Cryptographic Authentication for GPS Communications

Jiayang Wang, *Stanford University*
Yiyang Mu, *Stanford University*
Yong Lin He, *Stanford University*

I. INTRODUCTION

In an era of exclusively widespread use of GPS technology, it is more important than ever to have robust measures in place to ensure the integrity and security of GPS communications. The civilian GPS L1C signals are publicly available, and thereby can be synthesized and manipulated by anyone with the right skills and equipment, possibly for spoofing purposes. Potential motivations for spoofing can be diverse, ranging from fraud to espionage to terrorism. The spoofers can manipulate the position and/or time estimation and mislead a target receiver by altering or delaying the GPS signals processed by the target receiver.

The vulnerabilities in the GPS transmission, including spoofing and jamming, compromise the integrity and security of the location data. Ensuring the integrity of location-based services is crucial for applications like emergency response. When emergency services depend on accurate location data for timely and precise interventions, any compromise in the integrity of this information can have far-reaching consequences. As we rely more on GPS for navigation, emergency services, and so on, the consequences of compromised location data become increasingly severe. Additionally, potential threats to data accuracy from intentional attacks pose risks across industries. From logistics and transportation to finance and healthcare, we are facing threats to data accuracy from intentional spoofing attacks on GPS communications.

The vulnerabilities we aim to address not only threaten the accuracy of location-based data but also enable malicious activities with severe consequences. Our goal is to enhance GPS communications with cryptographic authentication, preventing GPS signal spoofing or message alteration and, therefore enhancing the robustness and security of applications that rely on accurate GPS data. One potential solution to address these vulnerabilities is Navigation Message Authentication (NMA). In this paper, we explore NMA for GPS civilian signals. We take the Time Efficient Stream Loss-tolerant Authentication (TESLA) as a baseline and address its vulnerability by introducing an additional certification service by a trusted third-party as an improvement. This certification process ensures a more secure and efficient process for validating the authenticity of the public key used in the TESLA algorithm.

II. RELATED WORK

GPS L1C signal is a modernized civilian GPS signal that is publicly accessible and broadcast on the L1 frequency. It comprises a pilot signal, $L1C_P$ that carries no navigation data but is modulated onto the L1 carrier using a time-multiplexed binary offset carrier, and a data signal, $L1C_D$ that carries the navigation message (Anderson et al., 2017)(Directorate, 2013). Distinct spreading codes (i.e., Pseudo-Random Noise) are employed to modulate both the pilot and data signal components. Each spreading code has a chipping rate of 1.023 MHz and a sequence length of 10230 chips, leading to chip sequences that cyclically repeat 100 times per second (Anderson et al., 2017).

As of today, all GNSS civilian signals, including GPS L1C signal, adopt clear and open interface specifications in signal transmission. The receivers will trust any input that confronts the specifications of GNSS signals. Additionally, the GNSS signal has extremely low power levels, rendering it easily synthesizable or manipulable (Petovello, 2017). The openness of GPS civilian signals, therefore, leads to various security issues in GPS communications. Any individual with the right skills and equipment can synthesize and manipulate GPS signals, possibly for spoofing purposes (Anderson et al., 2017). Spoofing refers to the deliberate manipulation of GPS signals with the aim of deceiving target receivers in their estimations of position or time. The spoofers can manipulate the position and/or time estimation and thereby mislead a target receiver by altering or delaying the GPS signals processed by the target receiver. Potential motivations for spoofing can be diverse, including fraud, espionage, terrorism, or even evading detection and tracking by an employer (Günther, 2014). There are two fundamental types of spoofing attacks. The one is a replay of the authentic GPS civilian signals, so-called “Relay Attacks”. The spoofers capture and then fraudulently resend or delay the authentic, open GPS civilian signals to fool the target receivers. Replay attacks can be attractive to the spoofers because they do not even require advanced decryption skills to spoof the receivers. The spoofers are, however, restricted to the delayed signals (Günther, 2014). This implies that a spoofer can only manipulate the target receiver using authentic GPS navigation messages from earlier timestamps. However, achieving a more substantial deception, such as misleading the target receiver to have entirely incorrect position and/or time estimates, would be challenging. The other form of spoofing is sending fully synthetic GPS civilian signals to the target receiver. The current open GNSS services do not employ any cryptographic or authentication processes. Hence, the open GNSS signals can be manipulated by anyone with the right skills and equipment (Anderson et al., 2017) (Günther, 2014). A target receiver can hardly distinguish the spoofing signals from the real GNSS satellite signals (Günther, 2014). Successful attempts for spoofing GPS civilian signals have been demonstrated in (Kerns et al., 2014) and (Divis, 2013). The risks for spoofing attacks motivate a demanding need to ensure the security and integrity of the GPS navigation messages.

To address these vulnerabilities in GPS communications, researchers have proposed various message authentication solutions to make GPS signal transmission secure and robust to spoofing attacks. Navigation Message Authentication (NMA) uses Cryptography. It mitigates the risk of spoofing attacks by providing the user with evidence that the received signal is from a reliable source (i.e., GNSS satellites) (Anderson et al., 2017). In NMA, the sender generates an authentication signature from a message using a secret key. The sender then transmits both the authentication signature and message to the receiver, who will use a key to verify both the message and the authentication. According to the verification results, the receiver can tell if the received message is identical to the transmitted one (i.e., the integrity of the message) and if the authentication signature was generated by the transmitter using its secret key (i.e., security of the message). The authentication signatures could be generated in two ways: symmetric key and asymmetric key techniques. In symmetric key techniques, a shared secret key between the sender and receiver is used for both authentication message generation and authentication signature verification. In asymmetric key techniques, a secret key is divided into a private key, exclusive to the transmitter, and a public key that is accessible to anyone. The receiver uses the public key to verify the received message and authentication signature, while the transmitter uses the private key to generate the authentication message (Petovello, 2017). Each of the NMA approaches, however, comes with some issues. For the symmetric case, it is very difficult to securely distribute the "private" secret key to all users without revealing the key to spoofers. In an asymmetric system, an additional authentication mechanism is needed for the receiver to ensure that the public key does come from the GNSS system operators. Additionally, compared to symmetric key encryption, asymmetric encryption is much more computationally intensive and requires a much longer key for the same level of security (Petovello, 2017).

