected. These methods purport to measure the clock drift θ. We provide NTS with Algorithm 2 and Figure 2. Previous work has evaluated their application to TESLA GNSS schemes (O'Driscoll et al., 2020).

Algorithm 2 Network Time Security. Diagrammed with Figure 2.

```
1: Receiver computes a nonce $k$.
2: Receiver sends a secure message $m_1 = (k, \tau_1, s_1^{\text{receiver}})$ where $\tau_1$ is the time receiver recorded at the moment of sending $m_1$
   and $s_1^{\text{receiver}}$ is an authentication signature on $(k, \tau_1)$.
3: Provider records $t_2$, the time of receipt of the message $m_1$.
4: Provider sends a secure message $m_2 = (k, \tau_1, t_2, t_3, s_2^{\text{receiver}})$ back to receiver, where $s_2^{\text{receiver}}$ is an authentication signature
   on $(k, \tau_1, t_2, t_3)$.
5: Receiver records $\tau_4$ at the moment of receipt of message $m_2$.
6: The measured clock drift is $\hat{\theta} = \frac{1}{2}(\tau_1 - t_2 - t_3 + \tau_4)$ assuming that the transit time of $m_1$ and $m_2$ are the same.
```

To derive an estimated clock drift θ, NTS assumes that the transit time for m_1 and m_2 are the same, which motivates Equation (5)

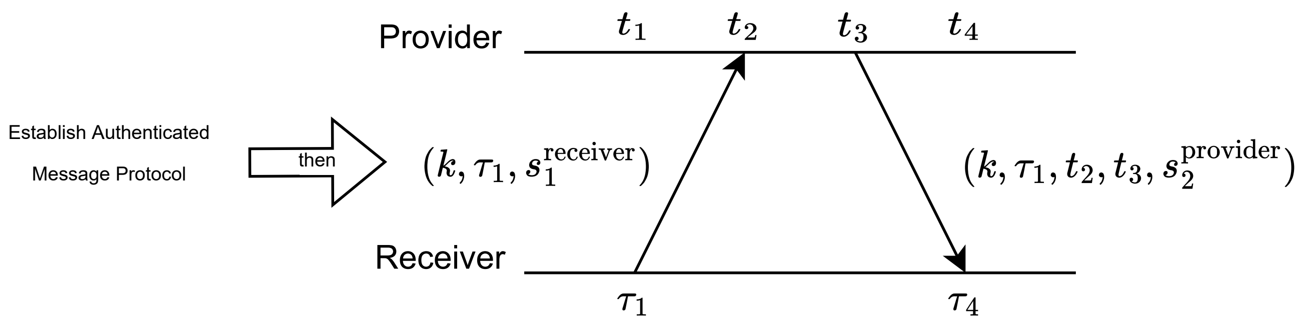
Provider
t_1
t_2
t_3
t_4
Establish Authenticated Message Protocol
then
$(k, \tau_1, s_1^{\text{receiver}})$
$(k, \tau_1, t_2, t_3, s_2^{\text{provider}})$
Receiver
τ_1
τ_4

Figure 2: A conceptual diagram of Algorithm 2. The diagram depicts increasing time from left to right in the provider and receiver clock frames, and the messages shared between them. The protocol presumes provider and receiver have already established a cryptographic authentication instance to protect the *content* of the messages transmitted. The protocol is susceptible to adversary-induced delays in message transmission.

in the provider clock frame. Substituting the measurement equation into Equation (5) yields Equation (6).

$$t_2 - t_1 = t_4 - t_3 \tag{5}$$
$$t_2 - (\tau_1 - \hat{\theta}) = (\tau_4 - \hat{\theta}) - t_3$$
$$\hat{\theta} = \frac{1}{2}(\tau_1 - t_2 - t_3 + \tau_4) \tag{6}$$

Once again, an adversary can delay the messages of NTS without affecting the content, meaning that an adversary can manipulate t_2 and τ_4 to *arbitrarily* modify the deduced $\hat{\theta}$. Given the threat model against which we wish to protect (see Section I.4), we cannot use the $\hat{\theta}$ for synchronization without additional certification.

