Addressing a Critical Vulnerability in upcoming Broadcast-only 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

Herein, we delineate and suggest mitigations for a critical security attack involving the time synchronization requirement of *any* broadcast-only Timed-Efficient Stream Loss-tolerant Authentication ("TESLA") scheme, including those in development for many Global Navigation Satellite Systems ("GNSS"). TESLA's bandwidth efficiency and loss-tolerant properties are advantageous, even necessary, to provide GNSS cryptographic authentication security on data channels and ranging signals. TESLA presumes a loose-time synchronization assumption, and receivers must externally (i.e., via an out-of-band channel) verify this assumption (e.g., at startup, routine maintenance, routinely) to assert authentication security. However, the combination of (1) TESLA's adaptation to the broadcast-only context and (2) the current network timing synchronization standards lends to an attack that could allow receivers to accept forgeries. The time synchronization protocol must be modified to mitigate this threat. We show this attack's concrete feasibility using data from a study conducted on a Network Time Protocol ("NTP") server.

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 provides users with 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.

Two methods exist in the literature to protect the spreading code with authentication cryptography: (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). One method is already in practice to protect the navigation data with authentication cryptography (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 the 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. Moreover, a one-way signal cannot achieve secure time-synchronization (Narula and Humphreys, 2018). The loose-time synchronization requirement specifies a bound on how much a receiver clock *lags* the provider clock. The description of TESLA provides a simple protocol to bootstrap loose-time synchronization (Perrig et al., 2005). Fernandez-Hernandez et al. (2020) discusses methods and requirements assuming limited clock synchronization, and Anderson et al. (2022c) discusses