GPS proposed the Chips-Message Robust Authentication (Chimera), which is a hybrid NMA and spreading code authentication techniques for the GPS L1C signal (Petovello, 2017). It jointly authenticates both the navigation message and the spreading code of a GPS L1C signal. A key advantage of Chimera is it combines NMA with spreading code components to authenticate the time of transmission. (Anderson et al., 2017). The Chimera proposal transmits each digital signature by using two sub-frames 3 pages of the C/NAV message, with a repetition at most once every three minutes. This approach allows a receiver to verify the authenticity of the navigation message every three minutes. The NMA of Chimera is an asymmetric elliptic curve digital signature algorithm (ECDSA) P-224. It utilizes a 448-bit public key that has equivalent security as a 112-bit private key in a symmetric key system. The ECDSA scheme, a well-established Federal Information Processing Standard (FIPS) standard, is employed in the majority of open-source and commercially available cryptographic libraries. For Chimera protocol, the authentic receivers must have occasional access to Public Key Infrastructure (PKI), which is essential for any asymmetric cryptographic system, via no-GPS channels to get authenticated GPS public keys. In this system, if an entity wishes to provide an authenticated public key, it must obtain a signed certificate from a trusted Certification Authority (CA). Users can then verify the provided public key corresponds to the one in the signed certificate (Petovello, 2017).

The Galileo proposed Galileo Open Service Navigation Message Authentication (OSNMA). Different from Chimera, OSNMA is a data-only authentication technique based on a hybrid symmetric/asymmetric NMA known as the Time Efficient Stream Loss-tolerant Authentication (TESLA) (Petovello, 2017) (Anderson et al., 2017). TESLA is a hybrid symmetric/asymmetric NMA approach that authenticates the plaintext message based on the transmission of a Message Authentication Code (MAC), while computing the MAC based on the delayed transmission of the keys. The key is generated through a one-way function, F (i.e., a mathematical transformation that is very difficult to invert (Petovello, 2017)) and differs at each session. TESLA protocol implicitly maintains a key chain that starts with a random seed key K_n, which is a secret, and ends with a root key K_0 that is public and certified as authentic. The keys are used in reverse order to generate the MACs at each session. The one-way function, F can be a hash function (e.g., SHA-256 (Standard, 2012)) that computes each element of the key chain by hashing the previous element (Fernández-Hernández et al., 2016). Knowing the one-way function used for key generation, the authentic users can validate that the key received in the preceding session was generated by the one-way function using the input of the key received in the current session, which proves that the key received in the preceding session is indeed from the key chain of TESLA algorithm. Due to the nature of the one-way function, the users cannot predict "future" keys from the current key (Petovello, 2017). This proves beneficial because, even if the spoofer has the knowledge of the one-way function employed for key generation, the spoofer can never synthesize a valid key for a future session. Once the GNSS satellite establishes a TESLA chain, it will transmit messages, MACs, and keys using the delayed release mechanism above. At each time session, the receiver stores the message, MAC, and the key and waits until the next key is received. The receiver will then use the key it just receives to verify if the stored key is part of a TESLA chain, followed by authenticating the GNSS navigation message using the verified key and MAC (Lo and Enge, 2010). In addition to maintaining the cryptographic security of the secret private key, K_n used, TESLA authentication must be employed with another authentication system that can validate the authenticity of the public key (i.e., the root key K_0) for enhanced security (Fernández-Hernández et al., 2016). The current solution for the OSNMA is to initialize the GNSS receiver with a valid Galileo system public key in the factory, which can be less efficient and secure (Petovello, 2017). There is one more critical security condition that must be satisfied to ensure the integrity and robustness of the TESLA protocol: the receiver must have an authenticated time synchronization with an accuracy better than the key delay at the minimum. Otherwise, the receiver cannot tell if the received message was generated by a spoofer who had received a valid signing key from a live satellite signal (Petovello, 2017). The OSNMA is still in draft form and subject to change. It currently uses 40 bits every two seconds of the Galileo E1b I/NAV message. The navigation message is grouped into subframes of 30 seconds duration. The length of each MAC is only 10 to 32 bits, while the size of the key ranges from 80 to 256 bits. This allows OSNMA to sign many more messages than Chimera each second (Petovello, 2017). Our solution is implementing the TESLA algorithm with certification services from a trusted third-party to ensure a more secure and efficient process for validating the authenticity of the public key. The current solution for the OSNMA to ensure the authenticity of public keys is to initialize the GNSS receiver with a valid Galileo system public key in the factory, which can be less efficient and secure (Petovello, 2017). We will discuss the TESLA algorithm and our improvement in more detail in a later session.

