ESTABLISHING TRUST THROUGH AUTHENTICATION IN SATELLITE BASED AUGMENTATION SYSTEMS

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF
AERONAUTICS AND ASTRONAUTICS
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Andrew Neish
August 2020

Abstract

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. As the adoption of GNSS technologies surges, knowledge of its vulnerabilities has become more common place and tools capable of disrupting and spoofing the service have become cheaper and more available. Open-source software is available on the internet that gives nefarious actors the ability to broadcast false (spoof) GNSS signals with little to no sophisticated knowledge of GNSS works. These tools make an attack on infrastructure and safety of life systems more accessible than ever. The GNSS community is developing methods to detect and mitigate spoofing threats and one such method is the use of cryptographic signatures to authenticate 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 strong solution to mitigate and detect intentional SBAS spoofing threats carried out through false signal generation.

This thesis presents a cryptographic authentication process designed to secure SBAS data. These pages include everything from how messages are signed at the broadcast data-level to how keys are securely distributed to SBAS users. Most importantly, this authentication process protects users while maintaining the 6 second Time-To-Alert (TTA) requirement with negligible impact to availability and continuity. The data authentication techniques presented in this thesis are currently being considered for SBAS systems such as the Wide Area Augmentation System

(WAAS) and the European Geostationary Navigation Overlay Service (EGNOS).

The contributions of this thesis can be summed up as follows:

This thesis is the first to design an SBAS authentication system that preserves integrity against spoofing threats and message loss. Many authentication solutions to date group messages together to form a single signature of a single batch of messages. Unfortunately, this means that if a single message is lost in that batch, all other messages in the same batch can no longer be authenticated by the user which could dramatically decrease the availability and continuity of the SBAS service. To address this, previous authors have adopted a methodology by which users would use messages even if those messages could not be authenticated. This provides an avenue by which an attacker could inject false messages into an aviation receiver, circumventing the cryptographic signatures altogether. This thesis presents an authentication scheme that protects receivers from harmful spoofed messages while maintaining high availability and continuity even in the case of message loss.

This thesis is the first to develop a detailed key delivery and management process that is interoperable and forward compatible for all SBAS receivers. Prior to this thesis, there was little treatment of public key infrastructures and how they could be designed for GNSS authentication and no work had been formally done examining how SBAS would manage the delivery of authentication keys. SBAS is a broadcast service operating at a very low data rate and aviation receivers do not have access to third-party networks to retrieve such key management information. This thesis develops a method that delivers keys over the air to users and paves the way for how this key management architecture can be standardized for all SBAS internationally.

Finally, this thesis is the first to complete a detailed analysis on quantum computing threats to authentication techniques for SBAS. The arrival of large scale quantum computers in the future will threaten many of the authentication algorithms in use today. This thesis is the first to analyze these quantum computing threats against authentication algorithms in the context of SBAS and points to potential solutions as these threats evolve.

Acknowledgements

This work would not be where it is today without the help and support of so many. I have found myself fortunate enough to have worked with some of the best and brightest in this field and have been gifted with an incredibly warm community full of family, friends, and mentors.

I feel deeply indebted to my advisors, Professor Per Enge, Professor J. David Powell, and Professor Todd Walter for helping me get here. I was an aspiring engineer with a meandering career path who had little awareness of the field of navigation before I met Per. Your inspiring teaching, generous personality, and the bright people you surrounded yourself with drew me in and I am very grateful to have worked with you. Per, you taught me many lessons to live by in the few years that I knew you and every day I'm inspired by the impact you left on this community.

Professor J. David Powell, you stepped in to advise the students in the lab when Per had passed and I am extremely grateful for your generosity and brilliant insights. You always made himself available to me and encouraged me to finish this work. Not to mention you're a great skiing partner and an all around great person to spend time with.

Professor Todd Walter, you've been my technical advisor from the very beginning. Without your guidance, this work wouldn't be anywhere near where it is today. You always encouraged me to keep things practical and realistic. With your experience with WAAS, not only were you able to make advise many of the technical aspects of this work, but also intuit what needed to be done to bring the SBAS community together to consider these solutions. With your encouragement and mentorship I had the opportunity to work with members of the SBAS community through several working groups and I owe you many thanks for those growing opportunities. I look forward to staying in touch as time goes on.

I would also like to thank my third dissertation reader, Professor Grace Gao. I remember first meeting you at an ION conference several years ago and was struck with how warmly you welcomed me into the GNSS community. As a graduate from the Stanford GPS lab yourself, I will always appreciated the special bond we all share. Now that you're here with a lab of your own, I'm reminded of the bright future that lays ahead. It's always exciting to see what your lab is accomplishing and I'm looking forward to the many reunions each ION will bring.

I would also like to thank many of my mentors and collaborators in the lab who have always been there when I've had questions: Dr. Eric Phelts, Dr. Sam Pullen, Dr. Juan Blanch, Dr. Yu-Hsuan Chen, and Dr. Sherman Lo. I was incredibly lucky to have such a vast range of expertise that I could draw from and you were always kind enough to give me your time.

In addition to the senior technical staff in the lab, I also learned a great deal from my colleagues and fellow lab mates over the years: Kaz Gunning, Adrien Perkins, Tyler Reid, Fabian Rothmaier, and Shiwen Zhang. Many of us specialize in different topics and I learned a great deal from each of you. Whether it be high-precision GNSS, RF and embedded hardware, LEO PNT, Multi-modal spoof detection, or multipath mitigation, the collection of knowledge and experience in this group is astounding and I'm lucky I got the chance to pick your brains from time to time.

During my PhD, I had the opportunity to collaborate with many amazing people working towards the improvement of GNSS services: Barbara Clark, Ken Alexander, Don Wilkerson, Jason Burns, Jed Dennis, Andrew Hansen, Chris Hegarty, Karl Shallberg, Ignacio Fernández-Hernández, Mikael Mabilleau, Eric Chatre, Jim Gillis, Andrew Binder, Mark Mendiola, Boris Pervan, Okuary Osechas, Santiago Perea Diaz. Whether we collaborated directly on a project or simply enjoyed a few conversations at some of our meetings, you all have shaped this work into being what it is today.

I would also like to deeply thank my funding source for this work, the FAA. You are some of the hardest working and most patriotic people I've ever had the fortune to work with. I appreciate your support for this work and the expertise of the people

I had the opportunity to work with.

I've been fortunate to have had several great mentors over the years and I want to thank you for continuing to believe in me: Stevan Spremo, Darin Brekke, Professor Steve Robinson, and Professor Mohamed Hafez. You've shaped my career path in major ways and have continued to be there for me whenever I needed guidance.

Although this work took up a significant portion of my life over the past several years, I would not have been able to get through it without my friends who were there with me every step of the way. This acknowledgements section can never fully express the gratitude I have for you. You've been there with me through the highs and the lows and there is no way this could have been accomplished without you. My friends from UC Davis: Ed and Shelley Necoechea, David and Haley Nelson Meade, Cory and Mimi Warshaw, Daniel Pulache, Logan Halstrom, Elena Kolorov, Ashley Coates, Hannah Dalrymple, Sheida Hosseini, Holger Teichgräber, Josh and Kelsey Barram. My friends from Stanford: Flora Mechentel, Corinne Lippe, Ben Estacio, Kaz Gunning, Krystina Tran, Adrien Perkins, Kate Stuckman, Javier Stober, Krishna Venkataraman, Fabian Rothmaier, Tyler Reid, Ashley Clark, Aditya and Megan Mahajan, Jeremy Morton, Alex Wolff, Camille Townshend, Sam Avery, Erin Gudger, Harsh Patel, Wally Maier, Anthony Bombik, and Brian Munguía. My friends from Blue Origin: Max Winn, Allison Quinn, Patrick Bennett, Rob McBride, Dan Greenheck, Sarah Knights, and Heather Nelson. To all of you, thank you so much for your support throughout the years and I'm looking forward to sharing the next adventure with you!