Using the same argument from Section I.2, we can derive a maximum lead on the clock drift. Equation (7) relates events 3 and 4 in the provider's clock frame, and event 3 proceeds event 4, regardless of which frame.

$$t_4 - t_3 > 0 \tag{7}$$
$$(\tau_4 - \theta) - t_3 > 0$$
$$\theta < \tau_4 - t_3 \tag{8}$$

After substituting the measurement function, we arrive at Equation (8), which places a upper bound on the actual receiver clock offset.

Combining Equations (4) and (8), we arrive at final bounds for θ that are not susceptible to man-in-the-middle delays, which we provide again with Equation (9).

$$-(t_2 - \tau_1) < \theta < \tau_4 - t_3 \tag{9}$$

Since we do not get to synchronize continuously, we can re-add an estimated clock drift D between synchronizations to arrive at Equation (10) with conservative bounds that can apply for the duration of a TESLA session.

$$-(t_2 - \tau_1) - D < \theta < \tau_4 - t_3 + D \tag{10}$$

While the transit times for messages m_1 and m_2 can be individually manipulated, the sum T is directly observable. Starting from the sum transit time with Equation (11) in the provider frame yields Equation (12) via the measurement equations.

$$T = t_4 - t_3 + t_2 - t_1 \tag{11}$$
$$= (\tau_4 - \theta) - t_3 + t_2 - (\tau_1 - \theta)$$
$$= \tau_4 - t_3 + t_2 - \tau_1 \tag{12}$$

Observing changes of T can either mean there is traffic hindering synchronization communication or there is a man-in-the-middle delay in the transmission in either m_1 and m_2. However, if one reasonably characterizes the typical traffic competing with the NTS request, one can protect against man-in-the-middle attacks by ignoring NTS results that takes a long time to come back.

4. Threat Model On Loose Time Synchronization

In this work, the concepts apply to vehicles capable of returning to safety upon losing GNSS. For instance, in civilian aviation, we presume a pilot can land the aircraft even if GNSS is denied. For an autonomous vehicle, we presume it can navigate itself to a safe area with other sensors (e.g., cameras or LIDAR) upon loss of GNSS. Since authenticating GNSS signals already poses such a challenge, we focus on ensuring the unforgeability of the signal. Therefore, we do not consider denial of service vulnerabilities as breaking our schemes for this work.

When a receiver clock lags the provider clock by more than Θ, then TESLA breaks, and messages are forgeable. This is because a receiver will accept a message's bit commitment after the TESLA key -from which the commitment was derived- is released. For instance, suppose a GNSS receiver has no onboard clock. An adversary can first jam the true GNSS signals. Then, upon receiving a particular TESLA key, it can forge the navigation data by generating message commitments as if they had occurred before the release of the key. The spoofed GNSS signal will have a receiver deduce a time that lags in real-time. Because an adversary-generated delayed or repeated signal would induce a GNSS receiver's timing estimate to lag real-time, we establish that, at a minimum, a receiver now must have its own onboard clock.

Using an onboard clock will observe if the deduced GNSS timing estimate instantaneously jumps or drifts quickly, mitigating the case of the previous paragraph. For instance, for the scenario of the last paragraph, an onboard clock can detect large, instantaneous changes in timing measurements resulting from a spoofed GNSS signal. The receiver cannot distinguish between a faulty onboard clock and a spoofed GNSS time; however, if a spoof with a large time jump occurs, it will be caught. However, an adversary could engage in a repeater attack while slowly spoofing a delay at a rate below what is detectable by the onboard clock (e.g., below the known clock drift rate) over a large length of time. In fact, a despicable adversary could plant a repeater device near the unsuspecting GNSS receiver that slowly induces a lag in the receiver timing estimate. Once the induced lag breaks TESLA security, the adversary can forge messages. We seek to protect against this despicable adversary in this work. Because of this delayed attack on the GNSS broadcast-only service, we cannot establish trust in the GNSS timing estimate with GNSS.

To establish trust in the GNSS timing estimate, we employ an isolated onboard clock and certain algorithms that require a non-GNSS connection to ensure that the clock meets TESLA loose-time synchronization requirement. We discuss the case where this non-GNSS connection is available all the time or infrequently (e.g., only during a maintenance session). In our threat model, these non-GNSS connections are also susceptible to man-in-the-middle delay attacks. Any signal, whether GNSS or otherwise, can be delayed in transit by an adversary.