There are some limitations associated with Chimera and OSNMA. Both approaches employ asymmetric encryption. Compared to symmetric encryption, asymmetric encryption is much more computationally intensive and requires a much longer key for the same level of security. On the receiver's end, OSNMA, or TESLA, has an even higher computation cost due to the MAC computation and the key validation through the evaluation of the one-way hash function. In contrast to ECDSA in Chimera, the TESLA scheme lacks a well-established standard, requiring additional efforts for integration and compatibility with GNSS receivers (Petovello, 2017). Moreover, TESLA is a data-only authentication method that cannot authenticate the time of transmission. Consequently, it relies on the assumption that receivers possess authenticated time synchronization with sufficient accuracy. Data-only authentication techniques are also susceptible to more sophisticated attacks (Anderson et al., 2017) (Petovello, 2017). On the other hand, compared to OSNMA, Chimera is less robust to replay attacks due to its TESLA key chain (Anderson et al., 2017). The receivers must have occasional access to a PKI via non-GPS channels to obtain the authentic public keys. This can be a problem in some scenarios. The same problem can arise for OSNMA when the receiver needs to have access to a trusted public key from PKI to authenticate the digitally signed root key (Petovello, 2017).

Our solution takes TESLA as the baseline, seeking to address the issues of the TESLA scheme. Chimera includes both spreading code and message authentication, which is very challenging for a term project. We also believe that TESLA has great potential to become a better authentication algorithm for future GPS communications.

III. PROBLEM STATEMENT

For precise location determination, GPS data is essential. It includes things like ephemeris data, ECEF coordinates, clock bias, and correction details to make sure GPS positioning is precise (as we learned from the lecture and Homework). While this data accurately determines our location, the data integrity may be jeopardized by techniques like signal spoofing and message alterations. Therefore, it is critical to secure GPS data from unauthorized alterations and message manipulation. We provide an extra layer of security to the data by using cryptographic techniques, which ensure that it cannot be altered or accessed by unauthorized parties.

After reviewing related work, we decided to utilize the TESLA algorithm as the NMA. To put it briefly, our project aims to comprehend the transmission of GPS data and enhance its integrity and security by implementing the TESLA technique in Cryptography, which increases the reliability of location-based services across a range of industries, such as Aviation, Urban Navigation, infrastructures, etc. We'll explore approaches within the TESLA algorithm, connecting theory to practical application in our pursuit of a more secure GPS framework. Last but not least, we will try to address one critical vulnerability in the TESLA algorithm. In TESLA algorithm, the following key chain is generated for key verification purposes:

$$[K_0, K_1, K_2, ..., K_N] \tag{1}$$

where the key, K_{i-1} is generated from its right neighbor, K_i using a one-way function. The receiver will first receive K_0, then K_1, K_3, and so on. The receiver can thus verify the validity of a key in a delayed manner after it receives the next key in the key chain. However, the current TESLA algorithm does not provide an efficient and secure approach to verifying the root key, K_0. The TESLA algorithm assumes that the root key, K_0 is reliable and authentic. Therefore, in this project, we will focus on addressing this issue as a potential improvement to the current TESLA algorithm.

IV. APPROACH

1. TESLA Algorithm

In our project, we employ Symmetric Message Authentication as a cryptographic technique primarily for ensuring "Integrity" in GPS communications, with a focus on preventing GPS signal spoofing and message alteration.

The sender utilizes a key and message through the function S to generate a message authentication code, which is then sent to the recipient for verification. The recipient, using the verify function, obtains either a "yes" or "no" output. In our approach, we employ HMAC (Microsoft, 2023) to generate the message authentication code. This process is illustrated in Figure 1.

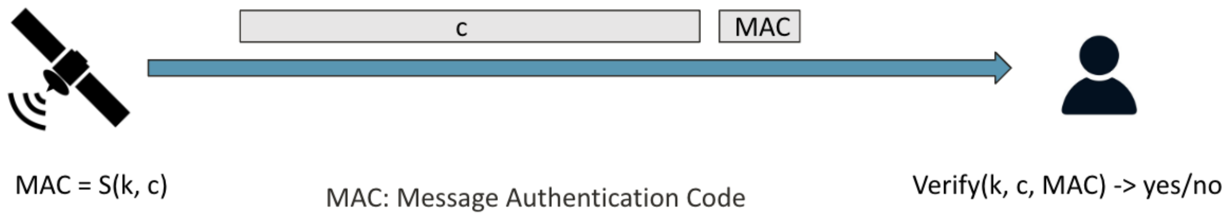


Figure 1: Symmetric Message Authentication

Algorithm 1 TESLA Sender

```
1: Generate key chain: $[K_0, K_1, K_2, ..., K_N]$
2: for $i \leftarrow 1$ to $N$ do
3:     $mac_i \leftarrow MAC(K, msg_i)$
4:     Send ($msg_i$, $K_{i-1}$, $mac_i$)
5:     if $i = 1$ then
6:         $sig \leftarrow Sign(sk, K_0)$
7:         Send ($sig$)
8:     end if
9: end for
```

Algorithm 2 TESLA Receiver

```
1: $msg_{prev}, mac_{prev}$
2: for $i \leftarrow 1$ to $N$ do
3:     $msg \leftarrow msg_i$
4:     $K_{prev} \leftarrow K_{i-1}$
5:     $mac \leftarrow mac_i$
6:     if $i = 1$ then
7:         verify ($pk$, $sig$, $K_0$)
8:     else
9:         verify ($K_{prev}$, $msg_{prev}$, $mac_{prev}$)
10:    end if
11:    $msg_{prev} \leftarrow msg$
12:    $mac_{prev} \leftarrow mac$
13: end for
```

Throughout this process, we address two major concerns. The first concern is key loss, which is a potential compromise of the security of the GPS communication. If the key is lost, the ability to verify the integrity of messages is jeopardized. An attacker may change the content of the messages, leading to potential misinformation, unauthorized access, or data corruption (Petovello, 2017). This undermines the confidentiality of the GPS communications. Secondly, the threat of a replay attack exists. A replay attack involves intercepting secure network communication to manipulate the recipient. The spoofers execute this type of attack by capturing and fraudulently resending or delaying the message to the wrong time (Günther, 2014). Since a replay attack doesn't require advanced decryption skills, it poses a significant risk to GPS communication. To address these problems, we draw inspiration from the TESLA Algorithm. In the event of key loss, we implement a key chain instead of a single key. Generating multiple keys in advance and using them sequentially in sessions ensures that even if one key is lost, the remaining keys can still be utilized to maintain the integrity of subsequent sessions. Additionally, the TESLA algorithm suggests establishing a different key for each session, which is a type of code that is only valid for one transaction. This prevents hackers from resending the message since the keys are different at each time session. In the TESLA algorithm, the key chain is generated in a one-way direction using one-way hashes (F). It is impossible to generate it in reverse. Thus, even if a hacker captures the message and key N from a session, they can only find the previous key in the previous session, which will be invalid at that time (Lo and Enge, 2010) (Fernández-Hernández et al., 2016).