Before I thank my family, I need to recognize another family that I have embraced along the way. Ted, Sue, John, and Chris Larson, you opened up your home to me on more than one occasion and you have done so much for me and my family throughout the years. I want to thank you from the bottom of my heart for everything. It's hard to imagine me being here today without your generosity and compassion.

And finally, I want to thank my family. Mom, you've always been an inspiration to me and you've always encouraged me to follow my dreams. Dad, I've always enjoyed sharing my passions with you and you've always been there to listen. I appreciate everything that you both have done for me over the years. And to Jenna, my amazing sister. I'm not quite sure how we came from the same parents. Your imagination,

creativity, passion for justice, shaded humor, and so many other traits continue to inspire me everyday. You may be my younger sister, but I look up to you in more ways than you may realize. I love you all so much.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Twenty years ago, a seminal report was published by the US Department of Transportation's Volpe institute warning of the impending threats that jamming and spoofing presented to large parts of America's infrastructure [2]. At that time, critical systems were just beginning to depend upon GPS as a source for precise timing and positioning. Today we see that our dependence upon GNSS has only accelerated. When we look around the technological world in 2020, it's hard to find an application that doesn't rely upon GPS. Everything from our phone alarm clocks that wake us up in the morning to the AC power that keeps our lights on traces back to this invisible, ubiquitous utility. In response to the Volpe report, members from academia, industry and governments have identified many potential solutions against these threats. One of these anti-spoofing solutions that has seen progress in the intervening time is the application of cryptographic authentication for GNSS. Today, there are plans to include data authentication on Galileo's Open Service (OS), GPS is testing signal authentication on the upcoming Navigation Test Satellite (NTS-3) mission, and members of the International Civil Aviation Organization (ICAO) are working to standardize data authentication systems for all Satellite Based Augmentation Systems (SBAS). This introduction traces back through time to witness the progress that has been made over the years and illuminates how we arrived at the designs being implemented today. Above all, the first part of this introduction represents something of an anthology in GNSS authentication with full awareness that there are still many stories currently being written.

Today, Global Navigation Satellite System (GNSS) provide positioning and timing services to people all over the globe. This meteoric rise from the first user of GPS in the 1980's to billions of users 30 years later with multiple constellations can be understood from several key design choices made by GPS early on. GNSS is a one-to-many broadcast service, delivering signals from satellites orbiting 10s of thousands of kilometers above the surface of the Earth. By the time these signals reach users on the Earth, the signal power is below the noise floor and receivers rely upon clever signal processing techniques to make use of them. The civilian signals have publicly available Interface Specifications (ISs) or Interface Control Documents (ICDs) that allow anyone to develop and market GNSS receivers. Because of this and the rapid rise in microelectronics, GNSS has become an important technological tool used in everything from cell phones to autonomous cars. These aspects of GNSS that encouraged mass adoption come with consequences, however.

With the means to receive and demodulate GNSS signals made publicly available, enough information is present that allows individuals to generate GNSS signals of their own. Until recently, this has proved to be a benign and even beneficial utility. From this, a market for high precision GNSS signal simulators was born to provide receiver manufacturers and end users the means to test their systems under a wide variety of conditions before bringing their products to market. However, with recent advancements in radio frequency microelectronics, generating GNSS signals has become more accessible at a lower cost over time. The threat of broadcasting these fake signals has become a great cause for concern as the technology required to spoof GNSS signals has outpaced the GNSS community's efforts to detect and mitigate these threats.

In response to this emerging hazard, several spoofing mitigation and detection strategies have been explored for application at the end user as well as the service provider. For the end user, these techniques include monitoring internal receiver metrics, the adoption of independent positioning and timing resources, and hardware augmentations to detect true GNSS signal direction-of-arrival. GNSS service providers have also come to the table to help secure their publicly broadcast signals. This introduction presents the efforts to provide authentication of GNSS ranging signals and data channels by both the core GNSS constellations as well as the augmentation systems that support them. The thesis later goes on to focus on data authentication for SBAS, specifically.

Cryptographic authentication uses signatures in the broadcast signal to provide confidence to end users on where the signals were generated. These signatures are created using a secret key that is privately held by the service provider and the signatures are verified using the secret key's corresponding public key that is available to all end users. The field of cryptographic authentication has been developed with the capabilities of modern internet data rates in mind. In other words, high data rates allow for longer key and signature lengths which typically leads to higher security. Because GNSS services are broadcast systems and the data rates are much lower than nominal internet protocols, special care should be taken when designing and tailoring existing authentication schemes for use in GNSS to ensure that they are sufficiently secure. In addition, interoperability between different systems has been a goal for different core constellations and SBAS systems. SBAS in particular, has taken great strides to standardize the authentication methods being considered for implementation. These authentication systems, once implemented, are envisioned to last for decades and so incorporating authentication schemes that are resistant to future computing vulnerabilities, such as quantum computing, is also considered in this thesis.

The Volpe report [2] is widely recognized as the original document motivating the past two decades of work in anti-spoofing and anti-jamming. In this report, the Department of Transportation outlined many concerns that they had with respect to GPS threats and their effect on the transportation sector. While the report focused on GPS, there was also mention of vulnerabilities in the Wide Area Augmentation System (WAAS), Local Area Augmentation System (LAAS) and other GPS correction services. In this report, they refer to an internal memorandum written by Edwin Key from MITRE [3] that identifies several spoofing mitigation strategies including IMU cross checks, time-of-arrival discrimination, and angle-of-arrival discrimination. While the report concluded that "the best anti-spoofing technique is probably the use of a multiple-element antenna to measure the angle-of-arrival of all received signals," the report also explicitly mentions the use of backwards compatible cryptographic authentication on civilian signals. There were several papers and patents in the public literature before the Volpe report [4], [5], and several internal presentations and reports that looked seriously at the prospect of anti-spoofing for civilian GPS [6], [7], but it is fair to say that this report was highly influential in spurring development towards anti-spoofing technology and methods.

In 2003, Logan Scott published a paper on incorporating authentication in civilian GPS signals [8]. In this paper, Scott recognized the democratization of software and hardware that would soon enable the masses to produce their own GPS signals. The solutions offered in his paper were to add cryptographic authentication to the data and signal levels of the civil GPS signals to protect them from spoofing attacks. He offered 3 levels of protection when it came to GNSS authentication:

- Data Message Authentication
- Public Spreading Code Authentication
- Private Spreading Code Authentication

Each of these methods offer successively more secure, yet more complex methods of authenticating GNSS signals. Data message authentication consists of putting digital signatures in the data stream of the GNSS and SBAS signals. While this protects users from data forgery attacks, it provides little protections on the ranging signals themselves. The second level, public spreading code authentication, consists of "puncturing" the chipping code using cryptographic means. Receivers would store digital samples of the signal where these punctures would be present and then verify the existence of these punctures with data later sent by the GNSS satellites. The drawback to data authentication and public spreading code authentication is that they both endure an authentication delay. In other words, the user must wait for a period of time before they can despread the stored cryptographic precorrelation samples or evaluate the digital signature sent by the GNSS satellite. The third level, private spreading code authentication, uses a symmetric key scheme where users have access to the same private key as the GNSS provider. Similar to public spreading code authentication, the chipping code would be punctured, but users in this case would not have to wait for data to be broadcast from the GNSS satellites to check the authenticity of the spreading code since they already have the keys available to them. This method allows for users to instantaneously verify that the received signals came from the correct source, but since all users would have access to the private keys, special protections would need to be in place so that users themselves would never gain access to these keys. This could be done with the inclusion of tamper-resistant hardware, but at a price. This hardware can be relatively expensive and cumbersome for some applications. Proper management of secret keys to be shared among users also provides a major impediment to private spreading code authentication.