II. ONBOARD CLOCK USE

There is more than one way a receiver can enforce that a bit commitment was received before its key distribution. In this work, we suggest a receiver requires that operations related to TESLA use its onboard clock estimate. While the GNSS timing estimate will provide an accurate bias on the onboard clock, that bias cannot be trusted for operations related to TESLA. The information provided by the accurate but untrusted clock bias should not be needed for the TESLA application. For example, for the SBAS proposal (Anderson et al., 2021), all timing operations pertinent to TESLA are rounded down to the nearest integer, meaning the GNSS timing corrections should not be germane over shorter periods of time (e.g., between maintenance sessions). The scheme will refuse to authenticate if the onboard clock deviates from the measured GNSS time by more than one second.

The onboard clock deviations against GNSS time can be used to indicate when a clock is either faulty or the signal is delayed. The onboard clock should monitor GNSS time for instantaneous jumps and throw alarms when that occurs. This will mitigate some of the attacks of Section I.4 but not ones whose delay rate is slow. A clock will not be able to detect those attacks. Therefore, the onboard clock should have its drift characterization well understood. From this characterization, we can apply Fernandez-Hernandez et al. (2020) to determine the maximum time the onboard clock estimate can be trusted before requiring non-GNSS re-synchronization procedure in Section IV. That maximum time must be enforced because comparing the onboard clock to GNSS cannot detect a slowly-delaying signal.

A vehicle should never report its onboard clock time except when that report leads to a check against the TESLA requirement. An adversary may not need to spoof the GNSS time of a specific vehicle; rather, given the stochastic nature of clock drift, it could wait and find a vehicle with a lagging onboard clock. In other words, to achieve its nefarious goals, an adversary needs any vehicle, not a specific one. Therefore, a vehicle should hold its onboard clock estimate in confidence and only disclose pursuant to checking its clock lag. To protect against the case where a server leaks the contents of encrypted communication to adversaries, future use of NTS pings do not include the τ_1 in the transmitted message.

III. GNSS TESLA SECURITY CONDITION

For GNSS, the signal is broadcast-only, meaning that Θ is immutably set for all receivers. In recent SBAS proposals, this time is 6 seconds (Neish, 2020; Anderson et al., 2021); in Galileo's OSNMA and GPS's Chimera, this time a few seconds to a few minutes, depending on the operating mode (Anderson et al., 2017; Fernández-Hernández et al., 2016; Gal, 2021; Götzelmann et al., 2021). To ensure the security of the GNSS TESLA-based schemes, a receiver must certify its onboard clock satisfies Equation (2) ignoring incoming current GNSS measurements.

At the most basic level, either the onboard clock lags the provider's clock by more than the system's Θ or not. Algorithm 3 provides a simple algorithm to determine the aforementioned condition.

Algorithm 3 A server-provided secure test to ensure a receiver satisfies the loose-time synchronization requirement for a TESLA-based GNSS Authentication.

```
1:  Receiver establishes a secure *encrypted and authenticated* connection for all of the following messages.
2:  Receiver computes a *time-independent* nonce $k$.
3:  Receiver sends a secure message $m_1 = (k, )$ and records $\tau_1$ at the moment of sending $m_1$. $\tau_1$ is held in confidence from
    the server. $m_1$ is not sent at a fixed frequency (e.g., at the top of a minute) to minimize leakage of $\tau_1$.
4:  Provider records $t_2$, the time of receipt of the message $m_1$.
5:  Provider sends a secure message $m_2 = (k, t_2, s_2^{\text{receiver}})$ back to receiver, where $s_2^{\text{receiver}}$ is an authentication signature on
    $(k, t_2)$
6:  Let $D$ be a bound on the estimated clock drift between now and the next application of this Algorithm 3.
7:  **if** $t_2 - \tau_1 + D < \Theta$ **then**
8:      TESLA Secure until next application of this Algorithm 3.
9:  **else**
10:     TESLA not secure. Receiver must refuse to authenticate TESLA messages and throw maintenance requirement alarms.
11: **end if**
```

Algorithm 3 is secure for the following reasons. For this instance, Θ is set by the system, so we derive Equation (13) to serve as a test that determines the outcome of Algorithm (3).

$$\begin{aligned} \theta &> -\Theta \\ -(t_2 - \tau_1) &> -\Theta \\ t_2 - \tau_1 &< \Theta \\ t_2 - \tau_1 + D &< \Theta \end{aligned} \tag{13}$$

Equation (13) is only satisfied when the receives clock's lag is sufficiently small to ensure the security of TESLA.