This algorithm is set up by having the sender generate a secret key chain of length N. The key distribution is designed by delaying one session. From session 1 to N-1, the sender (only the sender can generate the MAC) will send the corresponding message, previous key, and the Message Authentication Code generated by the current key and current message to the recipient at each session:

$$MAC(K, msg) \tag{2}$$

For our project, we use a Keyed Hash Message Authentication Code (MAC) (Turner, 2008) where a key and a hash function are used to generate the message authentication code. HMAC (Microsoft, 2023) is a specific method that we used for implementing a Keyed Hash MAC. The hash function SHA-256 (Standard, 2012) is employed in the HMAC process and we call it H. It used in:

$$H(K||H(K||msg)) \tag{3}$$

where H is SHA-256, generating the output MAC of 256 bits:

$$H : \{0,1\}^n \rightarrow \{0,1\}^{256} \tag{4}$$

After receiving the message authentication code, the recipient then can derive the MAC generation key:

$$K' = F'(K) \tag{5}$$

and verify that the MAC was generated from the message and the derived MAC generation key. The recipient cannot verify the current key until the next session. The reason to do this is to prevent key leakage. Since multiple message is generally sent to the recipient at each session. If the sender sends the current key to the recipient while sending the first message. Once the hackers catch the key, then they can use the key to generate MAC to jeopardize other messages at the same session. In the last session N, the sender only sends Key N-1 to the recipient.

The key chain starts with Kn and ends with a root key K_0. It is crucial to know that K_0 is from a trusted source, ensuring the entire chain's verification. Because the provenance of K_0 can guarantee the whole chain to be verified. To verify K_0, one way we can use is public key authentication, where the sender has a secret key and the receiver has the corresponding public key. In session 1, the sender can use the secret key to sign the signature, then send the signature to the receiver so the receiver can use it to verify the K_0 (Lo and Enge, 2010). This process is illustrated in Figure 2.

