Design and Analysis of a Public Key Infrastructure for SBAS Data Authentication

Andrew Neish, Todd Walter, J. David Powell
Stanford University

Abstract

Integral to any authentication system is the design of its public key infrastructure (PKI), allowing the system to divide and allocate the responsibility of security between different entities. Moreover, any PKI is closely tied to its associated key management (KM) architecture that is responsible for the distribution, maintenance, and revocation of cryptographic keys. This paper develops two candidate authentication schemes: one each for an L5 I-channel and L5 Q-channel implementation. A PKI is then proposed along with its associated KM architecture. A simulator, introduced as MCOS, is developed to enable large scale Monte Carlo tests of these different PKI and KM designs. The schemes are tested using MCOS producing results concerning key performance metrics of these authentication schemes and their associated PKI. This work concludes that a strictly over-the-air method of delivering PKI information is feasible for SBAS data authentication systems using the proposed PKI and KM architecture.

Introduction

Global Navigation Satellite System (GNSS) technologies have become pervasive in today's world. The number of receivers in the civilian sector is in the billions and continues to grow at an explosive rate. While GNSS for many individuals is used as a tool for navigation, it also represents the most accurate and affordable way to perform precise timing for operations including financial transactions and disciplining the power grid. According to the department of Homeland Security, 15 of the 18 critical infrastructure and key resources sectors in the United Sates rely upon GNSS to perform their functions. As the adoption of GNSS technologies surges, knowledge of its vulnerabilities has become more common place and tools of disruption and deception of the service have become cheaper and more available. GNSS signals have very low power by the time they reach earth, making them vulnerable to intentional and unintentional jamming from relatively low power devices. The structure of the civilian signals is open and freely accessible giving anyone with the knowledge of that structure to generate navigation signals of their own. Open sources tools are available on the web that allow users to broadcast false GNSS signals making an attack on infrastructure and safety of life systems more accessible than ever. Suspected spoofing cases have been reported in the Black Sea and near the Kremlin and searches on Google for GPS spoofing techniques have risen sharply in the last several years. The GNSS community is developing methods to detect and mitigate spoofing threats and one of the methods currently under consideration is the use of cryptographic signatures to authenticate the GNSS data. Spoofing threats to Satellite Based Augmentation Systems (SBAS) constitute a single point of failure for safety critical systems that rely upon correction and integrity services provided through the SBAS data stream. Data authentication, in addition to receiver-based detection methods, can serve as a viable method to mitigate intentional spoofing threats carried out through false signal generation

and is currently being considered for SBAS systems such as the Wide Area Augmentation System (WAAS) and the European Geostationary Navigation Overlay Service (EGNOS).

There are many cryptographically secure methods of authenticating data available, but any method that is used on SBAS must meet several criteria. The bandwidth and data rate of SBAS systems is limited in contrast with common internet infrastructure and so any authentication service implemented on SBAS must be small in message size. Adoption and implementation of new services and technologies is slow in sectors such as aviation, which will require the chosen method to be cryptographically relevant for at least 30 years. The SBAS information stream is also one-way, from the satellite to the user, and so any method that is implemented must also be one-way. This work considers two candidate schemes for SBAS authentication: The TESLA (Timed Efficient Stream Loss-tolerant Authentication) protocol and ECDSA (Elliptic Curve Digital Signature Algorithm).

Authentication of broadcast messages relies upon the notion that there is a private key used by the sender to create signatures for their messages and a public key used by receivers that can verify the authenticity of the messages using the received signature. These private/public key pairs are susceptible to numerous attacks depending on the authentication scheme implemented and so it is wise to consider the practice of replacing these keys at a reasonable rate. It is also good practice to have a method in place to revoke keys that may have been compromised. This is known as the practice of key management (KM) of a public key infrastructure (PKI) and a well-designed PKI should function without causing an interruption to the SBAS service. Several publications have called for the design of a KM architecture tailored to the needs of GNSS authentication [1]–[9] and one publication has offered a potential solution [10]. This solution was designed for Galileo's open service navigation message authentication (NMA) service and utilizes other networks/resources to carry out KM. The KM architecture presented in this work is designed for use in SBAS systems and is facilitated solely through the SBAS broadcast signal and does not rely upon any other networks or in-person maintenance. This is possible through the use of an Over-The-Air-Rekeying (OTAR) service that utilizes leftover bits in a signature message unused by the signature itself. Although the analysis of this paper will use WAAS as an example SBAS to test these particular KM architectures, the resulting designs can be used for other SBAS systems with some modification.

The paper begins with a short overview of the characteristics of ECDSA and TESLA and how they are implemented, after which a candidate PKI and method of key management is presented for both authentication schemes. A tool known as the Monte Carlo OTAR Simulator (MCOS) is developed in order to simulate and evaluate the performance of these key management designs. Two example PKI configurations are analyzed for use on the WAAS L5 in-phase (L5I) and quadrature-phase (L5Q) channels. These analyses aim to show the effects that OTAR broadcast methodology and demodulation accuracy (represented as C/N_0 or page error rate (PER)) have on an SBAS receiver's ability to reliably use the authentication/OTAR service. Preliminary PKI and KM architectures with associated operational performance values are put forth in conclusion to this work.

Authentication Implementation

A detailed description of both ECDSA and TESLA authentication protocols can be found in the literature [11], [12]. For the sake of brevity, a brief overview of their traits pertinent to the discussion of SBAS authentication and KM are outlined here. Their respective implementations are also presented.

L5Q - ECDSA

ECDSA is a standardized, common asymmetric scheme used in public key encryption and digital authentication. The security of the algorithm relies upon the toughness of the discrete logarithm problem through the vehicle of elliptic curves. There are several curves that have been standardized by the National Institute of Standards and

Technology (NIST) [11] with their public key, private key, and signature sizes shown in Table 1. EC-Schnorr is another elliptic curve authentication scheme that has smaller signature sizes than ECDSA and has been discussed as a possible candidate for authentication [9], but as it has not yet been standardized, this work will consider ECDSA.

Table 1: ECDSA Curves and associated parameters

NIST Curve	Security (bits)	Public Key Length (bits)	Signature Length (bits)
P-192	96	192	384
P-224	112	224	448
P-256	128	256	512
P-384	192	384	768
P-521	256	521	1042

Compared with TESLA, ECDSA has a longer signature and public key length for an equivalent security level. Unlike TESLA, however, ECDSA is a completely asymmetric scheme that does not require any loose time synchronization between the sender and receiver. As ECDSA has a longer signature length, it is being considered for implementation on the L5Q channel as its signature will take more than a single SBAS message frame. Implementing a service on the L5Q channel would require that the service either split the current power between the I and Q channels or procure more power from the satellite provider.

NIST has released recommendations for security strength and future usage of standardized algorithms ([13] Table 4). These recommendations are for federal government unclassified applications for sensitive data but serve as useful guides for data of similar sensitivity. The table is reproduced and presented here in Table 2.

Table 2: Security-strength time frames presented in [13]

<table>
<tr><th colspan="2">Security Strength</th><th>Through 2030</th><th>2031 and Beyond</th></tr>
<tr><td rowspan="2"><112</td><td>Applying</td><td colspan="2">Disallowed</td></tr>
<tr><td>Processing</td><td colspan="2">Legacy-use</td></tr>
<tr><td rowspan="2">112</td><td>Applying</td><td rowspan="2">Acceptable</td><td>Disallowed</td></tr>
<tr><td>Processing</td><td>Legacy-use</td></tr>
<tr><td>128</td><td rowspan="3">Applying/Processing</td><td>Acceptable</td><td>Acceptable</td></tr>
<tr><td>192</td><td>Acceptable</td><td>Acceptable</td></tr>
<tr><td>256</td><td>Acceptable</td><td>Acceptable</td></tr>
</table>

These suggested security strengths are implicitly tied to their intended uses and respective cryptoperiods. As defined by NIST [13], “a cryptoperiod is the time span during which a specific key is authorized for use by legitimate entities, or the keys for a given system will remain in effect.” Any well-designed authentication scheme will require the private/public key pairs to be replaced periodically to limit the time available for computationally intensive cryptanalytic attacks. NIST does not provide guidance on the exact length of cryptoperiods that should be used [14], but do provide general guidelines for the specific use case of unclassified federal documents mentioned above. For a public signature-verification key, a cryptoperiod of several years depending on the key size is suggested. Because SBAS is a broadcast only system, the number of known cryptanalytic attacks on ECDSA is less than it would be otherwise, and so, barring the physical security needed to protect the private keys owned by the SBAS provider along with ensuring that the scheme is implemented properly, much of the risk of the scheme is

shouldered by the protocol itself. It is proposed that ECDSA with NIST curve P-224 be used to authenticate L5I SBAS messages if L5Q is allocated for data authentication. It is recognized that P-224 is a smaller curve and will be phased out in future encryption and authentication tasks. Because the validity of the information is short-lived, this curve will be acceptable as long as the cryptoperiod of the key is sufficiently short. As will be shown later after the PKI and KM architecture are introduced, OTAR will allow the system to update the keys at a relatively high rate to ensure the security of the scheme and give flexibility to the selection of these cryptoperiods. An example sequence of this scheme is provided in Figure 1. Figure A - 1 in Appendix A shows the message structure of L5Q.