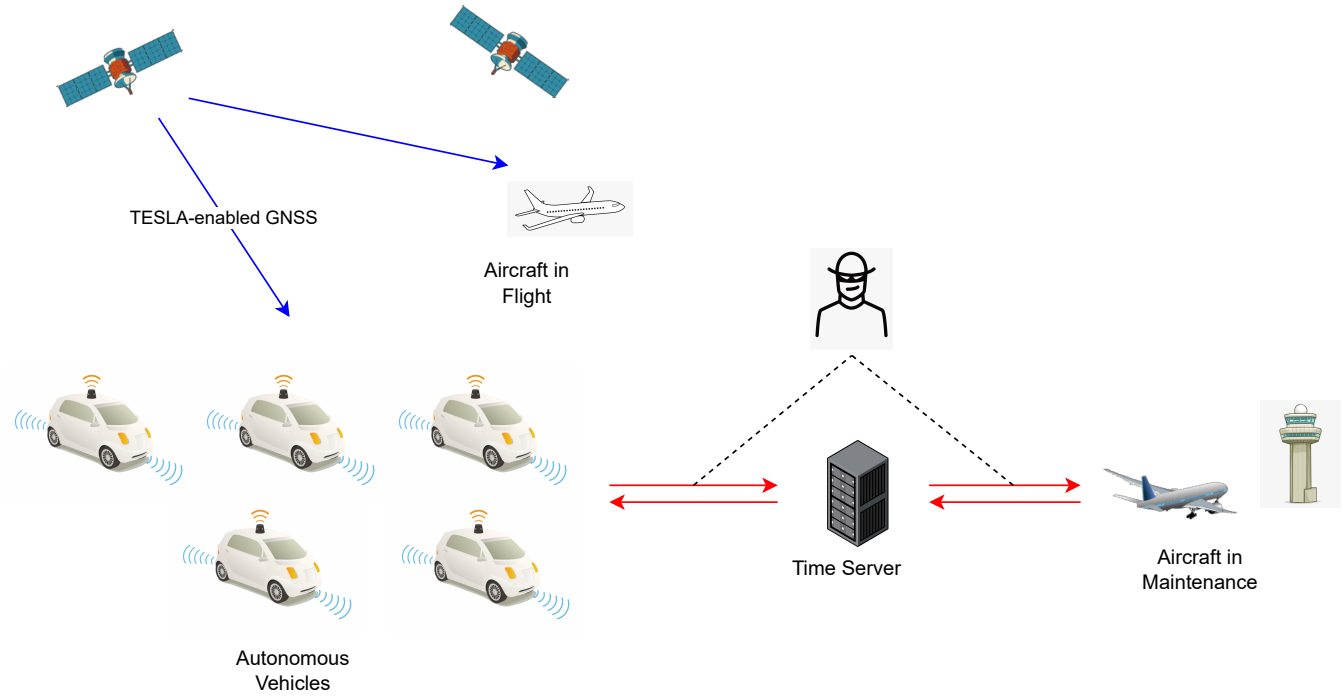

Figure 1: A conceptual diagram of our expectation of the future. We expect certain systems will need to use the authentication provided by current and future TESLA-based GNSS services. Some receivers, such as autonomous cars, will have continuous access to a time server allowing for continuous checking. Other receivers, such as aircraft, might not have continuous access to a server but will use periodic maintenance and more accurate clocks between maintenance sessions to enforce the timing requirement. An adversary can manipulate the receipt times of synchronization signals from a synchronization server.

how to securely synchronize in the GNSS context and how to implement provably-safe loose-time synchronization to deter man-in-the-middle attacks. This method's applicability is principally for upcoming, practical broadcast-only TESLA, whereas previous literature includes a generalized approach (Narula and Humphreys, 2018). This work discusses how a vulnerability is still present in Anderson et al. (2022c) related to the current standards of internet time transfer.

1. Future Vulnerability of Concrete Users

To provide a concrete and plausible scenario and show the importance of this work, we discuss our vision and provide the conceptual diagram of Figure 1. We envision a future where systems, particularly those connected with the safety of life, will want (or be required) to enforce the authentication of current GNSS signals and future GNSS signals. To assert the security TESLA provides, all must verify that their clocks satisfy the TESLA loose-time synchronization requirement. Because of the broadcast-only nature of GNSS and how simply repeating a GNSS signal could be accepted by the receiver as a lagging signal, the TESLA loose-time synchronization requirement must be enforced outside GNSS with an additional connection.

Users might have continuous access to an internet connection (e.g., autonomous cars) or not (e.g., aircraft). Receivers may have access to onboard clocks of varying drift rates. For instance, we expect receivers without an internet connection will have access to a low-drift clock and need to conduct clock checks on an infrequent base, whereas receivers with an internet connection can arbitrarily check their clock at any time.

2. Secure Time Synchronization

In previous work, we delineated a time synchronization procedure for GNSS TESLA that is secure against man-in-the-middle delays (Anderson et al., 2022c). In this section, we summarized the results of that work.

First, we begin with Network Time Security ("NTS") from Franke et al. (2020), which is the Network Time Protocol from Martin et al. (2010) augmented with cryptographic authentication security so that the *content* of the messages is protected. We provide NTS with Algorithm 1 and Figure 2. Previous work has evaluated their application to TESLA GNSS schemes (O'Driscoll et al., 2020). Using the procedure from Anderson et al. (2022c), we derive the man-in-the-middle spoofable range

Algorithm 1 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 τ_1 is the time receiver recorded at the moment of sending m_1 and s_1^{receiver} is an authentication signature on (k, τ_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^{receiver} is an authentication signature on (k, τ_1, t_2, t_3).
5: Receiver records τ_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.

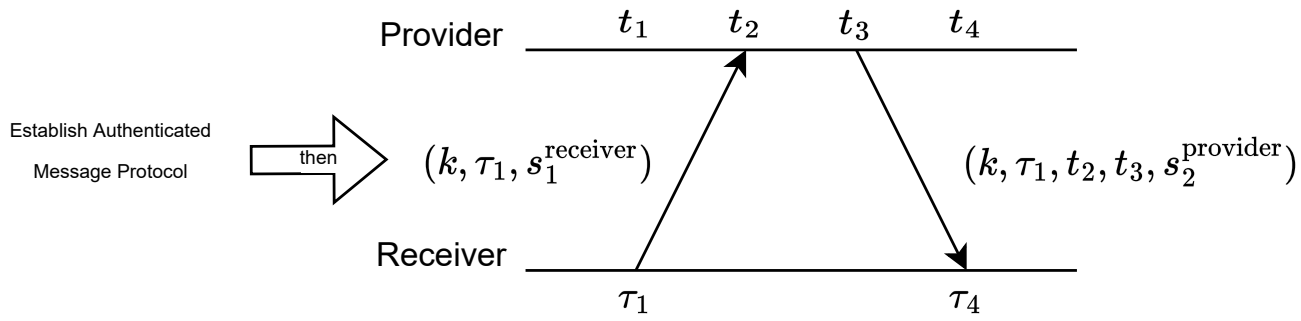

Figure 2: A conceptual diagram of Algorithm 1. The diagram depicts increasing time from left to right in the provider and receiver clock frames and the messages shared between them. The protocol presumes provider and receiver have already established a cryptographic authentication instance to protect the *content* of the messages transmitted. The protocol is susceptible to adversary-induced delays in message transmission.