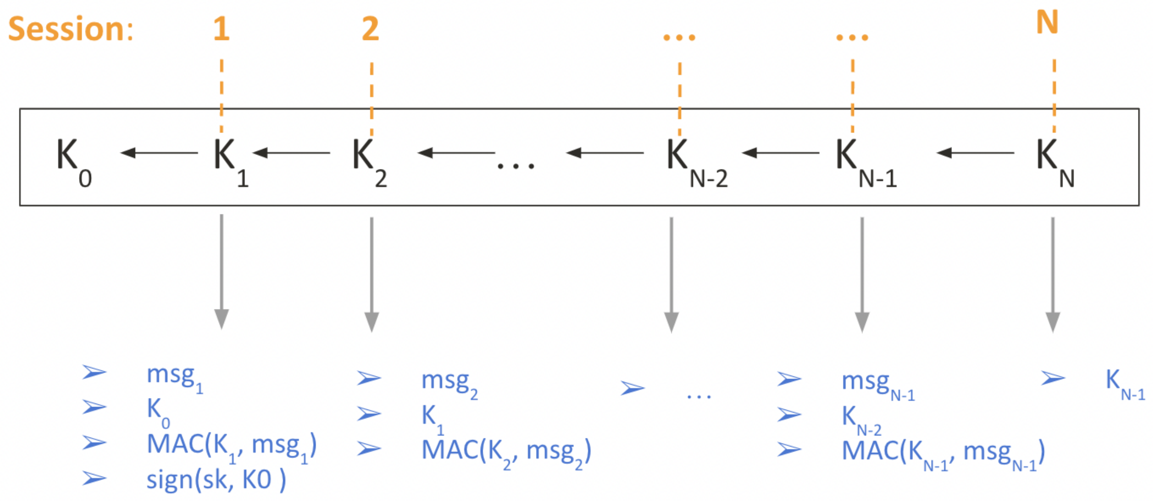


Figure 2: TESLA Algorithm Diagram

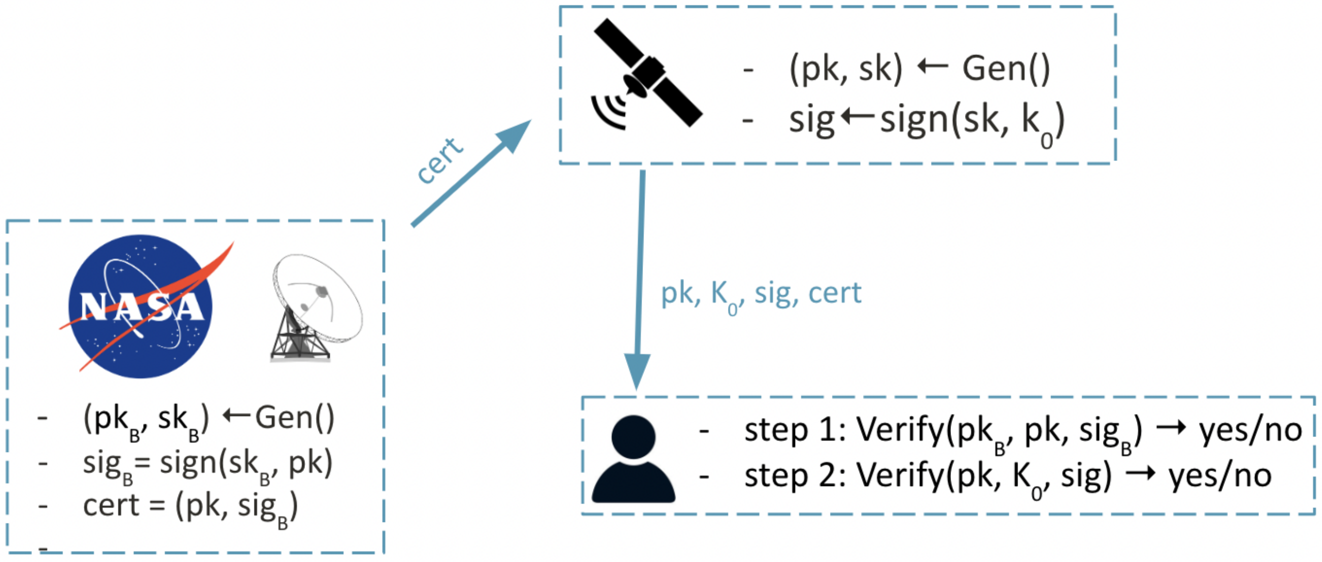


Figure 3: Secret Key and Public Key

In this Public Key Infrastructure scheme, the satellite signs K_0 with its secret key (sk) in session 1, generating a signature

$$sig \leftarrow sign(sk, K_0) \tag{6}$$

The receiver, upon receiving the signature sig in session 1, can then validate K_0 with the public key (pk) of the satellite.

$$Verify(pk, sig, K_0) \tag{7}$$

The public key is pre-installed on each receiver device.

2. Improvement Using Certificate Service

This validation method, while effective, brings challenges related to the management of public keys on receiver devices: it requires that the receiver devices have already installed the public key of each satellite in operation. This comes with two drawbacks:

- Since the public keys are stored on each receiver device, it is hard to maintain their integrity. Hackers can exploit software vulnerabilities to alter, tamper with, or even delete stored public keys, making the user unable to establish safe sessions with GPS satellites.
- All public keys stored require timely updates when new satellites are launched, or when existing satellites make changes to their keys.

To overcome these drawbacks, we propose the implementation of a certification service facilitated by a trusted third-party. For instance, in the case of Galileo, which is operated by the European Union Agency for the Space Program (EUSPA), EUSPA could serve as the designated trusted third-party for certification services (Communities, 2007). Similarly, for GPS, the United States Space Force, as the operator, could take on the role of the trusted third-party (Team, 2014). This approach decentralizes the responsibility of key management from individual receiver devices and ensures a more secure and efficient process for validating the authenticity of public keys.

In this scheme, the receiver will only store and maintain the public key of such trusted third-party (pk_B). The receipt and updates of pk_B can be done through secure cellular service, or even through a reliable internet connection, such as Hypertext Transfer Protocol Secure (HTTPS) (Naylor et al., 2014), etc. Since such authority usually manages GNSS service networks themselves, they have the most updated and accurate information on the public keys of each satellite in operation within its network. The authority will, with its own secret key sk_B, generate a certificate for the public key of each satellite:

$$cert \leftarrow (pk, sign(sk_B, pk)) \tag{8}$$

Then, this certificate is sent to the satellite. When the satellite tries to establish a session with the receiver, it sends its public key (pk), the key chain root (K_0), the signature on K_0, sig, and finally, the certificate generated by the trusted third-party, cert.

Upon receiving the info during session one, the receiver will first validate the public key of the satellite by running:

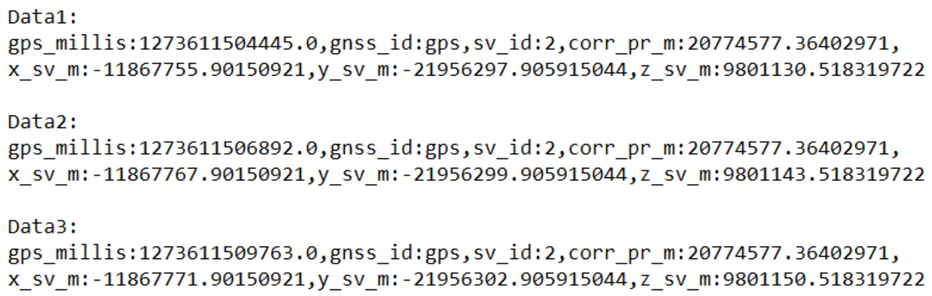
$$Verify(pk_B, cert) \tag{9}$$

to ensure that the public key (pk) of the satellite is authentic, before proceeding to the validation of K_0. This process is illustrated in Figure 3.

V. RESULTS

1. Target

We implemented the improved TESLA algorithm in Python with its native cryptography library (PyCA - Cryptography Authors, 2023). In our simulation, we utilized local network sockets to deliver example GPS data. In 3 sessions, the satellite will attempt to deliver these messages shown in Figure 4. Based on these GNSS navigational (Nav) messages from multiple different



```
Data1:
gps_millis:1273611504445.0,gnss_id:gps,sv_id:2,corr_pr_m:20774577.36402971,
x_sv_m:-11867755.90150921,y_sv_m:-21956297.905915044,z_sv_m:9801130.518319722

Data2:
gps_millis:1273611506892.0,gnss_id:gps,sv_id:2,corr_pr_m:20774577.36402971,
x_sv_m:-11867767.90150921,y_sv_m:-21956299.905915044,z_sv_m:9801143.518319722

Data3:
gps_millis:1273611509763.0,gnss_id:gps,sv_id:2,corr_pr_m:20774577.36402971,
x_sv_m:-11867771.90150921,y_sv_m:-21956302.905915044,z_sv_m:9801150.518319722
```

Figure 4: Example GPS messages

satellites, the receiver will be able to use the Newton-Raphson method to determine their precise location. Any tampering with the transmitted Nav data, either incorrect satellite coordinates or modified clock bias, can lead to significant errors in the final location determination. Therefore, our simulation aims at ensuring the integrity of such info using the improved TESLA algorithm. In particular, if any adversary attempts to spoof the communication with unauthorized Navigation data, the receiver should be able to detect it and reject such a message. Such a spoof should also not compromise the validation of any future legitimate Nav data delivered by the GPS satellite.