The content of each of the messages is protected with integrity checks, meaning k and t_2 are each secure. An adversary engaged in a delay attack on Algorithm 3 can manipulate to increase the t_2 measurement; however, since increasing t_2 only makes the test condition of Algorithm 3 more difficult to satisfy, that attack will only serve to deny receiver service. An adversary cannot manipulate a receiver, whose clock does not satisfy Equation (2), into falsely believing its clock does satisfy Equation (2) due to Equations (3) and (4). Therefore, the only feasible attack will only cause a denial of service, ensuring security.

We specify in Algorithm 3 that the messages should be sent in *encrypted and authenticated* deliberately. An adversary may not be concerned with spoofing the time of a particular receiver; rather, they might try to find a receiver whose clock is already close to breaking the loose-time synchronization assumption. By ensuring that the application of this protocol is confidential, an adversary cannot be tempted to exploit receivers with malfunctioning or near-maintenance-required clocks if τ_1 is sent with m_1 or if the nonce were to reveal information about τ_1. The *time-independent* nonce further mitigates leakage of τ_1, which could easily be done by the receiver secretly encrypting its nonce if it is time-dependent.

1. Application to Civil Aviation

For an autonomous vehicle with a non-GNSS connection (e.g., internet), it is trivially easy for anyone to implement a server that provides clients access to Algorithm 3. However, civilian aviation receivers do not have a non-GNSS connection. Therefore, the main challenge will be building the infrastructure and procedures needed to facilitate a connection between aviation receivers and a service that provides access to Algorithm 3. This could be done in a variety of ways, such as (1) allowing receivers to connect to an external network, (2) incorporated this check into receiver periodic and startup maintenance, (3) incorporated into pilot procedures, or (4) implemented with Automatic Dependent Surveillance Broadcast ("ADSB").

Allowing the receiver to connect to an external network serving Algorithm 3 would likely be the most ideal and future-proof solution. The security of a receiver connected to such a server should not be a concern given Equation (3) and the arguments provided in Section III. For the avoidance of doubt, a receiver connected to a server providing access to Algorithm 3 would not modify the internal clock. It simply notifies the user of a clock maintenance requirement, and any spoofing would cause a denial of service. In fact, such a connection (with Algorithm 3) effectively serves as a periodic time-reporting requirement. Imagine a requirement where, in order to fly, an aircraft must report its clock time to an authority that ensures that the aircraft time is not lagging too far behind. In the case where the server acts as an authority, τ_1 should be added back into the *encrypted* m_1. Regardless, receivers and the requirements of the standards are currently built without allowing an external connection, which may continue into the near future.

A compromise to the solution presented in the aforementioned paragraph would be to require the connection only during periodic maintenance or when the receiver is installed in the aircraft. For instance, before installing a receiver into the aircraft, a technician checks that the onboard clock does not lag in real-time via a temporary connection to a server. It would be prudent that receivers contain clocks with sufficiently low drift rates (among the operating latitude and altitude-induced temperature range) to maximize the length of time between checks.

As a further compromise, Algorithm 3 does not necessarily need to be conducted by an automated server. Rather, it could be incorporated into pilot procedures. For instance, if a pilot had visual access to the receiver clock, a pilot could compare with air traffic control as a part of take-off procedures or against their own personal cellular device. There remain human vulnerabilities to this security and assumptions about the security of cellular-provided time; however, this would serve to make an attack on the TESLA security more difficult. Such a procedure would require a D-term large enough to cover the entire flight duration and accuracy expectations with human communication so that it can be performed before take-off (and not on approach). For instance, a pilot and air traffic control can agree that a pilot will call out “mark” at the top of the next minute. For the SBAS proposal of (Anderson et al., 2021), the allowable delay is six seconds. Conservatively setting the tolerance to three seconds, if air traffic control hears “mark” three seconds after its clock moves past the top of that minute, air traffic control should notify the pilot that their onboard clock must be checked.