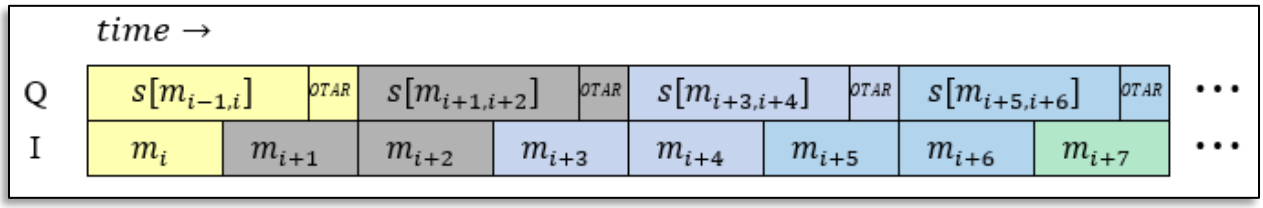

Figure 1: L5 Authentication with ECDSA

The notation m_i corresponds to an SBAS message, $s[\cdot]$ corresponds the signature of the messages in the brackets, and $OTAR$ represents the extra bits leftover from the signature message available for KM. A single signature is 448 bits, which is longer than a single message. Because of this, messages are grouped together in pairs and then their concatenated forms are signed. The signatures can be created once the second message of each pair is known, and so the signature can be transmitted starting at the beginning of the second message. This minimizes the impact to availability that data authentication poses. If a message is lost in the I-channel, its partner will not be able to be authenticated. If there is a disruption of the service where the receiver is not able to demodulate the data on either the I or Q channel in a single message, then the receiver will lose one message and not be able to authenticate either 2 or 4 messages, depending on the time at which the disruption occurred. With this implementation there would be a 1/3 probability that 2 messages would be lost, and 2/3 probability that 4 messages would be lost. Minimizing the signature length in the Q channel minimizes the impact to availability that a temporary disruption of the service would have. The impact to availability will grow for I channel implementations as in those cases the limited bandwidth forces the SBAS authentication service to group larger amounts of information together for each signature, necessarily meaning the loss of one message compromises the authentication of all messages within the same group. It is still an open question of how a receiver should treat a message that failed to authenticate.

For WAAS, each message is 250 bits length, and the L5I messages will have a 216-bit data field. The extra 34 bits in the message are left for a 4-bit preamble, a 6-bit message identifier, and a 24-bit CRC. There are several changes to this structure that can be made when implementing an authentication message in the Q channel. Since both the I and Q data are simultaneously demodulated in the receiver's PLL, the preamble for the I channel, used for frame synchronization, can also serve to synchronize the frames of the Q channel, thus allowing the Q channel to operate without a preamble. A message identifier is also not needed for the signature, as it is understood that all bandwidth of the Q channel is allocated for authentication and OTAR. Finally, it is proposed that a Q channel message need not contain a CRC. A CRC serves to give the user confidence that the incoming message has been demodulated correctly. The correctness of the demodulation of the Q channel can be verified by checking whether the signature algorithm authenticates its pair of messages. In the event that a signature does not authenticate the messages, the opposite cannot be determined, however. If the I channel messages were un-perturbed, it would not be clear if the messages failed to authenticate due to a demodulation error or through a spoofing incident.

With this in consideration, all 500 bits in the Q channel authentication message can be split to deliver 448 bits for the signature and 52 bits for OTAR.

The security of this implementation of ECDSA is feasible with current and near-future classical computer capabilities. With the advent of quantum computing, ECDSA and other algorithms relying upon the discrete logarithm problem for security are at risk [15]. NIST is taking this threat seriously and is currently in the process of searching for asymmetric algorithms and protocols that are quantum resistant [16]. There are many different predictions for when quantum computing will be available at a scale capable of breaking today's cryptographic algorithms. Several major technical milestones must be met before this technology is realized [17]. As there are currently no standardized post-quantum asymmetric algorithms, ECDSA will serve as the candidate algorithm for initial implementation of an L5Q SBAS authentication service.

L5I - TESLA

If a signature scheme implemented on L5I is desired, the implementation of ECDSA may have an impact to bandwidth of the nominal service that is too heavy to bear. Therefore, for L5I, the TESLA protocol is selected as the SBAS authentication candidate. TESLA requires loose time synchronization between the receiver and sender and requires a more elaborate PKI but has the advantage of being able to transmit a MAC (Message Authentication Code) and key in a single SBAS message data field. It is also less computationally expensive to implement when compared with ECDSA.

TESLA consists of a symmetric algorithm that is made asymmetric by delaying the release of keys. A MAC is generated by the sender for a given set of messages and a system time by using a secret key and the MAC is transmitted after those messages have been received. At a later time, the same secret key that was used to generate the MAC is sent to the user so that they may verify that same MAC and authenticate the data that it signed. Before a user is willing to verify the MAC though, they must verify that the key was generated by the sender. Before the first MAC is sent, a keychain is created by the sender using a series of recursive, publicly known one-way functions, in this case SHA-256 combined with a few other operations. When signing the MACs, the sender signs each with the keys in the keychain in the order opposite which they were generated. The first key that is sent, also known as the root key, is authenticated using an asymmetric algorithm whose signature is delivered using the part of the message designated for OTAR. ECDSA is the asymmetric algorithm used to authenticate the root key in this case. When a receiver first turns on and it receives a key somewhere in the keychain, the receiver applies these one-way functions recursively to the key until they arrive at the root key. The receiver then authenticates the root key using ECDSA and thus authenticates the keychain. Every key thereafter can be authenticated by applying the one-way function once to verify that the current key received produces the key that was previously released.

The implementation of TESLA for the L5I channel in this work contains the following features. In order to prevent pre-computation attacks on the TESLA keychain, a salt is incorporated into the one-way function that is unique to each keychain. A single iteration of this one-way function is shown in Figure 2. In this case, || indicates concatenation. In [18], a salt length of 30 bits was proposed and is incorporated in this work.

$$k_i \rightarrow k_i||\text{salt} \rightarrow \text{SHA256}(k_i||\text{salt}) \rightarrow \text{trunc}(\text{SHA256}(k_i||\text{salt})) = k_{i-1}$$

Figure 2: One-way function used in TESLA keychains

A nominal time between authentication (TBA) is chosen to be 6 seconds and Figure 3 depicts the inclusion of the authentication scheme in the L5I channel.

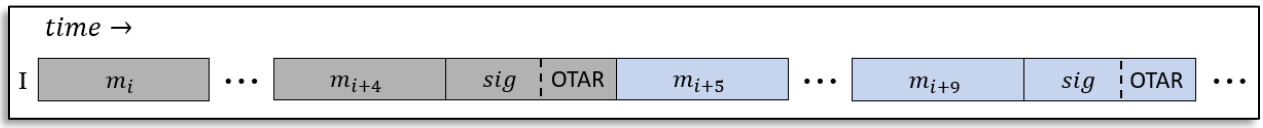

Figure 3: L5 Authentication with TESLA

As mentioned before, a drawback to the use of TESLA is the need for loose time synchronization between the sender and receiver. In the case of SBAS enabled receivers, there are no stable trusted external sources of time readily available without the use of alternative networks. If the TBA is not sufficiently large, loose time synchronization cannot be ensured and so the receiver would be vulnerable to time delay attacks. The larger the TBA is, however, the more messages there are that are signed by a single MAC, thus having a detrimental effect to the overall availability of the system.

In order to alleviate some of these drawbacks, a double instance keychain system is proposed. The basic premise of this design is such that there are two instances of the same keychain. Each instance operates with a different delay in their release. This allows the system to relax the loose time synchronization requirement to be on the order of minutes instead of seconds, which is much more feasible for internal receiver clocks. More information on the use of this method, known as Concurrent TESLA Instances, can be found in [19].

Public Key Infrastructure

In this section, a candidate PKI is introduced for both the L5I and L5Q authentication schemes. The design of a good PKI will try to take the following considerations into account. A PKI allows an authentication service to strike a balance between performance (smaller key/signature sizes), and security (larger key/signature sizes). It does so by creating a chain of trust whereby the smaller signatures have shorter cryptoperiods and are used for the authentication of SBAS data, while the larger signatures/keys are used to authenticate the smaller keys. From this point on, the designs will be specific to WAAS, but the concepts can be adapted to other SBAS systems. This section will introduce the PKI for both the L5I and L5Q authentication candidates and will be followed by a section on key management.

L5Q - ECDSA

The PKI of ECDSA consists of two levels of keys. Level 2 private keys are used by the WAAS Master Station (WMS) to authenticate WAAS messages and WAAS enabled receivers use level 2 public keys to check the authenticity of the incoming messages. These level 2 messages are delivered over the air using available OTAR bits. When these level 2 public keys are delivered to the receiver, they are signed with a level 1 signature also sent using OTAR bits. The hashed value of the level 1 public key is physically installed in the receiver by the receiver manufacturer. If there are a total of 1024 level 1 key hashes stored in the receiver, the required firmware capacity is roughly 150KB. Figure 4 shows the relative cryptoperiod of both levels of keys and Table 3 gives the parameters of the PKI. In the event of a level 2 key compromise, a new level 2 key can be introduced, and the compromised level 2 key can be revoked using the security of the level 1 keys. If a level 1 key is compromised, the system can introduce a private key to unlock a different level 1 key and force an early retirement of the compromised key. How this is done and how these keys are released and treated is discussed in more detail in the KM section of this paper.