2. Implementation and Code

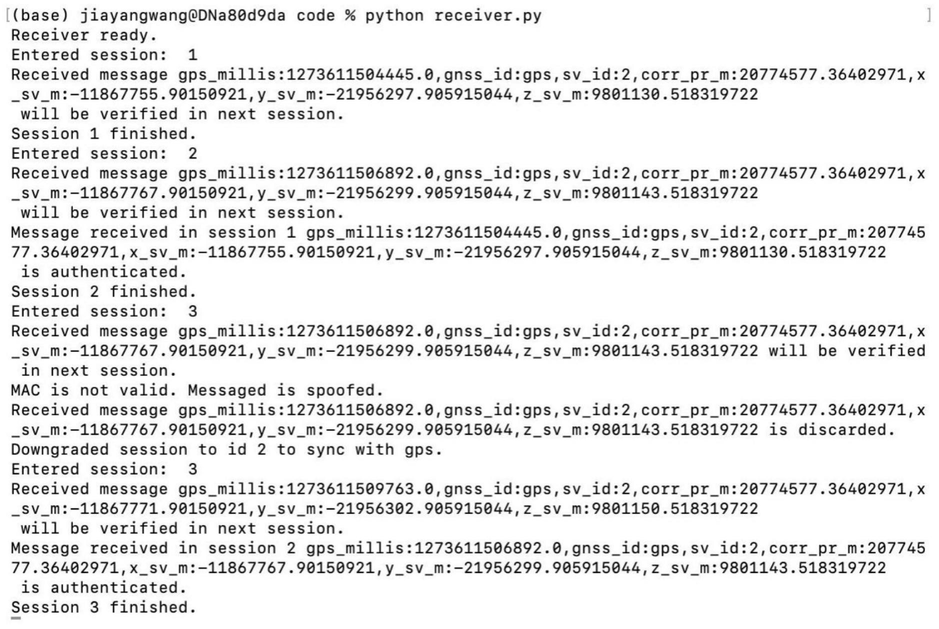
Our code is available at `https://github.com/jw4149/GPS_Project/tree/main`. The code base was organized into four files, each emulating the behavior of a key acting partner

- satellite.py - implements the satellite-side algorithm, which includes key chain generation, key delivery, Nav data delivery, Message Authentication Code (MAC) generation and delivery, etc.
- receiver.py - implements the receiver-side algorithm, which includes certificate validation, key validation, Nav data authentication based on MAC, etc.
- nasa.py - implements certificate generation for satellite's public key by the trusted third-party authority.
- spoofer.py - simulates an adversary that attempts to spoof the communication channel between the satellite and the receiver through a replay attack.

In our implementation, message delivery was performed on a local network socket. Upon starting, the receiver opens the portal and is ready to receive navigation data. The satellite then generates its public key pk and requests the authority to provide a certificate for it. The certificate, in turn, signs the satellite's pk and generates a certificate. The satellite then generates the TESLA key chain and starts the session with the receiver. As described above, the keys are delivered one session behind to provide security. The receiver, upon validating the satellite's public key pk and key chain root K_0, receives and saves Nav data from each session, then uses the received key to validate the message received in the previous session.

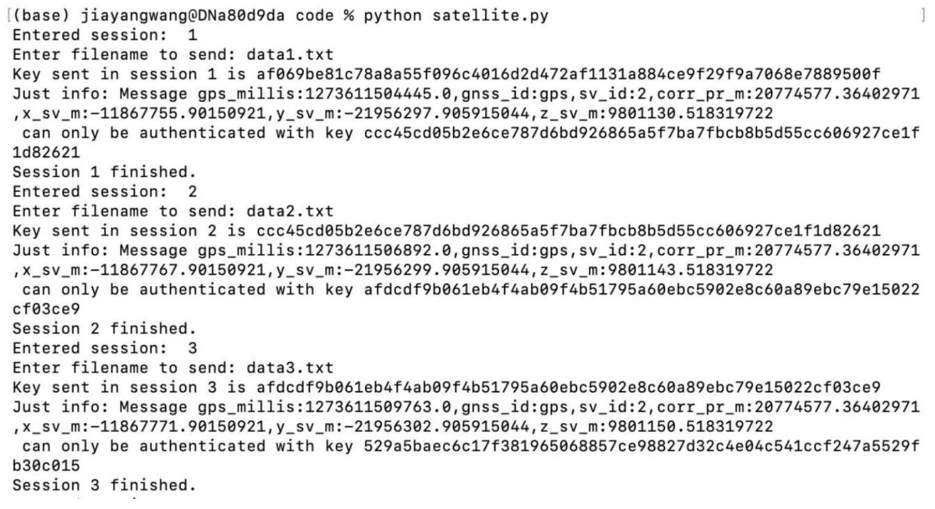
In our simulation, the satellite delivered data1.txt and data2.txt separately in Session 1 and Session 2, along with K_0 and K_1. The receiver saved the message and was able to validate data1.txt in Session 2. In Session 3, however, the simulated spoofer attempted a replay attack by re-sending data2.txt and K_1. The receiver would then attempt to validate data2.txt with the received K_1 from the spoofer. Since the Message Authentication Code was generated by satellite using K_2, this validation was not successful. Therefore, the receiver successfully detected the spoofing attack, and downgraded its session ID back to session 2, to synchronize with the satellite. Then, in the satellite's session 3, the satellite delivered data3.txt and K_2, and the receiver was able to use this information to validate data2.txt and continue the session.