These different GNSS authentication levels are not mutually exclusive and Scott outlined how they could be simultaneously implemented on the same signal channels. An important precedent was set by his work in that all GNSS authentication designs offer backwards compatibility for users who do not wish to use the authentication data incorporated in the GNSS channels. For the most part, civil authentication designs put forward since this work have operated with backwards compatibility in mind with the exception of the Galileo Commercial Service (CS), which employs a fully encrypted spreading code service that only authorized users have access to, similar to the GPS military P(Y) or M code. These three different levels of protection still concisely classify the different authentication proposals that have come forward since Scott's publication and so they will be used to contextualize GNSS authentication proposals for the remainder of this thesis.

In 2004, the following year, two more publications came from across the Atlantic outlining potential designs for GNSS authentication. Kuhn [9] presented a data authentication design that added a secret spreading sequence below the thermal noise. Similar to Scott's proposal [8], receivers would buffer the bandwidth of the hidden markers while they are broadcast to be verified later. What was different with this implementation, however, was that these markers would be discovered by using a pseudorandom function where the seed for that function would be sent at a later time. It was recognized by both Scott and Kuhn that receivers would need to have loose-time synchronization. After the seed or key is released, anyone can generate the spreading code sequence so receivers needed to be sure of their timing with respect to the true GNSS time so that they would not accept signals that arrived after the seed had been released.

Pozzobon et al. [10] that same year published a paper that outlined the potential markets for a Galileo authentication service. It was already known that Galileo was considering pursuing encrypted services [11], but through this publication it became clear that Galileo could also be pursuing data authentication and potentially public spreading code authentication for the public signals. Namely, it was announced that the Open Service (OS) could provide Navigation Message Authentication (NMA), the Safety of Life Service (SOL) could provide NMA, and the CS and public regulated service (PRS) could provide access restriction through spreading code and data encryption. Since this time, there have been scores of publications on the designs of GNSS authentication, and while this introduction does not have the ability to go back through and examine each contribution in detail, it will focus on several publications that have made waves in the world of GNSS authentication.

A majority of the publications on GNSS authentication have been geared towards the Galileo signals and services. While there are many publications in the GPS and SBAS literature on authentication, Galileo committed early on to implementing authenticated and encrypted navigation channels for the general public. Over the years, several European groups made focused efforts on developing authentication for GNSS and, more specifically, Galileo signals. In their 2005 paper, Wullems et al. [12] suggested two candidate authentication methods for NMA: Timed Efficient Stream Loss-tolerant Authentication (TESLA) and Elliptic Curve Digital Signature Algorithm (ECDSA). These two algorithms would continue to be the main candidates looked at for NMA in the next decade and a half and they are the two algorithms that are studied in this thesis. This introduction takes a brief look at the strengths and weaknesses of these algorithms as applied to GNSS authentication and Chapter 2 takes a deeper dive into these algorithms as candidates for SBAS data authentication.

There are two forms of cryptographic algorithms in general: symmetric and asymmetric. As the name implies, symmetric cryptography involves methods where the sender and receiver both hold the same secret information. We can think of this secret information as a shared secret key. In contrast, asymmetric cryptography involves different information held by the two parties, typically a combination of a public key and a private key. Authentication is the process of verifying the validity of the data being sent as well as the origin of where that data was generated. As opposed to encryption, authentication does not obfuscate the data. It merely appends what is called a “signature” to the data so that the data can be later verified. In the context of GNSS, a simple vision of how GNSS would be authenticated would be through the use of widely disseminated public keys while the private keys were kept by the service provider. Another topic to recognize here is the method by which these public keys are securely delivered to receivers. Chapter 5 gives a detailed design and analysis of how these public keys might be delivered to receivers for use in SBAS data authentication.

The two algorithms presented in [12] provide different ways of signing GNSS data. ECDSA is a classic example of an asymmetric authentication method. In ECDSA, the service providers produce a secret key that only they know as well as a corresponding public key. The public key part of that pair is then securely distributed by the service provider. Messages sent over the air are signed using the private key and users apply the locally stored public key to verify the data as it rolls in. Some drawbacks to this algorithm for use in GNSS are that the signatures (448 bits or greater) are large relative to GNSS data rates and the computation time at the receiver to verify a signature can be an intensive task for embedded systems. ECDSA does have the advantage of being standardized as opposed to TESLA.

TESLA reduces the key size by providing asymmetry of information through time with the delayed release of symmetric keys. The TESLA algorithm produces a signature known as a Message Authentication Code (MAC). This signature may be verified by the same key that signed it. Instead of storing the keys at the receiver, the keys are released to the user after the MACs have been sent. The trust for the receiver is that when it received the MAC, it believes that no one except for the service provider had access to the key that created that MAC when that MAC was created. Two issues arise from this implementation. First is that a receiver must have loose-time synchronization with the GNSS service provider to ensure that the MAC it received could only have been created by that service provider. This poses a potential problem in implementation since GNSS is typically the trusted source of timing for most applications. The second issue is that if the keys are being sent over the same data channel as the signatures, how are the keys going to be verified? The first issue is a problem that may be solved with an independent time source [13], but luckily the second issue is resolved by the design of the TESLA protocol.

In order to verify the keys, the keys are linked together in what is known as a "keychain". This keychain forms a cryptographic link between the keys in the form of a one-way function. A one-way function is a function that has no easy method of inversion. The keychain is created by doing a recursive set of these one-way functions and the keys themselves are released in the order opposite to that in which they were originally created. In this way, when a receiver obtains a key to verify a MAC, the first step is to perform the one-way function on that key to see if they derive the previous key. If they derive the correct key and they trust this previous key, then the key has been verified and they can use the new key to verify the most recently received MAC against the data that it signed. The last key generated for the keychain (and the first to be released to the public) needs to be verified by an asymmetric scheme like ECDSA. The advantage of using a scheme like TESLA is that MAC and key combinations can be much smaller than an ECDSA signature (on the order of 110 bits or more). They also provide a computational advantage in that the generation of a MAC is generally a much easier operation than executing elliptic curve cryptographic functions. By some experimental results, an ECDSA signature can take anywhere from 8 to 15 times longer for a CPU to verify than a TESLA MAC [14]. That being said, if the TESLA key verification is not handled properly by the protocol, this verification process can be highly time-consuming.

Applying either of the above authentication algorithms to GNSS is much more than just signing GNSS data. GNSS signals have a very limited number of available bits that they can broadcast and how these designs are implemented has tradeoffs at every decision. Early on it was recognized that parameterizing the impact of the authentication solutions put forward would help designers understand the efficacy of their solutions. These would become known as Key Performance Indicators (KPIs) and would be used to help characterize the impact of the different authentication design concepts under consideration [15], [16]. These KPIs have since been built upon and used by the GNSS community to communicate the effectiveness of their GNSS authentication designs. An example KPI metric is the Time Between Authentications (TBA). How often should a signature be sent? Shorter intervals will give smaller windows for potential spoofers to try and inject bad data, but they also come at a cost in greater required bandwidth. TBA simply communicates how often a receiver can expect to be able to authenticate data. Another metric is the Authentication Error Rate (AER). What if for some reason the data that is received is incomplete? This may be due to interference, scintillation, obstruction, or other causes. These events would cause a receiver not to be able to authenticate data. The AER helps determine the expected rate at which receivers would not be able to authenticate the full set of required GNSS data. There are other KPIs that focus more towards cryptographic strength of the algorithms, their key sizes, and how long the public keys have been exposed publicly. This work also offered some suggestions as to how to improve the performance with regard to these KPIs such as cross-authentication between GNSS satellites.