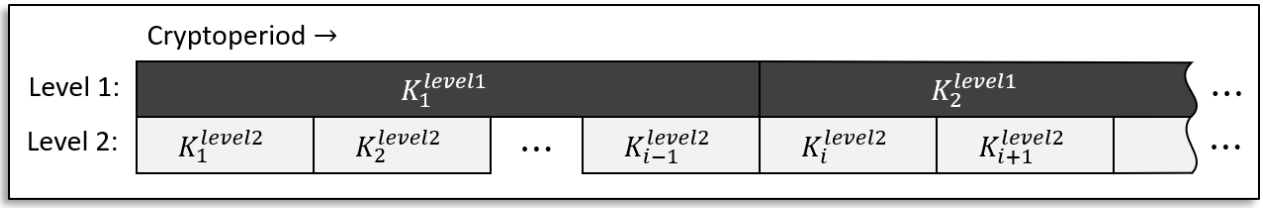

Figure 4: Relative Cryptoperiods between L5Q keys

Table 3: Summary of Parameters for L5Q PKI

Summary of Parameters for L5Q PKI	
Bits available for OTAR in each signature message	52 bits
ECDSA level 2 Signature Length	448 bits
ECDSA level 2 Public Key Length	224 bits
ECDSA level 1 Signature Length	768 bits
ECDSA level 1 Public Key Length	384 bits
# of locally stored ECDSA level 1 public key hashes	1024
Storage space required for ECDSA level 1 keys	~ 150 KB

L5I - TESLA

The PKI of TESLA is similar in structure to the L5Q implementation discussed above but contains added layers to support the TESLA keychains. There exists a level 1 and level 2 key structure where the level 2 keys are used to authenticate the root keys of the TESLA keychain. TESLA root keys are sent over the air via OTAR bits and are authenticated using level 2 keys with level 2 signatures also sent with OTAR bits. Figure 5 shows the relative cryptoperiod of the different keys and Table 4 offers a summary of their parameters.

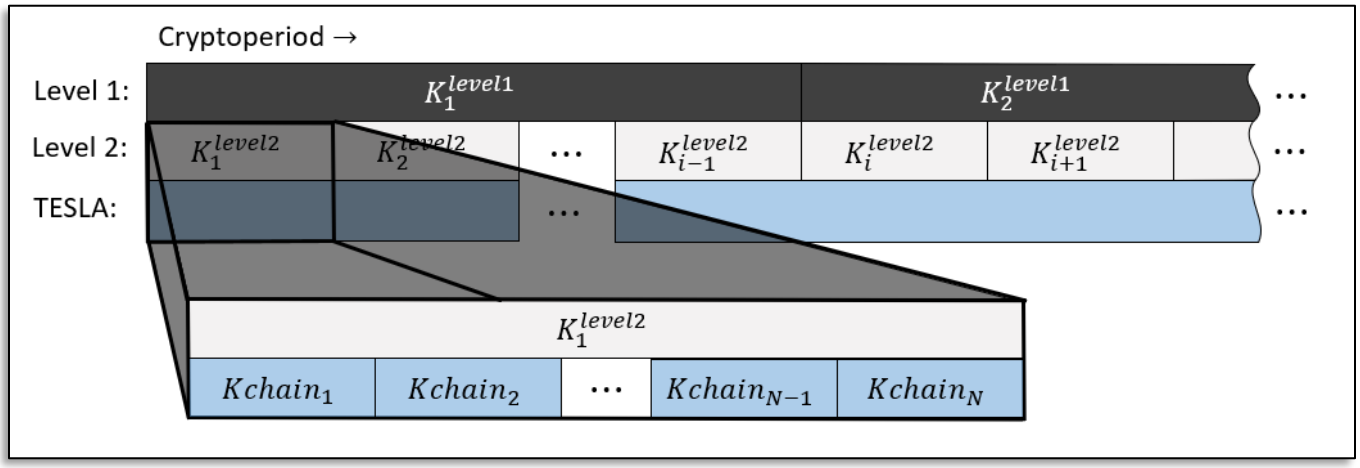

Figure 5: Relative Cryptoperiods between L5I keys

Table 4: Summary of Parameters for L5I PKI

Summary of Parameters for L5I PKI	
Bits available for OTAR in each signature message	68 bits
TESLA Key Length	115 bits
TESLA MAC Length	30 bits
TESLA Salt Length	30 bits
ECDSA level 2 Signature Length	448 bits
ECDSA level 2 Public Key Length	224 bits
ECDSA level 1 Signature Length	768 bits
ECDSA level 1 Public Key Length	384 bits
# of locally stored ECDSA level 1 public key hashes	1024
Storage space required for ECDSA level 1 keys	~ 150 KB

Key Management

Now that the PKI has been introduced, the method by which the public key information is managed is described. The level 1 keys are the root of trust in both PKI. Therefore, their security against physical attacks must be the greatest. One option is to create the level 1 keys and share them between the 3 WMS in a secure way. This way, when a new level 2 key is introduced, the WMS can create the signature for the level 2 key and send it along with its signature using OTAR bits. There are several challenges that face this implementation. This would require the process of creating and distributing the level 1 keys to be done in a secure manner, which may involve physically transporting hardware across the US. Once the keys have been securely distributed, the WMS must be capable of storing them in a secure way on site while still being able to access the keys to create level 1 signatures. The process of developing the infrastructure to sufficiently secure this information would likely be costly to the WAAS program. The WMS must also agree which level 1 key is to be used and when the change to the next level 1 key will occur. These level 1 keys would only be entrusted to WAAS and so any receiver that wishes to use an authentication service offered by another SBAS service provider using a similar PKI would require the receiver to be equipped with multiple sets of hashed level 1 keys.

Another option is to contract the responsibility of securing and using level 1 private keys to a 3rd-party certificate authority (CA) (E.g. Comodo, Symantec, etc.) that is already equipped to protect information with this level of sensitivity. When a new level 2 key needs to be introduced, the CA is sent the new level 2 public key by the WMS and is asked to create a signature for that key with the current level 1 key. Once the signature is delivered it can be sent to receivers using OTAR bits and designated as the next level 2 public key to be used. Using a CA allows WAAS to implement a secure PKI while minimizing the cost of its secure infrastructure. In addition to this, if a CA were agreed upon between different SBAS providers, only one set of hashed level 1 keys would need to be installed on receivers and would simplify the implementation of KM across different GNSS platforms.

For reasons of security, every level 2 and level 1 key will have an expiration date associated with it. The expiration will be broadcast using OTAR bits and will be signed with a level 1 signature. In order to facilitate a smooth transition between level 2 public keys, the next public keys to be used will be transmitted as well as the current keys. In the case of level 1 keys, the current and next level 1 keys are transmitted using OTAR bits and both are authenticated with a level 1 signature signed by the current level 1 key.

It is worth noting that there is a potential security threat with sending the current level 1 public key. Since it is essentially self-authenticated, the only thing that prevents an attacker from sending a false level 1 key is the stored hash values of the planned level 1 keys. In order to forge a level 1 key, an attacker would need to find the preimage of one of the hashed values (a level 1 public key) that has not been used yet, and then find the secret key to match the discovered public key. This attack is unlikely to succeed through both layers of security. If an attacker were able to break an old level 1 key pair that is no longer in use and discover an old private key, they could send an old level 1 public key and send forged messages to a receiver. In order to prevent this, the receiver is required to keep a revocation list of past level 1 key hashed values and as time goes on, expired keys will not be installed in the receiver firmware. Once a level 1 key has been superseded by another level 1 key, that level 1 key will never be used again and will be considered obsolete. Any attacker wishing to use an old level 1 key would need to ensure that the victim receiver has no knowledge of the expiration of the older level 1 key, which would be improbable given that level 1 keys only change once every several years.

All level 2 and level 1 keys will have associated expirations, but WAAS will have the option to retire them sooner if required. Level 2 and level 1 keys are retired the moment the next keys are used. Once these keys have been retired, they are considered obsolete and will never be used again. If a key that is designated as the next level 2 or level 1 key needs to be revoked, the system will broadcast a new key that is designated as the next key and the receiver will consider the previously stored next key revoked.

For the sake of the PKIs and KM architectures presented here, all level 1 and level 2 standard practices are the same between the L5I and L5Q implementations. In the following sections, the actual structure of how the OTAR information is transmitted is discussed. For the L5I and L5Q implementations, there are several OTAR Message Types (OMT) that organize how the OTAR bits are utilized.