Figure 5 and 6 demonstrate the experimental results at the Receiver's end and the Satellite's end. Our simulation showed a successful defense against a spoofing replay attack by an adversary, validating the improved TESLA algorithm we have proposed.



```
(base) jiayangwang@DNa80d9da code % python receiver.py
Receiver ready.
Entered session:  1
Received message gps_millis:1273611504445.0,gnss_id:gps,sv_id:2,corr_pr_m:20774577.36402971,x
_sv_m:-11867755.90150921,y_sv_m:-21956297.905915044,z_sv_m:9801130.518319722
 will be verified in next session.
Session 1 finished.
Entered session:  2
Received message gps_millis:1273611506892.0,gnss_id:gps,sv_id:2,corr_pr_m:20774577.36402971,x
_sv_m:-11867767.90150921,y_sv_m:-21956299.905915044,z_sv_m:9801143.518319722
 will be verified in next session.
Message received in session 1 gps_millis:1273611504445.0,gnss_id:gps,sv_id:2,corr_pr_m:207745
77.36402971,x_sv_m:-11867755.90150921,y_sv_m:-21956297.905915044,z_sv_m:9801130.518319722
 is authenticated.
Session 2 finished.
Entered session:  3
Received message gps_millis:1273611506892.0,gnss_id:gps,sv_id:2,corr_pr_m:20774577.36402971,x
_sv_m:-11867767.90150921,y_sv_m:-21956299.905915044,z_sv_m:9801143.518319722 will be verified
 in next session.
MAC is not valid. Messaged is spoofed.
Received message gps_millis:1273611506892.0,gnss_id:gps,sv_id:2,corr_pr_m:20774577.36402971,x
_sv_m:-11867767.90150921,y_sv_m:-21956299.905915044,z_sv_m:9801143.518319722 is discarded.
Downgraded session to id 2 to sync with gps.
Entered session:  3
Received message gps_millis:1273611509763.0,gnss_id:gps,sv_id:2,corr_pr_m:20774577.36402971,x
_sv_m:-11867771.90150921,y_sv_m:-21956302.905915044,z_sv_m:9801150.518319722
 will be verified in next session.
Message received in session 2 gps_millis:1273611506892.0,gnss_id:gps,sv_id:2,corr_pr_m:207745
77.36402971,x_sv_m:-11867767.90150921,y_sv_m:-21956299.905915044,z_sv_m:9801143.518319722
 is authenticated.
Session 3 finished.
```

Figure 5: Experimental Results at Receiver's End



```
(base) jiayangwang@DNa80d9da code % python satellite.py
Entered session:  1
Enter filename to send: data1.txt
Key sent in session 1 is af069be81c78a8a55f096c4016d2d472af1131a884ce9f29f9a7068e7889500f
Just info: Message gps_millis:1273611504445.0,gnss_id:gps,sv_id:2,corr_pr_m:20774577.36402971
,x_sv_m:-11867755.90150921,y_sv_m:-21956297.905915044,z_sv_m:9801130.518319722
 can only be authenticated with key ccc45cd05b2e6ce787d6bd926865a5f7ba7fbcb8b5d55cc606927ce1f
1d82621
Session 1 finished.
Entered session:  2
Enter filename to send: data2.txt
Key sent in session 2 is ccc45cd05b2e6ce787d6bd926865a5f7ba7fbcb8b5d55cc606927ce1f1d82621
Just info: Message gps_millis:1273611506892.0,gnss_id:gps,sv_id:2,corr_pr_m:20774577.36402971
,x_sv_m:-11867767.90150921,y_sv_m:-21956299.905915044,z_sv_m:9801143.518319722
 can only be authenticated with key afdcdf9b061eb4f4ab09f4b51795a60ebc5902e8c60a89ebc79e15022
cf03ce9
Session 2 finished.
Entered session:  3
Enter filename to send: data3.txt
Key sent in session 3 is afdcdf9b061eb4f4ab09f4b51795a60ebc5902e8c60a89ebc79e15022cf03ce9
Just info: Message gps_millis:1273611509763.0,gnss_id:gps,sv_id:2,corr_pr_m:20774577.36402971
,x_sv_m:-11867771.90150921,y_sv_m:-21956302.905915044,z_sv_m:9801150.518319722
 can only be authenticated with key 529a5baec6c17f381965068857ce98827d32c4e04c541ccf247a5529f
b30c015
Session 3 finished.
```

Figure 6: Experimental Results at Satellite's End

3. Overhead Computation and Performance Analysis

The security benefits of the proposed TESLA algorithm comes at the cost of data transmission time. On GPS, the L1C signal is transmitted on a carrier frequency of 1575.42 MHz, with a data transmission rate of 50 bps (Betz et al., 2007). According to our implementation, each navigation data delivery after a session has been established sends an additional overhead of at least 384 bits, which includes a key of 128 bits and a MAC of 256 bits. Due to this overhead, each single transmission will take additional time calculated below:

$$\frac{384\ bits}{50\ bits/second} = 7.68 seconds \tag{10}$$

This means that each message takes at least 7.68 additional seconds to be transmitted, let alone the time needed to run cryptographic algorithms to verify the validity of Nav data.