The algorithms that are being considered are known to protect data, but the question remains as to whether data level authentication can also protect the ranging component of the GNSS signals. An attack was posited in [17] and later refined by Todd Humphreys in [18] that explained how an attacker could attempt to use the same incoming cryptographically secure GNSS data bits while broadcasting false ranging information. This type of attack is known as a Security Code Estimation and Replay (SCER) attack. The basic concept of the attack is that if the spoofer listens to the true GNSS signals in real time, the attacker can attempt to estimate each data bit before it is completely received. If the receiver is successful in estimating each data bit, the receiver can then simply overlay the true GNSS data on their own synthetically created PRN sequence allowing a spoofer to generate spoofing attacks in the range domain. These papers gave a sobering message. Data level authentication can protect certain information from some attacks, but it is not a panacea against all forms of spoofing.

The Radionavigation Laboratory under the leadership of Todd Humphreys gave strong contributions to the field of GNSS authentication in the early 2010s. These contributions came in the form of authentication designs geared towards the GPS CNAV signal to be offered on L1C. In 2012, Wesson et al. presented a high-level architecture for how a receiver should successfully implement authentication and try to protect itself from these types of SCER attacks [16]. Some of these receiver functions included processes to do timing consistency checks and incorporated SCER detection methods introduced in [18]. Both ECDSA and TESLA were approached as possible authentication designs for GPS L1C. Later in 2014, Kerns et al. expanded the work done in [16] and offered an ECDSA/TESLA hybrid approach [19]. This would allow for different classes of user to authenticate the data. The ECDSA and TESLA signatures would be in the same L1C data stream so that users with high confidence in their time synchronization could authenticate the data more often, while those that were less certain could use the ECDSA signatures. This work also gave more detail into how the NMA messages would be incorporated into the CNAV channel and performed a series of analyses that evaluated the efficacy of the designs.

Work was done early on looking at authentication for GNSS augmentation systems as well. In 2010, the Stanford GPS Laboratory looked into the feasibility of incorporating digital signatures on SBAS and GBAS [6], [20]. The authors of [20] conducted case studies looking into how authentication could be incorporated into WAAS and GBAS and naturally the TESLA and ECDSA algorithms played a prominent role in this analysis. Later, in 2016, Dalla Chiara et al. looked more closely at the feasibility of implementing authentication on SBAS [21]. Introduced in this work was the concept of authenticating SBAS data using the currently unused quadrature channels on the L1 and L5 frequencies. If these channels could be used, new SBAS bandwidth would be created exclusively for authentication and the pre-existing data on the in-phase channel could be authenticated more often. Dalla Chiara et al. discussed solutions for both data level authentication and public signal level authentication and performed a preliminary analysis using the KPIs discussed in [15].

The work that has been discussed so far contains a mere glance at what has been done up until this point in GNSS authentication. There are 60+ referenced literature sources that address GNSS authentication directly and scores more that look at GNSS spoofing. The literature discussed up until this point offer a window into where we are today with authentication. The work completed over the past two decades has led to authentication being implemented through several upcoming services. Notably, Galileo is working towards an Open Service Navigation Message Authentication service [22], the Air Force Research Laboratory (AFRL) will be testing GNSS authentication on the upcoming NTS-3 mission [23], and members of ICAO are working to standardize an SBAS data authentication solution among all SBAS service providers [24], which will be the focus of the remaining chapters in this thesis.

In 2016, Galileo released a first signal-in-space specification for their Open Service Navigation Message Authentication (OSNMA) design [25]. This offers a bit-level specification of the service and how the OSNMA message structure is incorporated into the Galileo I/NAV message sequence. OSNMA currently is only authenticating navigation data and the protocol used is a TESLA-based sequence with a full specification of how the MACs and keychain are generated and verified. The program is likely to be the first publicly available GNSS authentication system with signal in space testing in 2020 and FOC in the near future.

AFRL is working on an authentication design known as Chips-Message Robust Authentication (Chimera) that incorporates both public signal authentication along with data level authentication [26]. This design finds its roots all the way back in Logan Scott's original 2003 paper and offers users two authentication variants: slow and fast. For users that have access to information over third party networks, authentication information will be passed over the network allowing users to authenticate data at a faster rate. Data and signal authentication will still be available to all other users, over the air through the GPS signals, although the time to authenticate will be larger. In 2019, the AFRL released a signal specification outlining how the cryptographic markers are distributed within the spreading code and how the navigation data signatures are created [27]. The NTS-3 satellite is scheduled to be launched in 2022 where the Air Force will have the ability to test this combined signal and data authentication design.

Finally, members of the ICAO Navigation Systems Panel are working together to standardize a solution across all SBAS providers. Implementing authentication on SBAS presents a unique challenge that can leverage the experience gained from the work on core constellations but must also encompass other special considerations. SBAS is primarily a data service, and although some service providers produce a ranging signal, SCER attacks are not as applicable here. A single spoofed range coming from an SBAS satellite can be caught using integrity methods such as RAIM. TESLA and ECDSA are the algorithms that are being looked at closely for use in SBAS.

This thesis presents an end-to-end design for how authentication can be implemented on SBAS and the thesis is organized as follows. Following the introduction produced in Chapter 1, Chapter 2 presents a more detailed look into SBAS and the special considerations that must be made when designing data authentication solutions. Chapter 3 looks closer at the algorithmic mechanisms in TESLA and performs an analysis on key and keychain lengths considered secure for use in SBAS. Chapter 4 then introduces how data authentication can be implemented on SBAS without compromise to integrity. It is in this chapter that the "Authenticate-Then-Use" approach is discussed which has large implications for how data authentication can truly secure SBAS signals. Chapter 5 introduces a method by which public keys can be securely and reliably delivered to users through the SBAS channel. Chapter 6 then gives more focused treatment to SBAS receivers and how they process SBAS authentication internally followed by Chapter 7 which gives a forward looking approach to the authentication algorithms presented in this thesis and how their designs will be influenced by future advances in quantum computing. Chapter 8 then takes a look at the road ahead for this work and this field.

Chapter 2

Cryptography in the context of SBAS

2.1 Background on Cryptography

According to Merriam-Webster, cryptography is the art and science of "enciphering and deciphering messages in secret code or cipher." As was touched upon in the previous chapter, cryptography in it's most basic form can be divided into two subcategories: Encryption and Authentication. Encryption is most often what people think of when they hear the term cryptography. Encryption entails the sharing of secret information by two parties so that they might communicate between each other and keep that information secret. Typically, we think of this secret information as "keys", such that if Individual A has access to the secret key SK and Individual B also has access to that same secret key SK, they can share a message m using the basic process outlined in 2.1 [28].

$$\begin{aligned}&\text{Individual A: } m \rightarrow f(SK, m) = ct \rightarrow \text{Individual B : } ct\\&\text{Individual B: } ct \rightarrow f^{-1}(SK, ct) = m \rightarrow \text{Individual B: } m\end{aligned} \tag{2.1}$$

As seen above, Both Individuals A and B have access to the secret key SK, and they also have access to an encryption function $f()$ and its inverse $f^{-1}()$. With these tools, they are able to create a ciphertext ct and also interpret that ciphertext to recover the original message m. Encryption is used everyday as we talk to friends and family over applications like WhatsApp and do any sensitive work on the internet such as setting up banking transactions. These types of transactions have symmetry in the way that both parties require access to the same secret information as well as the encrypting/decrypting function f. For many years, it was thought that both the secret key SK and the encrypting/decrypting function f needed to be kept secret in order to successfully deliver secret messages. In the past century, a great deal of mathematical development has taken place concerning the function f, and today it is preferred that the method f be publicly known and accepted as secure [29]. In this way, your web browser and the banks server will both know how to perform f to communicate with each other, but they will need to first establish what SK is to be used.