of receiver clock biases, reported here in Equation (1). In Equation (1), as in Algorithm 1, θ is the clock bias deduced via executing the NTS query (with $\theta < 0$ indicating a lagging receiver clock), τ_1 is the timestamp of the first request message, t_2 is the arrival time to the server of the request message, t_3 is the departure time of the response message, and τ_4 is the arrival time of the response message to the receiver. By delaying the request and response messages, an adversary may cause a receiver to deduce a clock bias in the range of Equation (1), but not beyond those bounds.

$$-(t_2 - \tau_1) < \theta < \tau_4 - t_3 \tag{1}$$

As shown in Figure 2, the Greek letter τ indicates a measurement in the biased receiver frame, whereas the Roman letter t indicates the event in the true frame. Times measured are related to events in the true frame via a measurement equation involving the clock bias. For instance, $\tau_1 = t_1 + \theta$. We also assume that θ does not change over the short time of an NTS request and response return trip.

Let Θ be the TESLA key disclosure delay for the scheme. For GNSS, Θ is a constant parameter set and used by the entire constellation and all of the receivers. A receiver enforcing the authentication security of GNSS TESLA will accept forgeries if $\theta < -\Theta$. In Anderson et al. (2022c), we use Equation (1) to derive a safety check and a safe synchronization procedure. In this work, we describe another vulnerability to time synchronization. The vulnerability derives from the combination of two aspects of the context.

First, TESLA is modified for the GNSS context to accommodate the multicast context. In the original TESLA design, the key-disclosure delay can be set individually by the communicating parties. If a man-in-the-middle interferes, the key disclosure delay increases, possibly resulting in a denial of service. However, forgeries will never be accepted. For GNSS, the system specifies the key-disclosure delay common among all users. There is no per-user accommodation (e.g., in the event of a poor or obstructed network connection for time-synchronization). If a receiver clock were lagging more than Θ, and a malicious actor knew about that, then a malicious actor could use a disclosed TESLA key to generate forgeries that the receiver would accept. Even though Anderson et al. (2022c) provides a procedure to protect against this scenario, the user is still vulnerable when the adversary has *knowledge* of the user's potentially lagging clock.

Second, NTS considers but does not require confidentiality (Franke et al., 2020), relegating to providers who want to restrict access to customers or users to protect their anonymity. Clock errors result from stochastic processes, meaning that there will

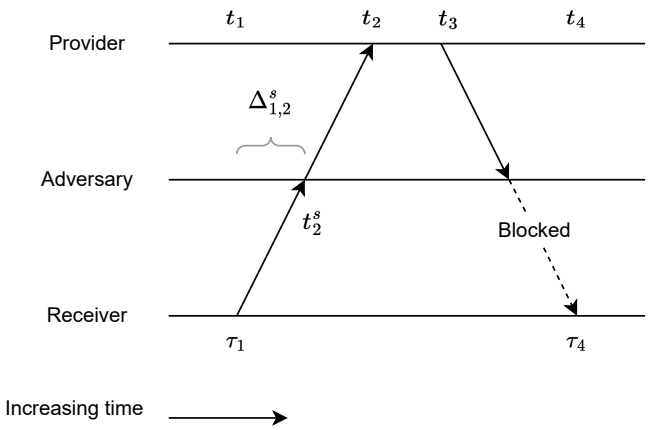

Figure 3: A conceptual diagram that depicts the structure of the attack on the vulnerability. t_1 through t_4 and τ_1 through τ_4 have the same definitions from Figure 2. $\Delta^s_{1,2}$ is the network transit time from the receiver to the adversary, and t^s_2 is the arrival time to the adversary of that message. As long as the return trip is blocked, the receiver will not be able to have any knowledge about the status of its clock.

be clocks that will drift into violating TESLA's loose-time synchronization requirement among the population of clocks. In the GNSS context with the system-wide key-disclosure delay, we now observe a critical security reason NTS time synchronization must be confidential or the NTS standard modified. An adversary observing non-confidential time-synchronization traffic will observe users not compliant with TESLA's loose-time synchronization requirement and know they will accept forgeries. In the context of GNSS-enabled vehicles, spoofing *any* specific vehicle might be sufficient for a malicious actor's nefarious goals, meaning neglecting to consider disclosures of stochastic vulnerabilities could render the security provided by cryptographic systems ineffective.