L5Q – ECDSA

In the case of L5Q, where ECDSA is used, there are 52 bits reserved for OTAR every 2 seconds, referencing Table 3. OMTs do not require a CRC or preamble. The basic structure of an OMT for L5Q ECDSA is shown in Figure 6. For clarification, Figure 6 depicts the contents of the “OTAR” boxes seen in Figure 1. The first 4 bits are the OMT header which announce the OMT that is contained in the OTAR data field. Depending on the OMT, the OTAR message may need to be split among several messages. In this case, a sequence number is used to identify which part of the specific OMT is being transmitted. The number of bits that need to be reserved for this sequence number will depend on the length of the OMT and will change for each OMT. The rest of the bits are then used to transmit the data of the associated OMT. Appendix A shares detailed information on each individual OMT. Referencing Table A - 1, OMT1 is the current level 2 public key and OMT2 is the level 1 signature of OMT 1. For any receiver that has been off for a short amount of time, this information is crucial as it lets the receiver know which level 2 public key to use in order to authenticate the incoming WAAS messages. Therefore, OMT1 and OMT2 will be broadcast more frequently than any other OMT. A description of the broadcast schedule of these OMTs is presented later in the analysis portion of this work. When a receiver starts from a cold start, it will go through the steps outlined in Figure 8 to begin authenticating data.

L5I – TESLA

Level 2 keys are used to authenticate the root keys the TESLA keychain. As the keychain goes on in time, a receiver just starting up may have to perform many recursive one-way functions to arrive at the root key. In order to alleviate some of the computational time required to authenticate the keychain, intermediate keys will be authenticated by level 2 keys and the intermediate keys will replace the root key at appropriate intervals. These intervals will be determined and publicly known so that a receiver knows not to apply the one-way function more

times than would be specified by this interval. For I-channel implementations, a bit field is also required in the signature that specifies how many of the previous messages are being authenticated. This is because it cannot be guaranteed that authentication message types will be delivered exactly on the interval the nominal TBA specifies. For this implementation, where TBA is nominally 6s, this field specifying which messages are authenticated is comprised of 3 bits. The OMT design for L5I is similar to the L5Q OMT. The number of bits transmitted per authenticated message is different, however. Figure 7 shows an example structure of an L5I OMT. Appendix B offers more information detailing all different OMTs used in the L5I case along with their composition. For L5I, OMT1 refers to the current salt and root/intermediate key for TESLA keychain. OMT2 is a level 2 signature of OMT1. These two messages are broadcast most frequently as they are the minimum amount of information needed to begin authenticating the WAAS message. From a cold start there is one more step that precedes step 1 in Figure 8. In this case, the receiver will first try to recursively use the one-way function until it finds its current authenticated root/intermediate key. In the case that it reaches the maximum number of iterations of the one-way function and does not find the root/intermediate key, it then listens for the broadcast of OMT1 and OMT2. The check of OMT2 validating OMT1 is the same check as 1 in Figure 8.

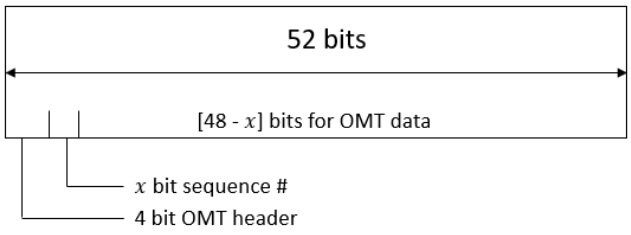

Figure 6: L5Q OMT message structure

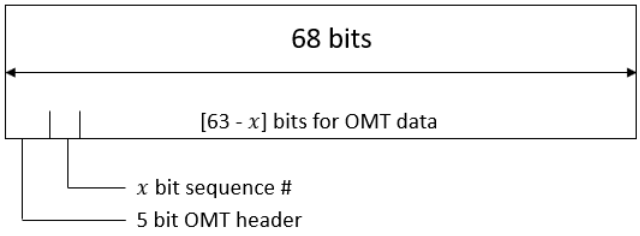

Figure 7: L5I OMT message structure

Receiver Authentication Protocol from cold start

1. Check if stored current level 2 public key authenticates received level 2 signature
   a. If 1. fails, go to 2.
2. Check if stored next level 2 public key authenticates received level 2 signature
   a. If 2. passes, throw away previous level 2 public key and this becomes current level 2 public key
   b. If 2. fails, go to 3.
3. Wait to receive current level 2 public key and level 1 signature for key
   a. If level 1 signature for current level 2 key does not pass, try stored next level 1 public key
      I. If 3a. Passes, throw away level 1 public key and this becomes current level 1 public key
      II. If 3a. Fails, wait for current level 1 public key and go to 4.
4. Authenticate self-signed current level 1 public key and check if the hashed value exists in the installed firmware

Figure 8: Receiver authentication protocol from cold start

MCOS: Monte Carlo OTAR Simulator

Now that the PKI and KM have been introduced, there is a need to test the feasibility of these designs. Some major questions that need to be answered include how long it will take to receive a given set of messages after a cold start given the environment (C/N_0) and what is a message schedule that will allow users to receive important OTAR information in the quickest time. In order to answer these questions, the Monte Carlo OTAR Simulator (MCOS) was built. There are several advantages with creating a simulation environment for receivers to test OTAR message reception. Different OMT messages and OMT message configurations can be tested, allowing the designer to iterate upon different strategies and measure the effectiveness of each design. Different broadcast algorithms can be tested to minimize the amount of time it takes for a receiver to become fully operational in an authenticated context given different starting states of OTAR knowledge. Sensitivity analyses can be performed to measure the effect of the aforementioned variables and statistically relevant results can be derived from large scale simulations.

MCOS is broken up into 3 stages. The first stage is the definition of the OMT and each message's bit allocation. The second stage is the implementation of the broadcast algorithm that creates a sequence of OTAR messages to be delivered from the WMS through a WAAS satellite. The third stage is the user simulation which tracks how many messages it received and how long it took to receive each set of messages. The first stage is covered in the previous sections and in the appendix of this work. The following sections cover the 2nd and 3rd stages in more detail.

MCOS allows the user to define a broadcast scheduling algorithm that processes when each OMT will be sent over the air. Fixed interval algorithms can be implemented that allows the designer to fix a value to how often each message is delivered. Designers also have the option to input a custom broadcast algorithm. One such algorithm is similar to packet fair queueing [20] and was used in the analysis of this PKI. Packet fair queueing has two distinct advantages for scheduling broadcasts: designers can designate the level of importance of each OMT or each set of OMTs through the assignment of weights and the algorithm will output a broadcast that delivers the messages with a bandwidth allocation proportional to the weights set. Packet fair queueing also attempts to evenly distribute the OMTs and avoids "bursty" message patterns. This helps minimize the time it takes to receive the most important OMT and the time it takes to receive all OMTs. An analysis of how to weight the level of importance for each OMT is carried out later in this work.

The final stage of MCOS is the user simulation. This stage creates a set of users that are attempting to receive the broadcast created in stage 2. These users all start from a cold start and are evenly distributed to start listening to the broadcast at different times. If the simulated broadcast is made sufficiently long, this process gives statistically relevant information concerning how long it takes to receive any given set of OMTs. In addition to this, the users all work in a specific (C/N_0) environment that may or may not lead to the loss of parts of the OMTs. This (C/N_0) value translates to a Page Error Rate (PER) that effects their ability to receive all messages in the broadcast stream. Example outputs can be seen in Figure 11 - Figure 15 in the analysis section of the paper. The derived results, such as mode and average time to receive a message can then be used to iterate on either the broadcast algorithm or OMT parameters. Each simulation can be run in a number of seconds on an i7 processor depending on the broadcast length and number of users. Moreover, this code is open source and made available through the Stanford GPS Lab github page: https://github.com/stanford-gps-lab/mcos.

Analysis

With a PKI and KM architecture along with MCOS, design parameters can be iterated upon to achieve an OTAR service with a desired performance. Two cases, one for each channel and scheme implementation, are defined for this study. The schemes use the key sizes defined above in Table 3 and Table 4 .

Broadcast Weight Analysis

As mentioned above, a packet fair queueing algorithm requires the input of weights for individual OMTs or sets of OMTs in order to generate the broadcast. These weights are based upon the rate at which the information the OMT delivers is updated or changed. The information with the shortest cryptoperiod, such as the level 2 key information for ECDSA, is more likely to be unknown to a receiver starting from a cold start than longer term information such as level 1 keys. Because of this, OMT1 and OMT2 (Table A - 1) are given weight w_{12} and the rest of the messages are given weight w_{rem}, or remaining weight. The same is done for both scheme implementations. In the case of TESLA, the keychains have the shortest cryptoperiod and so OMT1 and OMT 2 (Table B - 1) are given weight w_{12}, OMT3 and OMT4 (current level 2 key) are given w_{34}, and the rest of the messages are given weight w_{rem}. For this analysis, w_{12}/w_{34} is set to be 2.

This analysis looks at the ratios of these values and how they impact several performance metrics. For ECDSA, the time to receive an authenticated level 2 key and the time to receive all OMTs, and for TESLA the additional metric of the time to receive an authenticated keychain. All analyses were run assuming perfect data demodulation (PER=0) and were run on a scale of increasing weight ratios (w_{12}/w_{rem}). The results are seen below in Figure 9 and Figure 10. In these figures, the average values are plotted as well as the region bounded by minimum and maximum reception times observed in the simulation. In all cases, as the weighting ratio increases, the time to receive the higher weighted messages decreases, while the total time overall increases. This decrease in the time to receive OMT1 and OMT2 approach a limit, however, and as the weighting ratio increases, the sparseness of the other messages becomes very apparent as the time to receive all messages rises. According to the WAAS MOPS [21], a valid position fix must be obtained within 5 minutes of a receiver receiving power. With this in mind, along with the fact that it takes a receiver some time to acquire the signal, a desirable weighting solution would enable a receiver to receive OMT1 and OMT2 in all cases within 5 minutes. This threshold is marked by a black dashed line in the figures.