If encryption is thought of as a symmetric process, then the second subcategory of cryptography, authentication, is most typically referred to as asymmetric. Whereas encryption requires the sharing of secret information between the sender and receiver, authentication generates asymmetry between the two, typically through the creation of a corresponding public key. Authentication is important for the purpose of digitally signing information such that individuals who receive those messages and their signatures can verify the authenticity of the messages sent as well as verify the sender of those messages. 2.2 gives an example of such a process.

$$\begin{aligned}&\text{Individual A} : m \rightarrow G(m, SK) = [m||sig] \rightarrow \text{Individual B} : [m||sig]\\&\text{Individual B} : [m||sig] \rightarrow h(m, PK) = \text{verified}\end{aligned} \tag{2.2}$$

In this case, instead of obscuring the message m as a ciphertext, the sender creates a digital signature with the function $G()$ using a secret key SK. As opposed to the encryption function used earlier, $f()$, $G()$ has no easily computable inverse. In this way, it is difficult to take the output of this function, $[m||sig]$, and find what the secret key is even if the function, $G()$ is known. When the Individual B receives the message and the signature, they can perform an operation using a public key, PK, and a function, $h()$, to confirm whether the signature verifies the data sent. This technique of authentication information is used in many different places. For instance, when you type in your web browser to navigate to your bank's website, before your browser will let you trust information coming from your bank, it will first confirm whether the signatures that are attached to the messages verify the information. We trust that the bank is the only holder of their secret key, and if we have the correct public key, we will be able to verify any signatures they send our way.

For broadcast systems like GNSS, any new feature that is implemented must be backwards compatible. For this and other reasons concerning the securing of secret keys, encryption of public signals is not on the horizon for many GNSS. Therefor many have turned to authentication as a cryptographic tool to help secure the signals broadcast from space.

Chapter 7 goes into more detail behind the functions introduced above ($f()$, $h()$, etc.) and their vulnerabilities to emerging technologies such as quantum computing.

2.2 Background on SBAS

SBAS, or Satellite-Based Augmentation Systems, broadcast augmentation signals that provide correction and integrity information to end users. Corrections for L1 SBAS signals come in the form of ionosphere, GPS ephemeris and clock corrections that end users apply to the pseudoranges they receive from GPS satellites [30]. Current day SBAS augment GPS only, but in the future different SBAS may choose to augment other constellations as well and future L5 SBAS signals will only provide corrections to ephemeris [31]. According to [32]: "Integrity is the measure of the trust that can be placed in the correctness of the information supplied by a navigation system. Integrity includes the ability of the system to provide timely warnings to users when the system should not be used for navigation." Integrity information takes several different forms in SBAS. As mentioned above, certain SBAS messages convey "Alerts" which notify users to not use certain GPS PRNs. SBAS messages also convey information to aviation users so that they can confidently bound the errors they may be seeing in individual GPS pseudoranges.

Figure 2.1 shows an overview of how the United States SBAS, the Wide Area Augmentation System (WAAS), operates. Reference stations are placed over the area of coverage of an SBAS. In the case of WAAS this includes the contiguous United States, Canada, Alaska, and parts of Mexico. These reference stations collect observations from GPS satellites at surveyed locations clear of obstructions and other harmful features. They then continuously report these observations to a master station where the observations are used to compute correction and integrity information to be broadcast to the users. Once these corrections are computed, they are sent to a ground uplink station (GUS) which then broadcasts the signals to a communications satellite which relays that same information back down to users in GNSS frequencies that are received and demodulated by SBAS-enabled receivers.

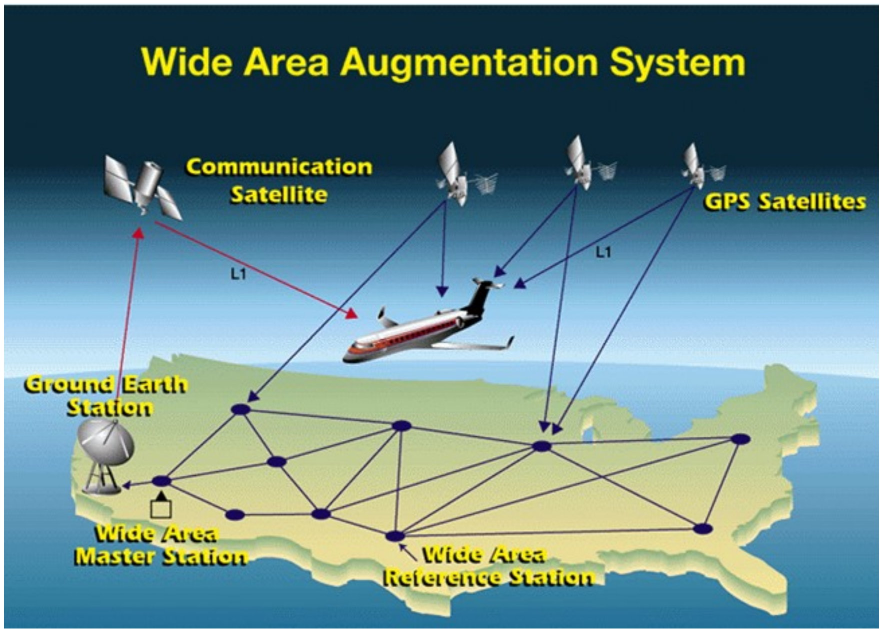


Figure 2.1: WAAS System

For more detail on how SBAS are designed and operated, the reader is pointed to [33, 30, 31]. The next section focuses on certain SBAS aspects that are pertinent to the discussion of incorporating authentication on SBAS.

2.3 Cryptography and SBAS

As mentioned above, there are many details that go into the design and operation of an SBAS. This section focuses instead on those aspects which matter most when designing authentication solutions for these channels. For the points listed below, and for the remainder of this thesis, the authentication designs introduced will be focused towards the upcoming L5 SBAS signal. Many of these designs are also applicable for SBAS L1 and when possible, analyses will be shown how the designs presented here are also compatible with L1.

The following list organizes some of these main factors:

- Broadcast System: SBAS is a broadcast system, meaning it is a unidirectional signal sent from communications satellites overhead to all users in the satellites footprint. Many cryptographic architectures that were developed for the internet require two-way communication to establish security and so these authentication methods are not applicable here.

- Backwards Compatibility: Any authentication scheme that is designed for SBAS must be backwards compatible with current signals. It is not expected that all users will utilize the authentication portion of the signal if it is offered. Many receivers that are currently in use will not be mandated to be retrofitted to accommodate any new signals.

- Aviation receivers are isolated: SBAS-enabled aviation receivers typically only have an output to other equipment and typically take no input except for power. This means that if any information needs to be sent to the receiver regarding cryptographic keys or other authentication related information, it will have to be relayed through either the SBAS signal, pre-installed by the manufacturer, or the receiver will have to be manually updated by hand.

- Long-Term Security: Currently, SBAS-enabled receivers are expected to last up to 30 years in some cases, with minimal maintenance required. As an extension to the bullet before this one, this means that not only is updating cryptographic information in the receiver difficult, but any cryptographic solution designed today will either need to be cryptographically secure for the next 30 years, or have a plan in place for a smooth replacement of cryptographic algorithms over time.

- Data Service: Unlike core constellations like GPS where the signals not only convey information but also are meant for ranging, SBAS signals are primarily used for data only. While there are some SBAS that do provide ranging, SCER style attacks introduced in Chapter 1 for core constellation data authentication services generally do not apply to SBAS data authentication services. The data is the primary information that must be protected and any irregularities discovered in the SBAS ranging signal can be caught by RAIM.

- Bandwidth Limitations: The data rate for SBAS is 250 bits/second. L1 messages contain 212-bits of data content and L5 messages contain 216-bits. Most authentication schemes are designed with much faster communication channels in mind (kbps to Mbps), which makes authentication algorithm selection for SBAS fairly limited. Not only is the data rate low, but there is also little available bandwidth left in the current message streams. If authentication messages are to be included in the message stream, care must be taken so that the nominal service will still retain its performance for users who choose not to use authentication.