Lastly, if an aircraft reported its time via ADS-B, it would be immediately apparent to the ground and other aircraft with secure access to time when a particular aircraft’s time is not meeting the security requirement. However, we do not make a claim on the feasibility of updating the ADS-B standard. If an aircraft does broadcast its time, it must be checked; otherwise, an adversary may select a vulnerable aircraft to spoof.

IV. PROCEDURE TO SECURELY SYNCHRONIZE AN ONBOARD CLOCK

This section describes the synchronization procedure of an onboard clock securely against our threat model of Section I.4. This procedure would be performed on an ongoing basis with a receiver with a non-GNSS connection. Otherwise, this procedure would be performed on an infrequent basis, balancing (1) the onboard clock drift characterization and (2) a reasonable maintenance schedule. Naively, one could apply the procedure of Section I.3 to compute θ; however, as discussed, that procedure is vulnerable to man-in-the-middle delays. Therefore, an NTS ping can only provide the bounds on θ derived with Equation (10), with D a bound on the expected onboard clock drift between applications of this procedure.

Lets consider the naive case where an NTS query is trusted. An adversary would want to spoof $\hat{\theta} > \Theta$ with the NTS query (“a Leading Spoof”). The synchronizer would take that spoofed measurement and correct the clock leaving the clock with $\theta < -\Theta$, thereby allowing TESLA to break. Conversely, if an adversary were to spoof $\hat{\theta} < -\Theta$ with an NTS query (“a Lagging Spoof”), the synchronizer would correct the clock to have $\hat{\theta} > \Theta$. Whereas a Leading Spoof leads to breaking TESLA security, a Lagging Spoof causes a denial of service. In the Lagging Spoof case, the receiver will believe TESLA keys to having been released before receipt of a message bit commitment. To achieve a safe synchronization procedure, we modify the NTS procedure to limit how much a clock offset can be modified with Algorithm 4. The limitation makes the TESLA time check step more brittle, but ensures that no forgeries would be accepted even if the synchronization were manipulated by an adversary.

From the bounds of Equation (10), we arrive at Algorithm 4, which ensures the re-synchronized clock satisfies both bounds are met after synchronization.

Let θ^* be the actual correction applied to a clock during synchronization. To arrive at the limiting conditions on θ^*, we start with the Equation (10). We then apply the correction θ^* to arrive at Equation (14), which provides bounds on the clock drift after correction.

$$\begin{gathered} -(t_2 - \tau_1) - D < \theta < \tau_4 - t_3 + D \\ -(t_2 - \tau_1) - D - \theta^* < \theta - \theta^* < \tau_4 - t_3 + D - \theta^* \end{gathered} \tag{14}$$

If the onboard clock bounds after correction violate the TESLA requirements, as in Equation (15), then the NTS query failed to provide a good enough synchronization that is secure against man-in-the-middle attacks.

$$-\Theta < \theta - \theta^* < \Theta \tag{15}$$

Therefore, for the upper bound, we arrive at Equation (16).

$$\begin{aligned} -\Theta &< -(t_2 - \tau_1) - D - \theta^* \\ \theta^* &< -(t_2 - \tau_1) - D + \Theta \end{aligned} \tag{16}$$

For the lower bound, we arrive at Equation (17)

$$\begin{aligned} \tau_4 - t_3 + D - \theta^* &< \Theta \\ \tau_4 - t_3 + D - \Theta &< \theta^* \end{aligned} \tag{17}$$

Typically, there will be a range of θ^* that satisfy both bounds. In that case, picking the middle would be reasonable, which is the naive NTS estimate. In the case where there does not exist a θ^*, we derive the maximum acceptable NTS query return time.

$$\begin{aligned} \tau_4 - t_3 + D - \Theta &< -(t_2 - \tau_1) - D + \Theta \\ \tau_4 - t_3 + 2D &< -(t_2 - \tau_1) + 2\Theta \\ \tau_4 - t_3 + t_2 - \tau_1 &< 2\Theta - 2D \end{aligned} \tag{18}$$

Equation (18) nicely reflects an intuitive argument: a larger Θ enables a larger NTS query transit, and vice versa for a larger D. Therefore, in all, the Algorithm (4) limits how much a clock can be changed or rejects an NTS query if it takes too long. Any spoofing of the synchronization procedure will serve to deny authentication (by denying service) preventing the authentication of forgeries.