3. Threat Model

The Dolev-Yao Model provides a formal model to prove properties involving interactive cryptographic systems (Dolev and Yao, 1983). In this work, when we say *adversary*, we mean a malicious actor with all the capabilities specified by Dolev-Yao. In essence, the adversary can overhear, intercept, and create messages but cannot break the underlying cryptographic primitives. Succinctly, the attacker carries the message. There are a variety of threat models from the literature, including those targeting the context of vehicles (Ponikwar et al., 2016). Since we provide a mitigation that should hold up against a Dolev Yao adversary, we allow that model to form our adversary. However, for this work, an attacker needs only to be able to eavesdrop and block messages, which is a subset of the Dolev-Yao capabilities.

In Anderson et al. (2022c), the time synchronization procedure protects against man-in-the-middle, delay-capable adversaries. While the time-synchronization messages in that work would be protected with authentication cryptography, prohibiting adversaries from forging messages, there is no protection against message arrival times being delayed. Cryptography does not protect against delays (Annessi et al., 2018). The procedure of Anderson et al. (2022c) ensures that if delays were introduced, the synchronization procedure would fail, limiting attacks to denial of service and preventing forged messages. However, in this work, we acknowledge that the procedure notifies adversaries that a receiver is currently vulnerable, and then adversaries can block the notification of the need for correction.

II. DESCRIPTION OF ATTACKS

As an overview of the attack, Figure 3 diagrams the attack with the mathematical quantities for our description. t_1 through t_4 and τ_1 through τ_4 are the same from NTS. We assume that the adversary and provider are perfectly synchronized to actual time, whereas the receiver may have an offset of θ that an adversary will hope is $\theta < -\Theta$. The receiver will initiate an NTS query and note τ_1 when it sends its initial request. The adversary will observe the NTS request and note its incoming time, which we denote as t^s_2. The adversary can allow the NTS request to continue to the provider, but the adversary must block the return journey. If the NTS response is allowed back and then recorded at τ_4, the receiver can correct its clock or know that the round trip time disqualifies using the query.

To derive a model of the attack scenario, we start with the events in the adversary frame with Equation (2). Substituting the

measurement equation $\tau_1 = t_1 + \theta$, we arrive at Equation (3).

$$t_2^s = t_1 + \Delta_{1,2} \tag{2}$$
$$t_2^s = \tau_1 - \theta + \Delta_{1,2}$$
$$\theta = -(t_2^s - \tau_1) + \Delta_{1,2} \tag{3}$$

1. Forming a Model on Traffic

At a high level, the adversary will use (a) evidence from the NTS traffic and (b) a model of the network traffic transit time to decide whether a specific receiver is vulnerable. In Equation (3), $-(t_2^s - \tau_1)$ is (a), and $\Delta_{1,2}$ is (b). For (a), this is a search for outliers. We provide a simple estimation procedure of how long an adversary will need to search before finding one in Section III.1. For (b), a simple and conservative approach will suffice. Using a study of NTP network traffic, such as from Novick and Lombardi (2015), we can come up with a conservatively correct but somewhat heuristic bound of Equation (4). We justify our bound since NTP round-trip-time is generally around 100 ms and expect the $\Delta_{1,2}$ to be around 50 ms. Nevertheless, an adversary can experientially adjust these considerations to the context.

$$\Pr(\Delta_{1,2} < 100\text{ms}) > 0.99 \tag{4}$$

Substituting in Equation (3) into Equation (4), we arrive at Equation (5).

$$\Pr(\theta < -(t_2^s - \tau_1) + 100\text{ms}) > 0.99 \tag{5}$$

Lastly, we present a practical test that an adversary could use to determine if a specific receiver is vulnerable to accepting forgeries. If $-(t_2^s - \tau_1) + 100\text{ms} < -\Theta$, then it is very likely that the receiver clock is lagging sufficiently to accept forgeries and the adversary has identified a vulnerable receiver.