- Fast Data Timeout: The goal of authentication in typical applications is to check the authenticity of the data before it is used. In the case of SBAS, that goal remains the same, but is limited by bandwidth limitations and fast data timeouts. SBAS data times out fairly quickly (12 seconds in some cases) and so this data must be authenticated quickly as well or else risk using the data before it has been authenticated. As noted in the point above, the bandwidth on these signals is already limited. This thesis presents an authentication method by which receivers are able to authenticate crucial messages before they are used, even with these bandwidth limitations. In doing so, the integrity of the Position, Velocity, and Time (PVT) solution produced by the receiver is protected against potential spoofing attacks.

- 6 second Time To Alert (TTA): In addition to fast message timeouts, there also exists a 6 second TTA requirement. This requirement states that from the point at which a satellite or satellites produces Hazardous and Misleading Information (HMI) in a receiver output, there are 6 seconds to communicate an alert to users to ensure they are no longer using this faulted information. Most of these 6 seconds are taken by the SBAS infrastructure as it discovers and communicates these hazardous scenarios. The authentication solutions presented later in this thesis preserve this 6-second time to alert for integrity information and also protect receivers from using information that could produce HMI.

- Interoperability: SBAS are designed at the outset to be interoperable. This means that a receiver which is capable of receiving SBAS signals should be able to use any SBAS service that is operational, whether it be the American WAAS, European EGNOS, or any other service. This means that incorporating authentication on SBAS signals will not just be a technical feat, but a political one as well. Any solution should be agreed upon for standardization among all SBAS and any key management mechanism that is put in place must also be agreed to be the appropriate member states. Interoperability among these systems is facilitated by voluntary participation through the International Civil Aviation Organization (ICAO) and these cooperation efforts are codified in a document known as the Standards and Recommended Practices (SARPs).

- Forwards Compatibility: In an extension of interoperability, SBAS are also designed to be forward compatible. When a new system comes online, SBAS-enabled receivers do not need to be modified to receive and use these new SBAS signals. In order to preserve this feature of SBAS, any authentication solutions would need to be forwards compatible as well. That is to say, that if a future SBAS comes online or a current SBAS decides to incorporate data authentication in the future, receivers will need a method of obtaining and using the proper cryptographic public keys used by the new service under the constraint of receiver isolation mentioned above.

- I-channel vs Q-channel: Currently, SBAS operate on an L1 I-channel (in-phase). The coming SBAS L5 channel is now being designed to function on an L5 I-channel as well, but an important question that has been asked is whether a Q-channel (quadrature-channel) should be incorporated in the broadcast as well to grant more bandwidth to providers who would use it for cryptographic authentication. In general, Q-channel authentication designs offer shorter authentication latency and higher authentication availability. While there are many arguments for a Q-channel authentication configuration, there are several considerations that must be kept in mind. When implementing a Q-channel, power must either be taken from the I-channel and given to the Q-channel or the overall power of the transmitted signal must increase. If power is to be taken from the I-channel, it must be assured that the service provided on the I-channel still meets the availability and continuity requirements promised by the service. In addition to this, if the power broadcast on the Q-channel is not sufficient, then the availability of an authenticated service may be hindered as well. If, however, the total power transmitted is increased to accommodate the new Q-channel, spectrum compatibility issues arise. SBAS signals share frequencies with the core constellations and so care must be taken such that any increase in power does not interfere with these other systems. Additionally, L5 resides in a band allocated to Aeronautical RadioNavigation Services (ARNS) such as DME and interference mitigation concerning overall power flux density from signals in space must remain under certain levels to not cause any degradation to any of these services.

The above considerations cover the larger issues at the intersection of SBAS and authentication. There are other considerations that come into account when it comes to specific issues regarding protection from certain spoofing attacks and these are addressed later in the thesis when appropriate.

2.4 The three potential solutions

As noted above, authentication solutions for SBAS need to be lightweight in terms of bandwidth while delivering enough protection that would be useful to aviation users. This section covers three potential solutions that could possibly be employed on an SBAS service: Timed Efficient Stream Loss-tolerant Algorithm (TESLA), Elliptic Curve Schnorr (EC-Schnorr), and Elliptic Curve Digital Signature Algorithm (ECDSA).

2.4.1 TESLA

Over the last several years, several papers have been published in the GNSS literature outlining different methods to facilitate data authentication [34, 19, 21].

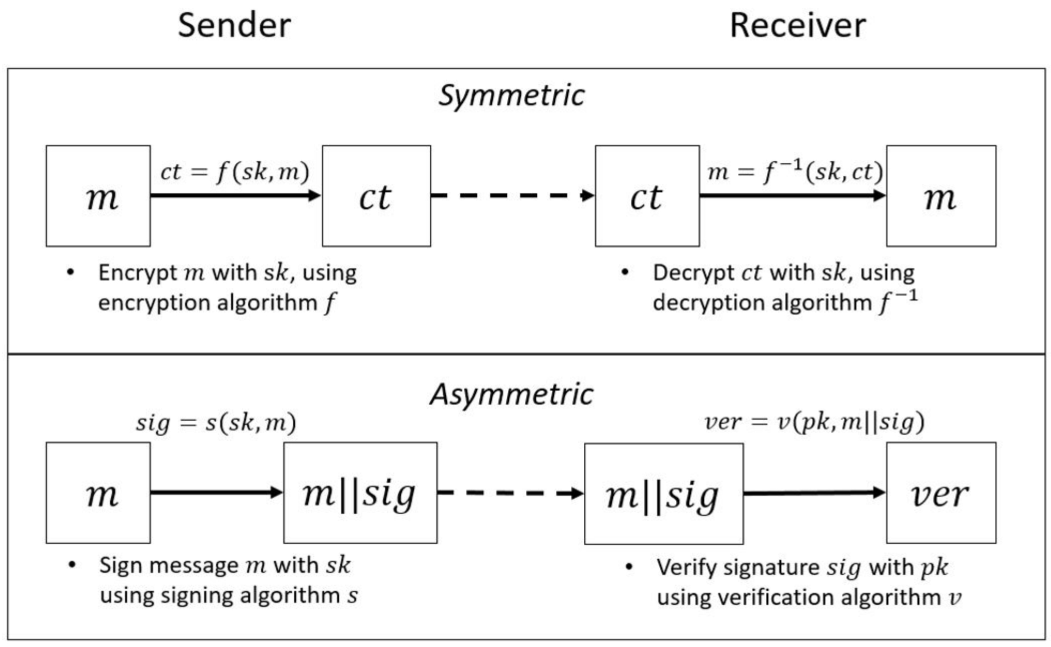

Sender
Receiver
Symmetric
m
$ct = f(sk, m)$
ct
ct
$m = f^{-1}(sk, ct)$
m
• Encrypt m with sk, using encryption algorithm f
• Decrypt ct with sk, using decryption algorithm f^{-1}
Asymmetric
$sig = s(sk, m)$
$ver = v(pk, m||sig)$
m
$m||sig$
$m||sig$
ver
• Sign message m with sk using signing algorithm s
• Verify signature sig with pk using verification algorithm v

Figure 2.2: Symmetric and asymmetric cryptography example

TESLA was developed as an integrity and authentication method for data packets in multicast and broadcast data streams [35]. TESLA uses symmetric primitives to check for authenticity, but it does so using a key release delay to create an asymmetry of information. TESLA creates a Message Authentication Code (MAC) to authenticate the GNSS data using a secret key. This secret key is also part of a "keychain," which is a sequence of keys that can be reproduced with a series of one-way functions, typically secure hash functions. Since the keychain is constructed using a one-way function and these keys are released in the order opposite to which they are created, only previous keys can be reproduced with the most recent key, and no future keys can be discerned from the keychain. Figure 2.3 shows an example keychain, and Figures 2.4 and 2.5 provide an outline for how TESLA is implemented. The details concerning the creation of the keychain, message sequence, and receiver implementation are covered later in this thesis or drawn from the literature and they are simplified here for the sake of understanding this scheme.