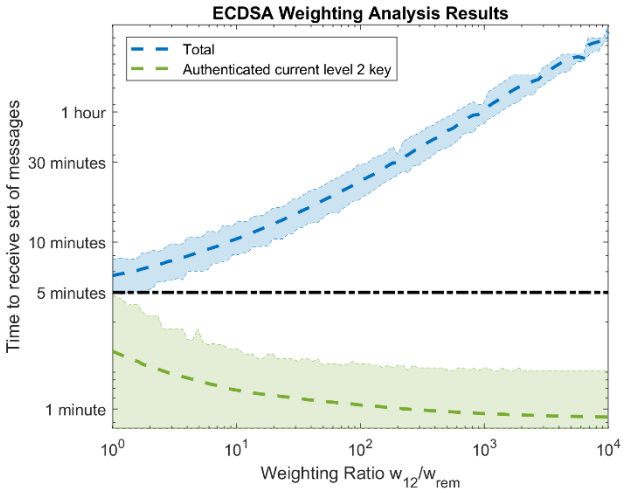

Figure 9: ECDSA Impact of weighting on OMT reception

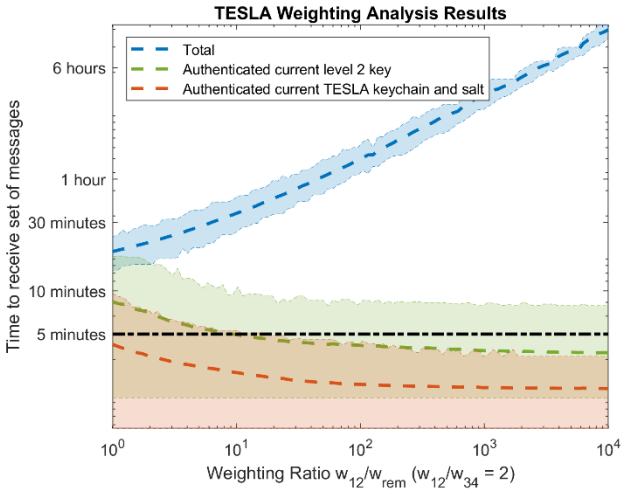

Figure 10: TESLA Impact of weighting on OMT reception

For the rest of the paper, a weighting ratio of $w_{12}/w_{rem} = 100$ is chosen for the rest of the simulations. Table 5 shows several performance parameters related to the chosen weighting ratio.

Table 5: Broadcast results for both cases

Scheme	Average time to receive all messages	Average time to receive current authenticated level 2 kay	Average time to receive current TESLA keychain and salt
ECDSA L5Q	23.15 min	63.62 sec	N/A
TESLA L5I	83.16 min	252.57 sec	134.18 sec

Histogram results with PER = 0

The resulting metrics for each of the cases using the chosen weighting values are showcased in this section. In each case, there are figures that show the simulation results with respect to time to receive all messages, time to receive OMT1 and OMT2, and time to receive OMT3 and OMT4 for TESLA. These figures are normalized histograms showing the counts of how long it took to receive each set of messages for each simulated receiver. In all cases, the chosen weighting parameters yield a time to receive less than 5 minutes for the OMTs with the shortest cryptoperiods. Many of the results appear bimodal in nature with a clear distinction between which mode is more common than the other. This is due to the quasi-periodic output from the broadcast algorithm.

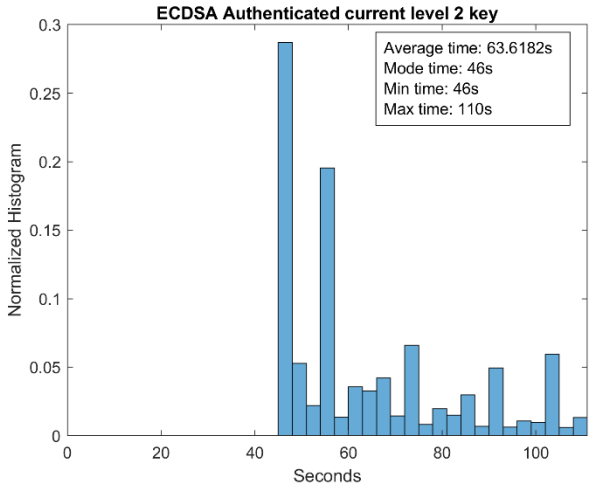

Figure 11: Times to receive authenticated current level 2 keys for ECDSA

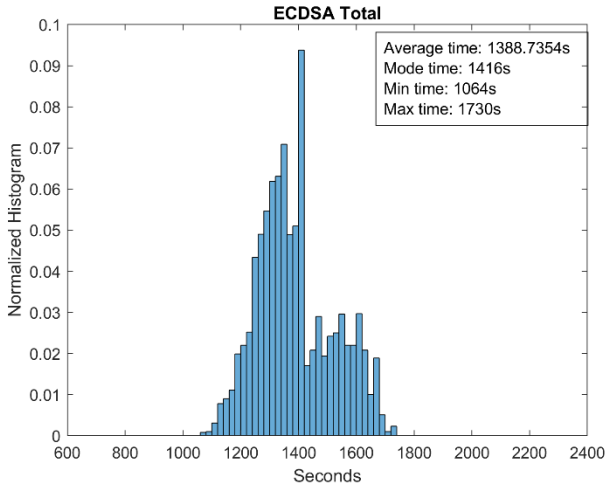

Figure 12: Times to receive all OMTs for ECDSA

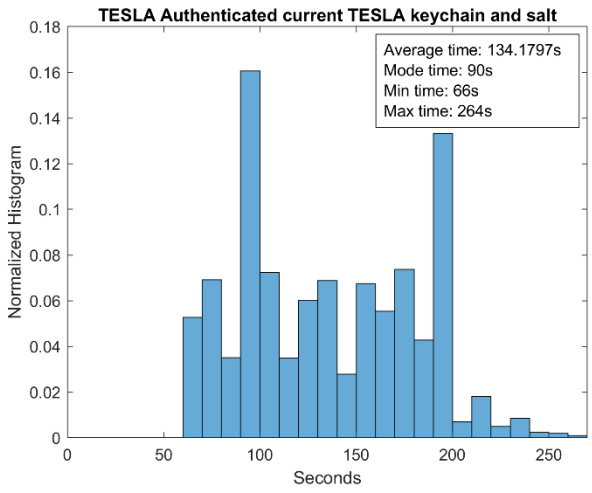

Figure 13: Times to receive current keychain and salt for TESLA

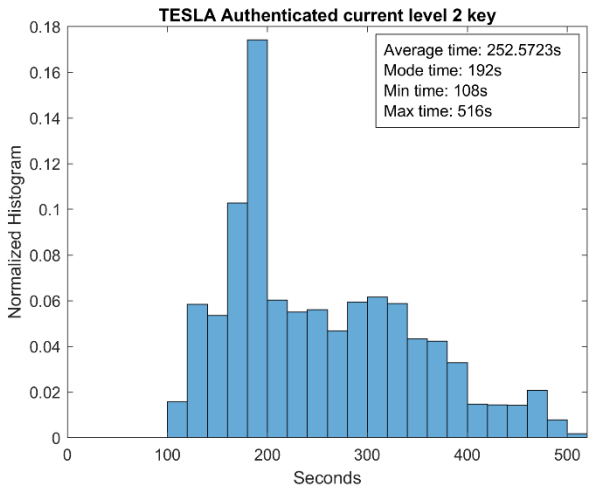

Figure 14: Times to receive authenticated current level 2 keys for TESLA

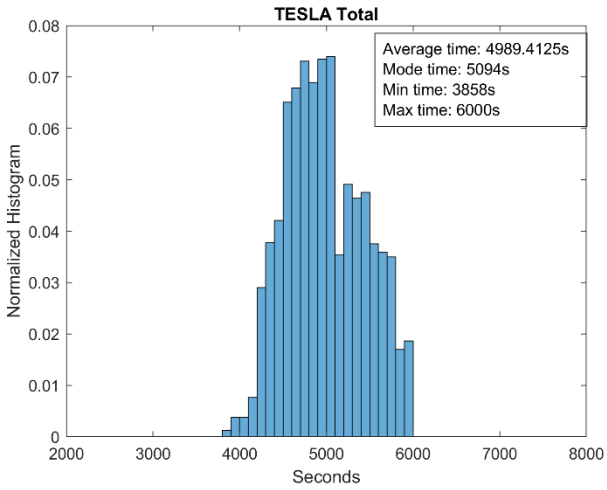

Figure 15: Times to receive all OMTs for TESLA

Effect of PER on both Schemes

All simulations that have been performed up until this point have assumed a perfect demodulation of the signal with no errors. An analysis was performed over a range of page error rates for each case. For each PER, a full simulation is run, producing histograms similar to the results shown in the previous section. These results are then aggregated to produce Figure 16 and Figure 17. Similar to Figure 9 and Figure 10, the average time is plotted with a dashed line and the region spanning from the minimum to maximum time is shaded in. As expected, an increase in PER increases the average time it takes to receive each message. The minimum time does not increase as there is still a chance that all relevant parts of each OMT will be received correctly, but the probability of correctly demodulating all parts of a message quickly deteriorates as the PER increases past 10^{-3}. This region roughly corresponds to a C/N_0 of 27.5 to 29 dB-Hz for WAAS L5, a signal using FEC $1/2$ rate constraint length 7 Viterbi encoding at 250 bits per second [22]. Since this is a Monte Carlo simulation, it does not show all possible outcomes, and so jumps seen in the maximum and minimum values seen in these figures would not be present for run times

that are sufficiently long. This goal of this simulation is to capture a trend, not absolute bounds in a theoretical sense.