Algorithm 4 A TESLA onboard clock synchronization procedure safe from man-in-the-middle attacks and denial of service.

```
1: Receiver computes a nonce $k$.
2: Receiver sends a secure message $m_1 = (k, s_1^{\text{receiver}})$ and records $\tau_1$ at the moment of sending $m_1$. $s_1^{\text{receiver}}$ is an authentication signature on $(k, )$. $\tau_1$ is held in confidence from the server. $s_1^{\text{receiver}}$ is an authentication signature on $(k, )$. $m_1$ is not sent at a fixed frequency (e.g., at the top of a minute) to minimize leakage of $\tau_1$.
3: Provider records $t_2$, the time of receipt of the message $m_1$.
4: Provider sends a secure message $m_2 = (k, t_2, t_3, s_2^{\text{receiver}})$ back to receiver, where $s_2^{\text{receiver}}$ is an authentication signature on $(k, t_2, t_3)$.
5: Receiver records $\tau_4$ at the moment of receipt of message $m_2$.
6: The measured clock drift is $\hat{\theta} = \frac{1}{2}(\tau_1 - t_2 - t_3 + \tau_4)$ assuming that the transit time of $m_1$ and $m_2$ are the same.
7: Let $\Theta$ be the TESLA bit-commitment to TESLA key release time.
8: Let $D$ be an estimated upper-bound on the on-board clock drift before the next application of this Algorithm 4.
9: if $\tau_4 - t_3 + t_2 - \tau_1 > 2\Theta - 2D$ then
10:     NTS Query took too long. There is no secure correction $\theta^*$
11:     return ERROR: Synchronization failed.
12: end if
13: $\theta^* = \hat{\theta}$
14: return $t_i \leftarrow t_i - \theta^*$ Apply secure clock correction.
```

While we derive the two bounds on θ^*, we only include the NTS query length from Equation (18). This is because, for this work, we care about both the lower and upper limits on θ^* to minimize the likelihood a receivers clock drift-*leading* into a denial of service in between synchronizations. However, if one adjusts their TESLA receiver authentication operations, one might only care about the lower bound on θ^*. Since only a one-sided bound on θ^* is possible because forgeries are only possible when receiver clocks lags, we provide Algorithm 5.

1. Monitored GNSS Safe Zones

As an alternative to Algorithm 3 and Algorithm 4, one could implement low-cost monitoring at airports to create safe zones known to be without GNSS spoofing (Taleghani et al., 2022). When an aircraft is known to be in a safe zone, it may allow its onboard clock to be reset to the measured GNSS time. Then, after takeoff, its onboard clock is not allowed to reset itself to

Algorithm 5 A TESLA onboard clock synchronization procedure safe from man-in-the-middle attacks and denial of service that considers only a one-sided bound on the synchronization correction.

```
1:  Receiver computes a *time-independent* nonce $k$.
2:  Receiver sends a secure message $m_1 = (k, s_1^{\text{receiver}})$ and records $\tau_1$ at the moment of sending $m_1$. $s_1^{\text{receiver}}$ is an authentication signature on $(k, )$. $\tau_1$ is held in confidence from the server. $s_1^{\text{receiver}}$ is an authentication signature on $(k, )$. $m_1$ is not sent at a fixed frequency (e.g., at the top of a minute) to minimize leakage of $\tau_1$.
3:  Provider records $t_2$, the time of receipt of the message $m_1$.
4:  Provider sends a secure message $m_2 = (k, t_2, t_3, s_2^{\text{receiver}})$ back to receiver, where $s_2^{\text{receiver}}$ is an authentication signature on $(k, t_2, t_3)$.
5:  Receiver records $\tau_4$ at the moment of receipt of message $m_2$.
6:  The measured clock drift is $\hat{\theta} = \frac{1}{2}(\tau_1 - t_2 - t_3 + \tau_4)$ assuming that the transit time of $m_1$ and $m_2$ are the same.
7:  Let $\Theta$ be the TESLA bit-commitment to TESLA key release time.
8:  Let $D$ be an estimated upper-bound on the on-board clock drift before the next application of this Algorithm 4.
9:  $\theta^* \leftarrow \min(\hat{\theta}, -(t_2 - \tau_1) - D + \Theta)$ Clip $\hat{\theta}$ to provide a TESLA-secure clock correction in between synchronizations.
10: **return** $t_i \leftarrow t_i - \theta^*$ Apply secure clock correction.
```