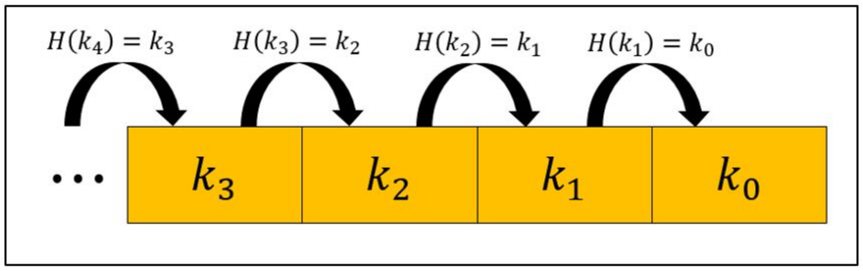


Figure 2.3: Keychain for Timed Efficient Stream Loss-tolerant Algorithm (TESLA)

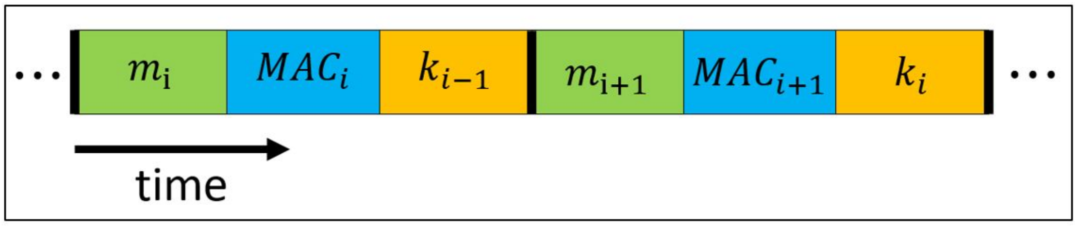
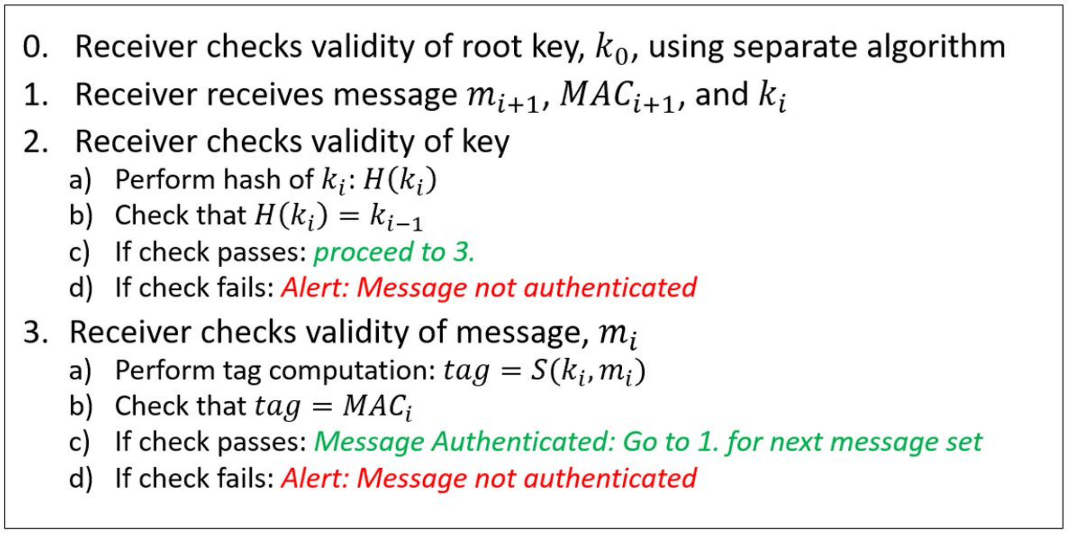


Figure 2.4: Basic authentication message sequence

The keychain of length $n + 1$ starts with key k_n, and the hash function, $h(\cdot)$, is carried out recursively n times to create the rest of the keys in the keychain. This keychain is created a priori on the ground and stored securely. SBAS satellites then transmit the messages, their MACs, and keys in ascending order, opposite from which they are created. This allows the users to use the same deterministic function, $h(\cdot)$,



0. Receiver checks validity of root key, k_0, using separate algorithm
1. Receiver receives message m_{i+1}, MAC_{i+1}, and k_i
2. Receiver checks validity of key
   a) Perform hash of k_i: $H(k_i)$
   b) Check that $H(k_i) = k_{i-1}$
   c) If check passes: *proceed to 3.*
   d) If check fails: *Alert: Message not authenticated*
3. Receiver checks validity of message, m_i
   a) Perform tag computation: $tag = S(k_i, m_i)$
   b) Check that $tag = MAC_i$
   c) If check passes: *Message Authenticated: Go to 1. for next message set*
   d) If check fails: *Alert: Message not authenticated*

Figure 2.5: Timed Efficient Stream Loss-tolerant Algorithm (TESLA) algorithm implementation outline

on the received keys to check their validity against the keys previously received. Since the function $h(\cdot)$ is assumed to be one way, the future keys in the keychain are only known by the satellites and the satellite operators. An advantage to using a keychain is that if a key is missed due to a page error or brief signal loss, $h(\cdot)$ can be executed on the next available key recursively until the keychain is recovered, allowing the receiver to regain message authentication. For the purposes of this chapter, $h(\cdot)$ will be assumed to be the hash function SHA256, but will be examined more closely in the next chapter. This hash function produces a 256-bit hash of any input, regardless of input bit length, and is accepted in the cryptographic community as cryptographically secure with a preimage security level of at most 256 bits [36]. Preimage security level refers to the order of operations required to find the hash function input given its output (e.g. security level of 256 bits takes $\mathcal{O}(2^{256})$ operations).

The signing function, $S(\cdot)$, is a deterministic function, which creates a MAC, or message tag. Both the message and MAC are sent in plaintext, and, at a later time, the key is sent in plaintext allowing the user to verify the message using $S(\cdot)$.

TESLA is not intrinsically an asymmetric algorithm based on primitives such as the discrete logarithm problem; therefore, loose time synchronization is required between the receiver and satellite in order to maintain integrity. If a receiver was to be spoofed into believing that it was a whole message step in time behind $(t_i - 1)$ the current time (t_i), the spoofer could then use the same key $(k_i - 1)$ that was transmitted to sign a different message and create a different MAC. The receiver would believe this MAC because the key is a part of the keychain $(h(k_i) = k_i - 1)$ and the message from the spoofer is authenticated by $k_i - 1$. In addition to this, if a spoofer were able to find a preimage or second preimage (an input different from the correct input that gives the same hash function output) of an element in the hashed keychain, the spoofer could then use their false keychain to authenticate different messages to provide a false position, velocity, and time solution. Mitigation of these timing attacks are treated in [13].

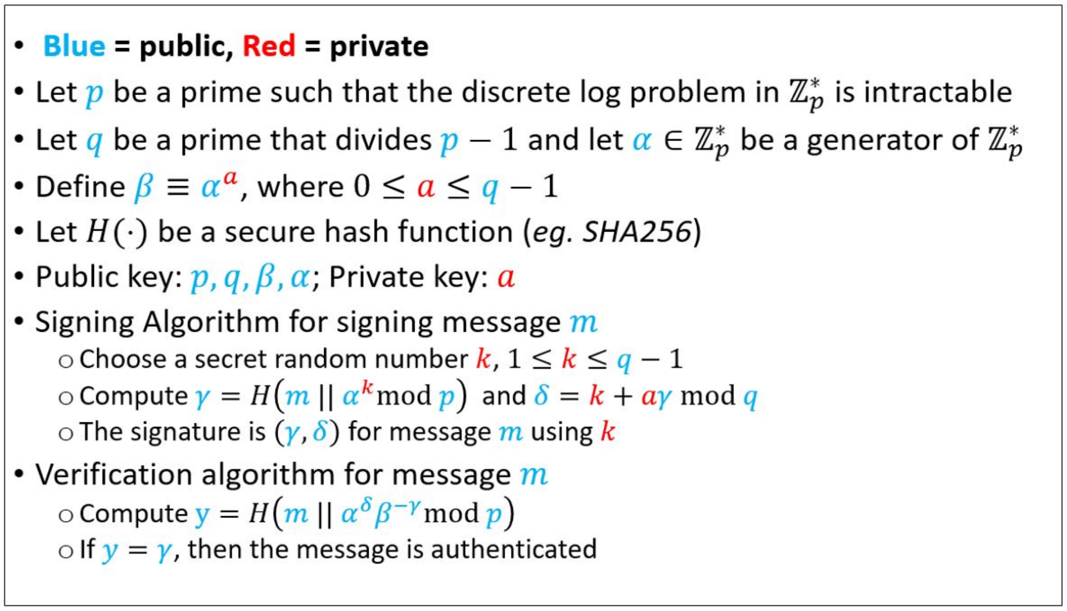
An important design consideration for TESLA is how to authenticate the root key, k_0. When a receiver first starts to use a keychain, it must first authenticate the keychain itself. Typically, this is done using a truly asymmetric scheme such as ECDSA or EC-Schnorr. EC-Schnorr is a provably secure algorithm that has 128-bit level security with as little as 384-bits transmitted in its signature [37], but it requires more computational resources relative to TESLA. TESLA offers a relatively small number of computations but requires loose time synchronization and a separate authentication scheme for the root key.