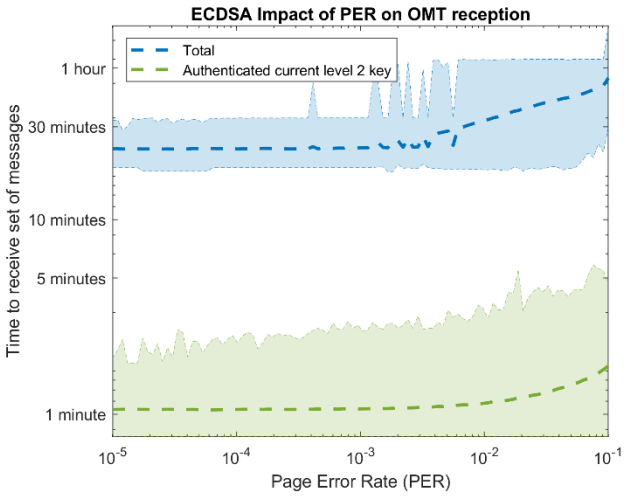

Figure 16: ECDSA Impact of PER on OMT reception

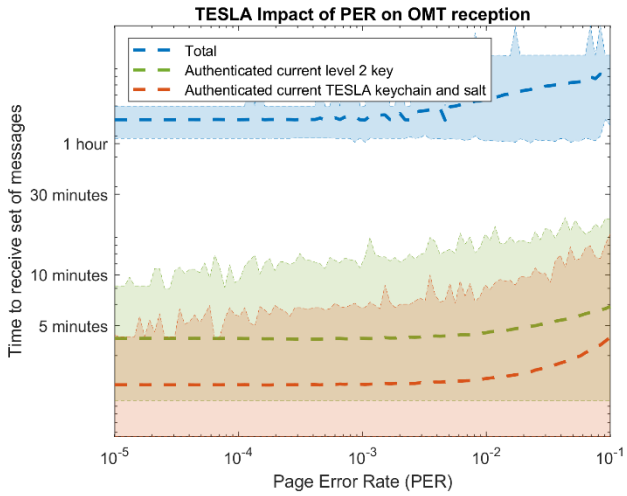

Figure 17: TESLA Impact of PER on OMT reception

Broadcast Algorithm Analysis

This broadcast algorithm used up until this point will be referred to as 'PFQ-Standard' in this work, referencing the packet fair queuing algorithm proposed in [20]. For all OMTs, the message length is longer than the OTAR frame provided and so they are split into a sequence of messages. These sequences are broadcast sequentially, meaning that if OMTx is broken into OMTx_1, OMTx_2, and OMTx_3, these are broadcast in that order and not interrupted by other messages. This analysis takes a look at two alternative broadcast algorithms that alleviate this restriction. These analyses only look at the ECDSA case.

The first algorithm, referred to as 'PFQ-Split', completely does away with the assumption that the messages must be broadcast sequentially. Since a receiver is capable of constructing the full OMT if they are broadcast in any order because of the associated sequence number, the algorithm does not specify that they must be broadcast in any particular order. In 'PFQ-Standard', weights are distributed to full OMTs. The equivalent happens in this case, where the weights are distributed to each subpart of each OMT. Resulting histograms for this algorithm is shown below in Figure 18 and Figure 19.

The second algorithm, referred to as 'PFQ-Semi-Rigid', acts similarly to 'PFQ-Standard', the only difference being that the maximum time between delivery of important OMTs can be set. For instance, in the case where OMT1 and OMT2 are the most important messages, this algorithm allows the designer to specify the maximum time interval between when they are sent, effectively ensuring an upper bound on time to receive these crucial messages. This algorithm allows the broadcast to interrupt other messages temporarily while the important messages are sent. For this study, a maximum interval time of 60 seconds for OMT1 and OMT2 was set. Resulting histograms for this algorithm is shown below in Figure 20 and Figure 21.

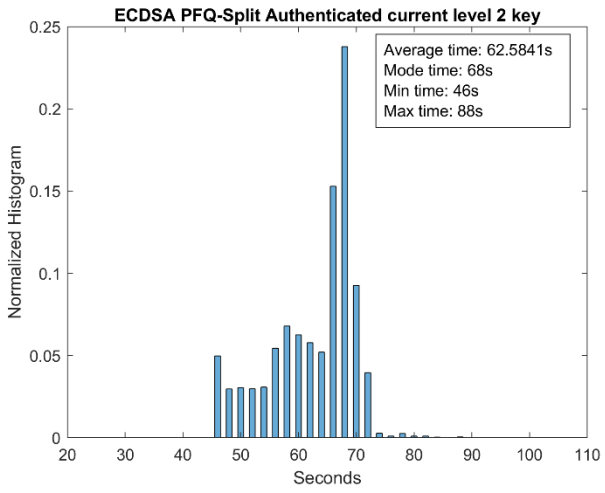

Figure 18: ECDSA PFQ-Split Authenticated current level 2 key reception times

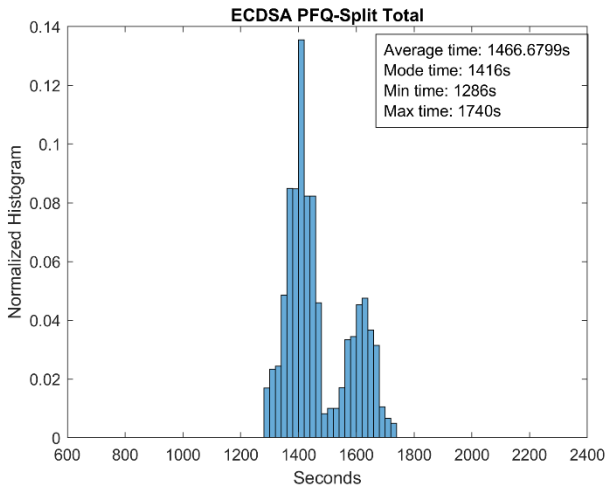

Figure 19: ECDSA PFQ-Split times to receive all OMTs

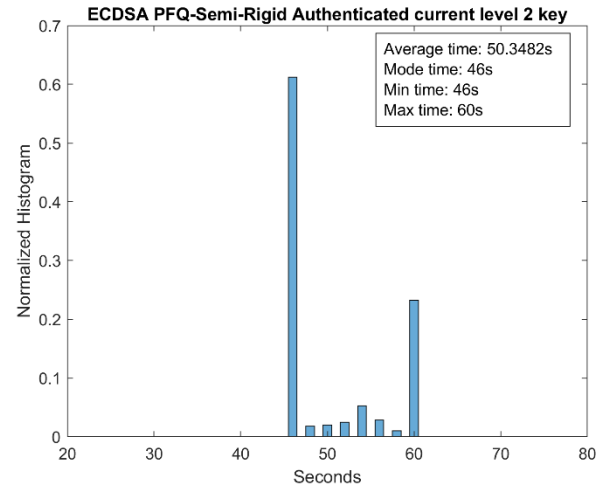

Figure 20: ECDSA PFQ-Semi-Rigid Authenticated current level 2 key reception times

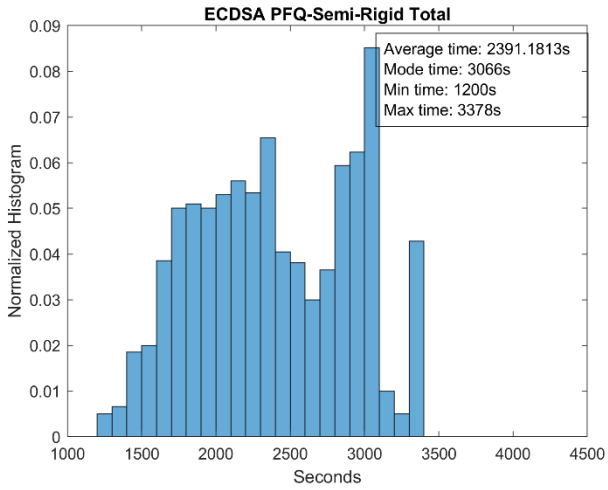

Figure 21: ECDSA PFQ-Semi-Rigid times to receive all OMTs

Table 6: Broadcast algorithm analysis results

Broadcast Algorithm	Average time to receive authenticated level 2 key	Max time to receive authenticated level 2 key	Average time to receive total OTAR digest
PFQ-Standard	63.6 sec	110 sec	23.1 min
PFQ-Split	62.6 sec	88 sec	24.4 min
PFQ-Semi-Rigid	50.3 sec	60 sec	39.9 min

Referencing Table 6, there are advantages to these different broadcast algorithms when compared with 'PFQ-Standard'. For 'PFQ-Split', the average time to receive an authenticated level 2 key is slightly decreased while slightly increasing the time it takes to receive all OMTs. This is because the low weight of all other messages is

distributed evenly, decreasing the frequency of receiving a full message. 'PFQ-Semi-Rigid' sets the maximum time between authenticated level 2 key delivery to 60 seconds and achieves better performance for receivers starting from a cold start with no level 2 key information. The penalty is a much longer time to receive all other OMTs. Because the cryptoperiod for level 1 key information is much longer, this may be an acceptable sacrifice for a guarantee on maximum level 2 information latency.