2. Attack on Unencrypted Traffic

Algorithm 2 provides our explicit attack that an adversary, with the abilities from Section I.3, could conduct to find vulnerable receivers.

Algorithm 2 Attack on Unencrypted Traffic by Adversary

1: Adversary begins observing the time-synchronization traffic of the vehicle class associated with a specific location to search for a vulnerable receiver.
2: Adversary forms a model on the network traffic transit time from the receiver to Adversary (e.g., Equation 5).
3: Adversary eavesdrops on the NTS request $m_1 = (k, \tau_1, s_1^{\text{receiver}})$ where τ_1 is the time receiver recorded at the moment of sending m_1 and s_1^{receiver} is an authentication signature on (k, τ_1).
4: Adversary records t_2^s, the time of receipt of the eavesdropped message m_1.
5: Using its internet traffic model, t_1 and t_2^s, Adversary observes traffic until it observes a receiver believed to have a $\theta < -\Theta$.
6: Adversary blocks the return NTS response $m_2 = (k, \tau_1, t_2, t_3, s_2^{\text{receiver}})$ and subsequent return NTS responses back to the receiver believed known to accept forgeries so that that receiver's clock is never corrected.
7: Adversary listens to the authentic GNSS signal for a disclosed TESLA key, generates a forged message and broadcasts it to the vulnerable receiver.

3. Attack on Encrypted Traffic

If τ_1 were omitted or obscured (e.g., with encryption) by the protocol, the observation of the NTS request could still reveal knowledge about the status of the receiver clock. For instance, if the adversary knows that the receiver will conduct NTS requests at a fixed schedule (e.g., at the top of the hour), then the adversary can reasonably guess τ_1 even if it is not observable in the message content. Algorithm 3 provides our explicit attack that an adversary, with the abilities from Section I.3, could conduct to find vulnerable receivers.

III. PRESENT FEASIBILITY OF ATTACK

Via private correspondence, the authors of Sherman and Levine (2016) graciously gave us access to the τ_1 and t_2 values from a study regarding NTP server usage. The data provided does not contain identifiable information among the NTP users (just the

Algorithm 3 Attack on Traffic with missing τ_1 by Adversary

1: Adversary begins observing the time-synchronization traffic of the vehicle class associated with a specific location to search for a vulnerable receiver.
2: Adversary forms a model on the network traffic transit time from the receiver to Adversary (e.g., Equation 5).
3: Adversary forms a model on the the incoming traffic's τ_1 (e.g., based on its belief that the observed m_1 was sent on a fixed schedule).
4: Adversary records t_2^s, the time of receipt of the eavesdropped message m_1.
5: Using its internet traffic model, t_1 and t_2^s, Adversary observes traffic until it observes a receiver believed to have $\theta < -\Theta$.
6: Adversary blocks the return NTS response $m_2 = (k, \tau_1, t_2, t_3, s_2^{\text{receiver}})$ and subsequent return NTS responses back to the receiver believed known to accept forgeries so that that receiver's clock is never corrected.
7: Adversary listens to the authentic GNSS signal for a disclosed TESLA key, generates a forged message and broadcasts it to the vulnerable receiver.

time stamps in the NTP messages). Using that data and a conservative NTP round trip time model, we found large numbers of users that likely had TESLA-non-compliant clocks were they using a future TESLA-enabled GNSS. We emphasize the careful interpretation of the meaning of these results, limiting our conclusion to just that our attack is immediately feasible.

The population of timestamps reflects the varying inconsistencies of users not faithfully implementing the NTP protocol. We provide the distribution of τ_1 in Figure 4. Among the approximate 12.7 million requests analyzed, about 26% transmitted a null τ_1 (i.e., $\tau_1 = 0$), and about 5% transmitted an integer τ_1. While the data was collected in 2015, there were a large number of τ_1 with calendar year of 1970 (likely from the unix epoch). Moreover, a substantial section of users form an apparent uniform floor. We suspect this uniform band results from users transmitting a random τ_1, perhaps as a nonce to differentiate repeated requests. Lastly, we note that we observe elevated bursts, such as the top of the hour and top of the minute, which could reveal information about specific requests' τ_1.