the GNSS timing measurement. The low-cost monitoring must itself contain a secure clock that is secured via Algorithm 3. However, if the low-cost monitoring service is spoofed and does not notify airport authorities or departing aircraft, the aircraft would be susceptible to spoofing.

V. SOME SERVER CONSIDERATIONS

NTP specifies several layers of clocks that eventually synchronizes to atomic time. The best server, for the purpose of securing TESLA for GNSS users, would have its own direct connection to an atomic clock or the collection of clocks that determine atomic time. Any server, for the purpose of securing TESLA for GNSS users, should be synchronized to a clock as closely connected as possible to the atomic time clocks to minimize the risk that a server is actively attacking a receiver. Or at a minimum, that server should query many level-1 NTP servers to mitigate the single-server-attacker risk. Hopefully, the catastrophic security risk of spoofed aircraft and autonomous vehicles will motivate priority access of these vehicles to servers as closely coupled to atomic time as possible.

VI. CONCLUSION

In this work, we provide the algorithms required to certify and re-synchronize an onboard clock to satisfy TESLA's loose-time synchronization requirement, even in the presence of an adversary engaged in man-in-the-middle delays with TESLA-secure GNSS. Receivers will either need access to a service that provides Algorithm 3 or a modified NTS server that provides Algorithm 4. Receivers should also use an isolated onboard clock to perform TESLA cryptographic operations.

ACKNOWLEDGMENTS

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

REFERENCES

(2021). Galileo open service navigation message authentication (osnma) user icd for the test phase. volume 1.

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

Anderson, J., Lo, S., and Walter, T. (2022a). Cryptographic ranging authentication with tesla, rapid re-keying, and a prf. In *Proceedings of the 2022 International Technical Meeting of The Institute of Navigation*, pages 43–55.

Anderson, J., Lo, S., and Walter, T. (2022b). 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, 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.

Fernandez-Hernandez, I., Walter, T., Neish, A., and O'Driscoll, C. (2020). Independent time synchronization for resilient gnss receivers. In *Proceedings of the 2020 International Technical Meeting of The Institute of Navigation*, pages 964–978.

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

Franke, D. F., Sibold, D., Teichel, K., Dansarie, M., and Sundblad, R. (2020). Network Time Security for the Network Time Protocol. RFC 8915.

Götzelmann, M., Köller, E., Semper, I. V., Oskam, D., Gkougkas, E., Simon, J., and de Latour, A. (2021). Galileo open service navigation message authentication: Preparation phase and drivers for future service provision. pages 385–401.

Martin, J., Burbank, J., Kasch, W., and Mills, P. D. L. (2010). Network Time Protocol Version 4: Protocol and Algorithms Specification. RFC 5905.

Neish, A. (2020). *Establishing Trust through Authentication in Satellite Based Augmentation Systems*. PhD thesis, Stanford, CA.

O'Driscoll, C., Keating, S., and Caparra, G. (2020). A performance assessment of secure wireless two-way time transfer. pages 3938–3951.

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

Shankarkumar, V., Montini, L., Frost, T., and Dowd, G. (2017). Precision Time Protocol Version 2 (PTPv2) Management Information Base. RFC 8173.

Taleghani, L., Rothmaier, F., Chen, Y.-H., Lo, S., Walter, T., Akos, D., and Gattis, B. G. (2022). Low cost rfi monitor for continuous observation and characterization of localized interference sources. In *Proceedings of the 2022 International Technical Meeting of The Institute of Navigation*, pages 1216–1245.