Conclusions

In this paper, two different authentication schemes were examined as possible candidates for SBAS data authentication. An ECDSA scheme was chosen as a candidate for the L5Q channel, and a TESLA scheme was chosen as a candidate for the L5I channel. Each scheme's parameters were designed so that the schemes would be secure from a cryptographic standpoint using the analyses produced in the latest research. In the case of TESLA, the concept of using concurrent TESLA instances was introduced to alleviate the requirements concerning loose time synchronization between receiver internal clocks and GPS time.

The choice between an I/Q channel implementation in the end requires many more considerations than those covered in this work. The Q channel requires either extra power to be delivered from the satellite, or power to be diverted from the I channel, asking for a sacrifice from users who are unable to use the authentication signal. Moreover, this Q channel would be competing on an increasingly crowded L5 band and so any extra power added to the L5 band would require that the link budget to existing users still closes. A challenge with implementing SBAS data authentication on the L5I channel is the limited bandwidth that is available on the signal. Any addition of SBAS data authentication will lead to a decrease in availability for the system, but this by and large can be mitigated with proper design of the TBA and procedures in the event of a failed authentication. Since each L5I authentication signature necessarily authenticate larger blocks of data, an incorrectly demodulated message or signature could lead to a failed authentication event. Each user platform (E.g. aviation, timing services, etc.) may develop a procedure in the event of such failed authentication events. A declaration of suspected spoofing may be tied to the monitoring of other metrics such as AGC and solution sanity checks, but these are beyond the scope of the paper and left for future work.

A PKI was developed around the two chosen schemes that allows public keys to be updated solely over the air, not requiring any extra maintenance or other communication channels. This PKI was developed in a way that allows the SBAS provider to distribute and allocate levels of security and more firmly safeguard against potential attacks. Central to both scheme PKIs was the use of a trusted level 1 source, either the SBAS provider or a 3rd party entity, such as a current day certificate authority, whose business model is central to the idea of protecting sensitive information. A secure data base is available to the public, giving access to the hashed values of future level 1 public keys that would be installed in a receiver's firmware by the receive manufacturer. These level 1 keys would be unlocked using over the air messages delivered through the KM architecture, allowing the receivers to authenticate any level 2 keys introduced by the SBAS provider. In the case of ECDSA, a two-level system was introduced, allowing for the WMS to generate its own level 2 private/public key pairs. These level 2 public keys were authenticated by level 1 signatures using extra bits available in the signature messages. In the case of TESLA, a 3-level system was introduced, presenting a third layer comprised of the TESLA keychains. Level 2 signatures, as opposed to level 1 signatures, were still used to authenticate the root/intermediate keys of these keychains. The main reason for this being that level 2 signatures have bit lengths significantly shorter than level 1 signatures, which allow a receiver on cold start to authenticate the current keychain much quicker than it would be able to otherwise.

Integral to the design of the PKI is the KM architecture that allows the user to receive all required information to function using the chosen PKI. This architecture is designed for the purpose of doing key management solely over the air to avoid costs involving extra maintenance, high internal non-volatile storage required for the receiver, and expanding a receiver's access to other networks. Over the air methods also reduces the number of potential weak points that may be exploitable if 3rd party networks are used. A series of OTAR message types was derived for each scheme with the intent on delivering all required PKI information using minimal bandwidth.

In order to test this KM architecture, an analysis tool, MCOS, was developed based on Monte Carlo methods to assess a receiver's performance in correctly demodulating and receiving all necessary OMTs to authenticate the SBAS data. The tool proved useful in testing different OMT configurations, TBAs, authentication schemes, PKIs, broadcast schedulers, and a host of other important design parameters. A sensitivity analysis was performed using a variant of the fair packet queueing algorithm to determine the proper weighting to allocate to each OMT. A series of analyses were performed looking at the effect different PKI input parameters and demodulation error rates had on key performance metrics concerning the time to correctly demodulate important PKI messages. Finally, an analysis comparing different broadcast algorithms was carried out showing some potential adaptations to packet fair queuing.

Thus far, this paper has avoided suggesting any exact cryptoperiod for each level of keys, but the analysis performed can help guide this discussion now that estimates of how long it takes to demodulate these keys has been presented. Of all conclusions derived in this work, the most important result is that an SBAS data authentication system can perform key management solely over the air. This fact, along with the analyses that have been presented, will help further the design of SBAS data authentication systems.

Future Work

While this work provides preliminary results for KM feasibility, the authentication scheme parameters to be used in SBAS data authentication will continue to be refined, and so this tool and these analyses will be performed again as these schemes become realized. Over the course of this work, MCOS underwent several upgrades that have allowed it to perform large scale simulations in shorter periods of time while using less memory. MCOS will continue in its development as more capabilities are added through the open source format. Potential future additions include a closer look at how the weights are chosen for packet fair queuing. The weighting appropriations were decided based on heuristics and can become defined to a higher level of fidelity utilizing a properly design cost function. With regards to optimization, the function of weighting parameters to output metrics is not convex and so a long-term goal may be the application of swarm-based optimization techniques to identify a proper broadcast schedule.

Outside of solely using the extra bits in the signature messages for OTAR, there is potential to replace the current MT63 with an OTAR message. This could greatly improve the performance of an OTAR scheme and will be looked at in future revisions of this PKI. Lastly, parts of MCOS may be applied in the future to existing test beds for NMA currently used for evaluating GNSS data authentication systems and there are plans to integrate MCOS with the MAAST tool in the future.

Acronyms

- CA: Certificate Authority
- CBC: Cipher Block Chaining
- CRC: Circular Redundancy Check
- ECDSA: Elliptic Curve Digital Signature Algorithm
- EGNOS: European Geostationary Navigation Overlay System
- FEC: Forward Error Correction
- KM: Key Management
- MAAST: MATLAB Algorithm Availability Simulation Tool
- MAC: Message Authentication Code
- MOPS: Minimum Operational Performance Standards
- NIST: National Institute of Standards and Technology
- NMA: Navigation Message Authentication
- OMT: OTAR Message Type
- OTAR: Over-The-Air-Rekeying
- PER: Page Error Rate
- PFK: Packet Fair Queuing
- PK: Public Key
- PKI: Public Key Infrastructure
- PLL: Phase Lock Loop
- SBAS: Satellite Based Augmentation System
- SHA: Secure Hash Algorithm
- TBA: Time Between Authentication
- TESLA: Timed Efficient Stream Loss-tolerant Authentication
- TOW: Time of Week
- WAAS: Wide Area Augmentation System
- WMS: WAAS Master Stations
- WNRO: Week Number Roll Over

References

[1] S. C. Lo and P. K. Enge, "Authenticating aviation augmentation system broadcasts," *Rec. - IEEE PLANS, Position Locat. Navig. Symp.*, pp. 708–717, 2010.

[2] J. T. Curran, M. Paonni, and J. Bishop, "Securing the Open-Service: A Candidate Navigation Message Authentication Scheme for Galileo E1 OS," *Eur. Navig. Conf.*, no. April, 2014.

[3] I. Fernández-Hernández, "Method and System to Optimise the Authentication of Radionavigation Signals," Patent EP14163902, 2015.

[4] P. Walker, "Galileo Open Service Authentication: A Complete Service Design and Provision Analysis," in *ION GNSS+*, 2015.

[5] I. F. Hernández, V. Rijmen, G. S. Granados, J. Simón, I. Rodríguez, and J. D. Calle, "Design drivers, solutions and robustness assessment of navigation message authentication for the galileo open service," *27th Int. Tech. Meet. Satell. Div. Inst. Navig. ION GNSS 2014*, vol. 4, pp. 2810–2827, 2014.

[6] A. J. Kerns, K. D. Wesson, and T. E. Humphreys, "A blueprint for civil GPS navigation message authentication," in *IEEE PLANS, Position Location and Navigation Symposium*, 2014, pp. 262–269.

[7] K. Wesson, M. Rothlisberger, and T. Humphreys, "Practical cryptographic civil GPS signal authentication," *Navig. J. Inst. Navig.*, vol. 59, no. 3, pp. 177–193, 2012.

[8] G. Caparra, S. Sturaro, N. Laurenti, C. Wullems, and R. T. Ioannides, "A Novel Navigation Message Authentication Scheme for GNSS Open Service," *Proc. 29th Int. Tech. Meet. Satell. Div. Inst. Navig. (ION GNSS+ 2016)*, no. March 2017, pp. 2938–2947, 2016.

[9] I. Fernández-Hernández *et al.*, "Impact analysis of SBAS authentication," *Navig. J. Inst. Navig.*, no. August, pp. 517–532, 2018.