The mode of the distribution from Figure 4 forms around the correct time. We provide a close-up of the distribution of $-(t_2-\tau_1)$ values around the correct time in Figure 5. We find a substantial number of users who, ostensibly, are providing a faithful τ_1 value, and we observe a substantial incidence of users whose clocks are likely lagging behind. In Figure 5, we annotate which users, if they were listening to a future authenticated SBAS concept with $\Theta = 6$ (Anderson et al., 2021), would accept forgeries.

From these results, we make the following limited claim. Since we, the authors, using data provided by the back-end of an NTP server, are able to observe clocks of users who would, with reasonable probability, accept forgeries in a TESLA-enabled GNSS were they using that TESLA-enabled GNSS, the NTP server would certainly be able to identify these vulnerable users. Moreover, given our threat model and how the traffic is unencrypted, an adversary would also be able to identify vulnerable receivers immediately. The adversary's game is to wait for a vulnerable user. We show the reasonable plausibility of exploiting the vulnerability concretely we point out. Therefore, this vulnerability should concern those wishing to provide authentication security via a TESLA-enabled GNSS.

We must limit the interpretation beyond a demonstration. Interpretation of the actual data should be limited given the wide variety of actual use of τ_1. Moreover, we do not warrant that this traffic is representative of future synchronization traffic for a future TESLA-enabled GNSS. We would have no indication whether the time τ_1 is measured from the clock used to enforce the TESLA-loose-time synchronization requirement or a separate clock. This is especially germane because prior literature on TESLA for GNSS suggests having an isolated clock perform the checks. Furthermore, we do not expect the drift rates of receiver clocks to be consistent with the drift rate of clocks from this NTP traffic.

1. Poisson Model

An adversary must wait to find a vulnerable clock. Depending on the receiver clock drift rate, the incidence might be similar to that of rare outliers. However, by characterizing the data, an adversary can reasonably model how long they will expect to wait. We suggest a Poisson model.

Suppose that this data came from users using a TESLA-enabled SBAS with $\Theta = 6$ (e.g., as in Anderson et al. (2021)). Considering the wide variety of τ_1 observed, we assume incoming NTP requests received within 20 seconds of real-time contain τ_1 that reflects their real clock. Therefore, among those requests, requests with a $-(t_2 - \tau_1) < -6.1$ should accept forgeries with reasonable probability. Taking the t_2 from those offending requests, we form a maximum-likelihood-estimated Poisson model on the incoming incidence of t_2 of target receivers. We calculate a Poisson model with $\lambda = 1.1$ seconds. Therefore, on expectation, for this data, the adversary should expect to encounter a vulnerable clock after 1.1 seconds of observation. There were about 7000 requests per second in this data set, which handily suggests that vulnerable clocks are certainly outliers. Moreover, we also note that we expect vulnerable clocks among those who did not send a faithful τ_1.

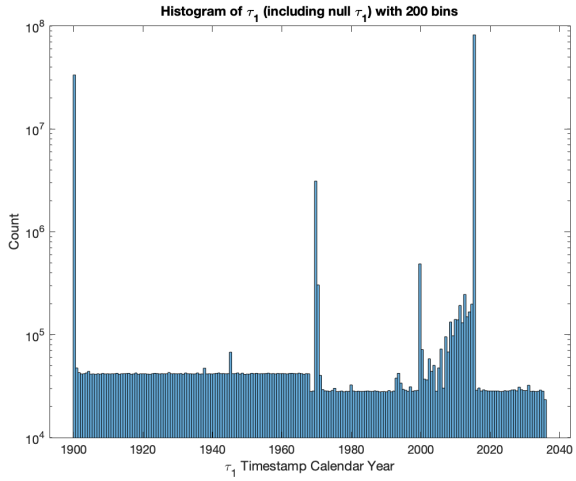

Figure 4: The distribution of τ_1 observed in Sherman and Levine (2016) as a histogram with 200 bins. Data reflects an 18-hour section of the overall usage study. We note the NTP epoch is 1 January 1900. We observe the varying levels of faithful implementation of the NTP standard, including the following: (1) those who transmit $\tau_1 = 0$ with the left-most bin; and, (2) those who transmit a time around 1970, possibly related to the Unix epoch. We observe a uniform floor for which we suspect results from users transmitting a uniformly random τ_1, possibly as a nonce. The mode of the distribution occurred at the correct time in 2015.