One way to reduce this overhead is to utilize a smaller key size, such as a 64-bit key, to generate the MAC. However, this comes at a cost of safety compromise. Because of the exponential nature of combinations resulting from the key size, cracking a 64-bit key by brute force is much easier than cracking a 128-bit key. It is therefore a trade-off between safety and additional transmission cost.

VI. CONTRIBUTIONS OF TEAM MEMBERS

The three group members contributed equally to the project. Jiayang was mainly responsible for code implementation. Yong Lin and Yiyang was mainly responsible for the presentation slides and the project report.

VII. CONCLUSION

In this paper, we proposed a novel Navigation Message Authentication (NMA) algorithm for GPS communications. The algorithm takes the TESLA algorithm as the baseline and introduces an additional certification service by the trusted third-party to ensure a more secure and efficient process for validating the authenticity of the public key. To address the security concerns in the transmission of civilian GNSS signals, both GPS and Galileo have proposed their solutions for NMA. GPS proposed the Chips-Message Robust Authentication (Chimera), while Galileo proposed Galileo Open Service Navigation Message Authentication (OSNMA), which implements the TESLA algorithm for key generation and verification. The current key distribution procedure for OSNMA, or TESLA, is to initialize the GNSS receiver with valid Galileo system public keys in the factory. This comes with two drawbacks. Firstly, it is hard to maintain their integrity because the public keys are stored on each receiver's device. Second, the public keys stored on each receiver's devices require timely updates when new satellites are launched, or when existing satellites change their keys. To solve issues with public key management on receiver devices, we implemented the TESA algorithm for key generation and verification, together with an additional certification service by a reliable third-party to validate the authenticity of the public key. Instead of storing the public key on the receiver's devices, the receivers will receive the public keys and their certificates from the reliable third-party. The certification service involves trusted entities like EUSPA for Galileo and the United States Space Force for GPS. Our implementation of the public key certification involves using a Keyed Hash Message Authentication Code with the SHA-256 hash function. Our simulation showed a successful defense against a spoofing replay attack from an adversary, validating the improved TESLA algorithm we have proposed. Due to the generation and verification of the keys and MACs, our proposed NMA algorithm introduces additional computation costs into GPS signal transmission, leading to an additional overhead of at least 384 bits and a longer transmission time of approximately 7.68 seconds. One way to reduce this overhead is to utilize a smaller key size, such as a 64-bit key, to generate MAC, but this comes at a cost of safety compromise. There's always a trade-off between safety and transmission costs. We believe that our proposed algorithm can enhance the integrity and security of the current TESLA algorithm, making it a better solution for NMA for future GPS communications.

VIII. FUTURE DIRECTIONS

For future directions, we will explore an improved algorithm that handles complete or partial loss of data due to signal interference during transmission. To reduce the bit overhead, we will explore the possibility of using a PRN code-level alteration scheme to authenticate communications.

ACKNOWLEDGEMENTS

We would like to thank Prof. Gao and Daniel for their great help with our project and throughout the course.

REFERENCES

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. In *Proceedings of the 30th International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS+ 2017)*, pages 2388–2416.

Betz, J. W., Blanco, M. A., Cahn, C. R., Dafesh, P. A., Hegarty, C. J., Hudnut, K. W., Kasemsri, V., Keegan, R., Kovach, K., Lenahan, L. S., et al. (2007). Enhancing the future of civil gps: overview of the l1c signal. *Inside GNSS*, 2(3):42–49.

Communities, E. (2007). Co-operation agreement on a civil global navigation satellite system (gnss) between the european community and its member states and ukraine. https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/243190/7199.pdf.

Directorate, G. P. S. (2013). Navstar gps space segment/user segment l1c interface, revision d. Technical Report Vols. IS-GPS-800D, U.S. Department of Defense.

Divis, D. A. (2013). Gps spoofing experiment knocks ship off course. *Inside GNSS*, 31:1–3.

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: Journal of the Institute of Navigation*, 63(1):85–102.

Günther, C. (2014). A survey of spoofing and counter-measures. *NAVIGATION: Journal of the Institute of Navigation*, 61(3):159–177.

Kerns, A. J., Shepard, D. P., Bhatti, J. A., and Humphreys, T. E. (2014). Unmanned aircraft capture and control via gps spoofing. *Journal of field robotics*, 31(4):617–636.

Lo, S. C. and Enge, P. K. (2010). Authenticating aviation augmentation system broadcasts. In *IEEE/ION position, location and navigation symposium*, pages 708–717. IEEE.

Microsoft (2023). Hmacsha256 class. `https://learn.microsoft.com/en-us/dotnet/api/system.security.cryptography.hmacsha256?view=net-8.0`. Accessed: 12/9/2023.

Naylor, D., Finamore, A., Leontiadis, I., Grunenberger, Y., Mellia, M., Munafò, M., Papagiannaki, K., and Steenkiste, P. (2014). The cost of the" s" in https. In *Proceedings of the 10th ACM International on Conference on emerging Networking Experiments and Technologies*, pages 133–140.

Petovello, M. (2017). What is navigation message authentication. https://insidegnss.com/wp-content/uploads/2018/04/janfeb18-SOLUTIONS.pdf.

PyCA - Cryptography Authors (2023). Welcome to pyca/cryptography. `https://cryptography.io/`. Accessed: 12/9/2023.

Standard, D. E. (2012). National institute of standards and technology, fips pub 46-2 (december 1993). *http://www. itl. nist. gov/fipspubs/fip46-2. htm.*

Team, G. P. (2014). Global positioning system (gps) standard positioning service (sps) performance analysis report. *GPS Product Team: Washington, DC, USA*.

Turner, J. M. (2008). The keyed-hash message authentication code (hmac). *Federal Information Processing Standards Publication*, 198(1):1–13.