[10] G. Caparra, S. Ceccato, S. Sturaro, and N. Laurenti, "A key management architecture for GNSS open service Navigation Message Authentication," *2017 Eur. Navig. Conf. ENC 2017*, pp. 287–297, 2017.

[11] NIST, "FIPS PUB 186-4 Digital Signature Standard (DSS)," no. July, 2013.

[12] A. Perrig, R. Canetti, J. D. Tygar, and D. Song, "Efficient authentication and signing of multicast streams over lossy channels," in *Proceedings of the IEEE Computer Society Symposium on Research in Security and Privacy*, 2000, pp. 56–73.

[13] E. Barker, W. Barker, and W. Burr, "Recommendation for Key Management," *NIST Spec. Publ. 800-57*, pp. 1–142, 2007.

[14] E. Barker and Q. Dang, "NIST Special Publication 800-57 Part 3 Revision 1 Recommendation for Key Management Part 3: Application-Specific Key Management Guidance," p. 102, 2015.

[15] A. Neish, T. Walter, and P. K. Enge, "Quantum Resistant Authentication Algorithms for Satellite-Based Augmentation Systems," *Navig. J. Inst. Navig.*

[16] G. Alagic *et al.*, "Status report on the first round of the NIST post-quantum cryptography standardization process," 2019.

[17] E. Grumbling *et al.*, *Quantum Computing: Progress and Prospects*. 2018.

[18] A. Neish, T. Walter, and P. Enge, "Parameter Selection for the TESLA Keychain," in *ION GNSS+*, 2018, vol. 1.

[19] A. Perrig and J. D. Tygar, *Secure Broadcast Communication in Wired and Wireless Networks*. 2003.

[20] S. Hameed and N. Vaidya, “Efficient algorithms for scheduling data broadcast,” *Proc. 3rd Annu. ACM/IEEE*, vol. 5, pp. 183–193, 1997.

[21] RTCA, “Minimum Operational Performance Standards for Global Positioning System/Satellite-Based Augmentation System Airborne Equipment (SBAS MOPS),” 2016.

[22] M. K. Simon, J. K. Omura, and R. a Scholtz, “Spread Spectrum Communications Handbook,” 2002.

Appendix A: OMT Details for L5Q ECDSA Implementation

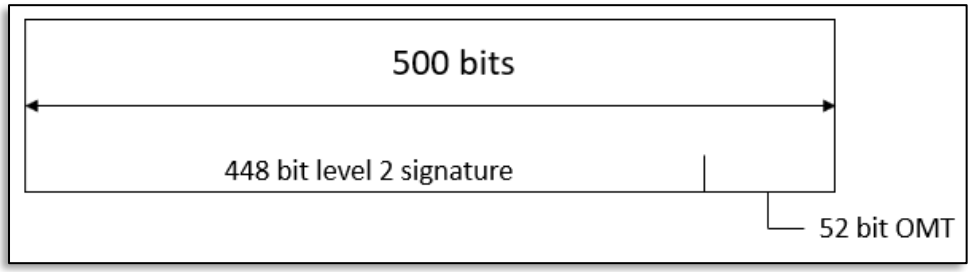

Figure A - 1: L5Q message structure

Table A - 1: OMT description for L5Q ECDSA implementation

OMT#	Description	Bit Allocation for each OTAR word	#bits/#OTAR words
OMT0	No OTAR information available		
OMT1	Current level 2 PK	OMT Header: 4, Sequence #: 3, Data: 44	259/5
OMT2	Level 1 signature of OMT1	OMT Header: 4, Sequence #: 5, Data: 42	930/18
OMT3	Next level 2 PK	OMT Header: 4, Sequence #: 1, Data: 46	78/2
OMT4	Level 1 signature of OMT3	OMT Header: 4, Sequence #: 5, Data: 42	930/18
OMT5	Expiration of current level 2 and level 1 PK	OMT Header: 4, Sequence #: 3, Data: 44	259/5
OMT6	Level 1 signature of OMT5	OMT Header: 4, Sequence #: 5, Data: 42	930/18
OMT7	Expiration of next level 2 and level 1 PK	OMT Header: 4, Sequence #: 1, Data: 46	78/2
OMT8	Level 1 signature of OMT7	OMT Header: 4, Sequence #: 5, Data: 42	930/18
OMT 9-11	Reserved		
OMT12	Current level 1 PK	OMT Header: 4, Sequence #: 4, Data: 44	456/9
OMT13	Level 1 signature of OMT12	OMT Header: 4, Sequence #: 5, Data: 42	930/18
OMT14	Next level 1 PK	OMT Header: 4, Sequence #: 4, Data: 44	456/9
OMT15	Level 1 signature of OMT14	OMT Header: 4, Sequence #: 5, Data: 42	930/18

Notes for Table A - 1:

The last column containing “#bits” refers to all bits needed to communicate the OMT (E.g. OMT Header + OMT data + etc.). All level 1 signatures use the current level 1 key. There are 52 bits available for OTAR in each signature frame. In most OMTs, there will be bits left over in the last OTAR word used to send the OMT. These bits may take any value and will be ignored by the receiver. OMT5 and OMT7 data fields use 34 bits to communicate the expiration time of each specified public key: 20 bits for TOW, 10 bits for GPS week, and 4 bits for WNRO. OMT5 report the expiration of the level 2 public key first and the level 1 public key second, both concatenated. PK refers to public key.

Appendix B: OMT Details for L5I Implementation

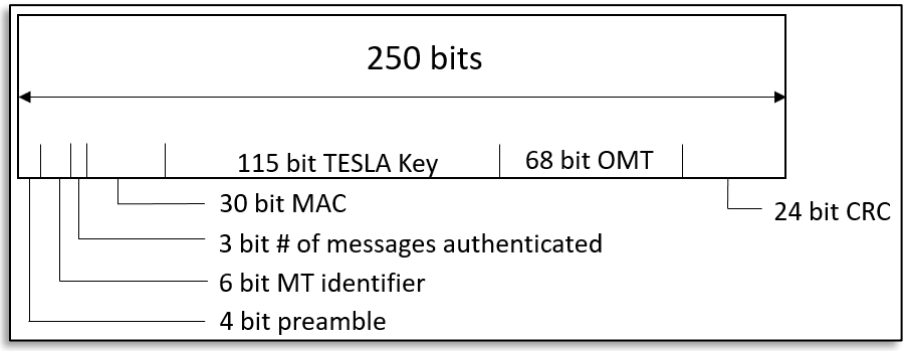

Figure B - 1: L5I message structure

Table B - 1: OMT description for L5I TESLA implementation

OMT#	Description	Bit Allocation for each OTAR word	#bits/#OTAR words
OMT0	No OTAR information available		
OMT1	Current salt and root or intermediate keys for TESLA keychain	OMT Header: 5, Sequence #: 2, Data: 61	166/3
OMT2	Level 2 signature of OMT1	OMT Header: 5, Sequence #: 3, Data: 60	512/8
OMT3	Current level 2 PK	OMT Header: 5, Sequence #: 2, Data: 61	252/4
OMT4	Level 1 signature of OMT3	OMT Header: 5, Sequence #: 4, Data: 59	894/14
OMT5	Expiration of current salt, TESLA keychain, and level 2 PK	OMT Header: 5, Sequence #: 1, Data: 62	114/2
OMT6	Level 1 signature of OMT5	OMT Header: 5, Sequence #: 4, Data: 59	894/14
OMT 7-8	Reserved		
OMT9	Next salt and root or intermediate keys for TESLA keychain	OMT Header: 5, Sequence #: 2, Data: 61	166/3
OMT10	Level 2 signature of OMT9	OMT Header: 5, Sequence #: 3, Data: 60	512/8
OMT11	Next level 2 PK	OMT Header: 5, Sequence #: 2, Data: 61	252/4
OMT12	Level 1 signature of OMT11	OMT Header: 5, Sequence #: 4, Data: 59	894/14
OMT13	Expiration of next salt, TESLA keychain, and level 2 PK	OMT Header: 5, Sequence #: 1, Data: 62	114/2
OMT14	Level 1 signature of OMT13	OMT Header: 5, Sequence #: 4, Data: 59	894/14
OMT 15-27	Reserved		

OMT28	Current level 1 PK	OMT Header: 5, Sequence #: 3, Data: 60	440/7
OMT29	Level 1 signature of OMT28	OMT Header: 5, Sequence #: 4, Data: 59	894/14
OMT30	Next level 1 PK	OMT Header: 5, Sequence #: 3, Data: 60	440/7
OMT31	Level 1 signature of OMT30	OMT Header: 5, Sequence #: 4, Data: 59	894/14

Notes for Table B - 1:

The last column containing "#bits" refers to all bits needed to communicate the OMT (E.g. OMT Header + OMT data + etc.). All level 1 signatures use the current level 1 key. There are 68 bits available for OTAR in each signature frame. In most OMTs, there will be bits left over in the last OTAR word used to send the OMT. These bits may take any value and will be ignored by the receiver. OMT5 and OMT13 data fields use 34 bits to communicate the expiration time of each specified public key: 20 bits for TOW, 10 bits for GPS week, and 4 bits for WNRO. OMT5 and OMT13 report the expiration of the salt/keychain first, the level 2 public key second, and the level 1 public key last, all concatenated. PK refers to public key.