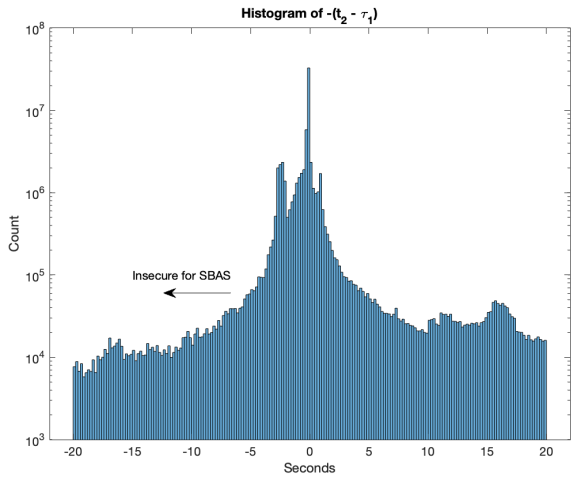

Figure 5: The distribution of $-(t_2 - \tau_1)$, zoomed into the mode of the distribution around near-synchronized clocks. Neglecting a conservative 100ms NTP request transmission time, a negative value indicates that a particular clock is lagging. If the user were to use a TESLA-enabled GNSS, a value less than $-\Theta$ indicates that the user would accept forgeries. We annotate where future TESLA-enabled SBAS users would accept forgeries.

IV. THREAT MITIGATIONS

We propose several mitigations of varying levels of implementation difficulty. Solutions that guarantee that no information about τ_1 is leaked will suffice. First, receivers should omit τ_1 from the protocol. We already observed users doing this in Section III. Second, providers and receivers could encrypt the NTS traffic. However, if the synchronization server had a security vulnerability or was collaborating with an adversary, the server could reveal τ_1, rendering this mitigation ineffective.

Receivers should also never submit an NTS request predictably. For instance, a continuously connected receiver should not send an NTS request at a fixed schedule (e.g., top of the hour). There are several ways that this could be implemented. For instance, suppose that a receiver wanted to synchronize every T seconds. Every second, the receiver could flip a biased coin that succeeds with probability $\frac{1}{T}$. On expectation, the receiver should synchronize at the desired frequency while ensuring that each synchronization event is independent of the last, obscuring the adversary's ability to estimate τ_1. Non-continuously connected receivers should not send an NTS request at the top of a minute or the top of a second, revealing information about when an operator initiates a synchronization.

Lastly, among the work on TESLA for GNSS, it has been suggested that the clock used for TESLA be isolated or held in confidence on-device from other onboard clocks (Anderson et al., 2021). Suppose the communication aspect of the onboard software does not have access to the time estimate used for TESLA security checks. In that case, we ensure that τ_1 is never disclosed mitigating this vulnerability.

V. CONCLUSION

In this work, we describe a security vulnerability for future TESLA-enabled GNSS receivers while attempting to synchronize to fulfill the loose-time synchronization requirement. We concretely show the plausibility of this attack using data provided by an NTP usage study. And we also suggest modifications to the NTS protocol to ensure that the vulnerability is not exploited.

ACKNOWLEDGMENTS

We gratefully acknowledge the support of the FAA Satellite Navigation Team for funding this work under Memorandum of Agreement #: 693KA8-22-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., Lo, S., and Walter, T. (2022c). Time synchronization for tesla-based gnss-enabled systems.

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.

Annessi, R., Fabini, J., Iglesias, F., and Zseby, T. (2018). Encryption is futile: Delay attacks on high-precision clock synchronization.

Dolev, D. and Yao, A. (1983). On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208.

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.

Narula, L. and Humphreys, T. E. (2018). Requirements for secure clock synchronization. *IEEE Journal of Selected Topics in Signal Processing*, 12(4):749–762.

Novick, A. N. and Lombardi, M. A. (2015). Practical limitations of ntp time transfer. In *2015 Joint Conference of the IEEE International Frequency Control Symposium & the European Frequency and Time Forum*, pages 570–574.

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.

Ponikwar, C., Hof, H.-J., Gopinath, S., and Wischhof, L. (2016). Beyond the dolev-yao model: Realistic application-specific attacker models for applications using vehicular communication.

Sherman, J. A. and Levine, J. (2016). Usage analysis of the nist internet time service. *Journal of Research of the National Institute of Standards and Technology*, 121:33.