2.4.2 EC-Schnorr and ECDSA

EC-Schnorr is a variant of Schnorr that uses elliptic curves as a vehicle for the discrete logarithm problem. Elliptic curves are common throughout the world in asymmetric cryptography as their operations are thought to be harder to invert than the normal discrete logarithm (See Figure 7.2). They are, however, beyond the scope of this thesis and the reader is pointed to Stinson [28] for details of their mechanics. For the sake of simplicity, the Schnorr algorithm is presented below without the use of elliptic curves although the algorithm methodology is similar.

Figure 2.6 presents the Schnorr signing and verification algorithm for a message, m. The example examines the case where the user needs to verify the root key using

Schnorr to authenticate the TESLA keychain. Schnorr employs a private key/public key architecture to authenticate data where the private key is held only by the SBAS service provider and the public key is known to all users.



- **Blue = public, Red = private**
- Let p be a prime such that the discrete log problem in $\mathbb{Z}_p^*$ is intractable
- Let q be a prime that divides $p-1$ and let $\alpha \in \mathbb{Z}_p^*$ be a generator of $\mathbb{Z}_p^*$
- Define $\beta \equiv \alpha^a$, where $0 \leq a \leq q-1$
- Let $H(\cdot)$ be a secure hash function (*eg. SHA256*)
- Public key: p, q, β, α; Private key: a
- Signing Algorithm for signing message m
  - Choose a secret random number k, $1 \leq k \leq q-1$
  - Compute $\gamma = H\left(m \,||\, \alpha^k \bmod p\right)$ and $\delta = k + a\gamma \bmod q$
  - The signature is (γ, δ) for message m using k
- Verification algorithm for message m
  - Compute $y = H\left(m \,||\, \alpha^\delta \beta^{-\gamma} \bmod p\right)$
  - If $y = \gamma$, then the message is authenticated

Figure 2.6: Schnorr signing and verification algorithm

Since the discrete logarithm problem is assumed to be a hard problem, the algorithm is assumed to be asymmetric. Unlike TESLA, EC-Schnorr does not require loose time synchronization, but the modular exponentiations, or modular elliptic curve added in EC-Schnorr, are more computationally expensive than the MAC function used in TESLA. The security of Schnorr is derived from the hardness of the discrete logarithm problem. It can be seen in Figure 2.6 that if the discrete logarithm problem was not hard, one would only need to compute a from β and α (both publicly known parameters) to break the integrity of Schnorr. While it may seem that 384 bits is rather small for a digital signature of that strength, the data field capacity for SBAS L5 messages is 216 bits. Because two messages would need to be used to deliver an EC-Schnorr signature, this algorithm is instead intended for use in Q-channel designs to minimize the time between authentications and the impact due to message loss.

Similar to EC-Schnorr, ECDSA is truly asymmetric while having signatures that are too large to be accommodated in the I-channel. There are two main difference between EC-Schnorr and ECDSA for the purposes of this thesis. EC-Schnorr can produce signatures of length 3x of the required security level while ECDSA produces signatures of length 4x meaning EC-Schnorr has shorter signatures for the same security level. Clearly having a shorter signature is better in this case, but the drawback to using EC-Schnorr is that it is not standardized by NIST which makes it more difficult to implement securely among all users.

Chapter 3

Parameter Selection for the TESLA keychain

3.1 Introduction

As it was introduced in the previous chapter, one algorithm that has been given considerable attention is the TESLA authentication algorithm [38]. Small signature length, high security, and robustness to quantum computing attacks as will be shown in chapter 7 make it an ideal candidate for SBAS authentication. As will become apparent in later chapters, TESLA is the candidate algorithm that is best suited for authentication carried out on the In-phase channel (I-channel) of the SBAS broadcast signal. Because of this, TESLA is the main focus of most chapters in this thesis.

TESLA's security relies on a "keychain" that is derived using a one-way function, in this case a secure hash function SHA-256. The common way to denote this function is $h(x) = y$ where h is the hashing function, x is the input value and y is the output. This one-way function is pre-image resistant, meaning there is no easy way to compute $h^{-1}(y) = x$. This one-way function is also collision resistant, meaning there is no easy way to find x^* where $x^* \neq x$, such that $h(x^*) = h(x) = y$. The security of these hash functions can be compared using a metric known as the security level. This metric serves as a common reference for different cryptographic schemes and represents the effort required to break these schemes. Keys from the TESLA keychain have a limited window of time in which they are valid. This introduces a key feature when incorporating cryptography in GNSS signals: the signals are only vulnerable for only a short period of time. Whereas a low security level might be disadvantageous for information that is valid for long periods of time, low security levels for a GNSS signal may be tolerable if the window of vulnerability is similarly small.

There are several variables that can significantly alter an attackers ability to discover a keychain. The length of the keychain, L, can change the number of opportunities an attacker has to collide their keychain with the true keychain as well as change the total number of hash functions an attacker must compute to find a collision. The key length, n, can exponentially change the probability of collision. The time between authentication (TBA) and time between key release can also change the frequency of how often a new keychain must be introduced. These parameters are constrained by bandwidth of the signal and the computing power of the receivers. The best design of the TESLA keychain will take these different factors into account in order to find the pareto optimum between security, bandwidth, and user effort.

This chapter examines more closely these properties of a keychain that would be securely used for an SBAS TESLA keychain. It is divided into the following sections: Section 3.2 provides a more detailed look at the TESLA algorithm; Section 3.3 briefly introduces the probabilistic model for TESLA security from [39] and derives an updated version of the attacker model; Section 3.4 covers the trade space for the TESLA keychain design that is within the scope of this chapter; Section 3.5 provides results, conclusions and recommendations from the trade space analysis.

3.2 The TESLA Keychain

Since TESLA is a symmetric algorithm, anyone with the correct key could generate their own message and corresponding MAC in an attempt to spoof GNSS data. Because of this, the key that is released must also be authenticated and so is a part of a keychain, created by a one-way function released in the order opposite from which it was created. A keychain, shown in Figure 2.3, is generated using one-way functions recursively, shown in Figure 3.1, where the subscript denotes the index of the key and hash function used and k_0 is the root key. There must be a separate, truly asymmetric algorithm that authenticates k_0, such as ECDSA or EC-Schnorr. In this way, it is hard for an attacker to generate their own keys that will be authenticated. These keychains must be carefully designed and implemented in order to operate with a high degree of certainty that an attacker would never be capable of discovering the true keychain or any keychain that would be authenticated. In Figure 3.2, messages $m_{j=1,i} \rightarrow m_{j=J,i}$ are sent and a MAC/key combination is sent in a message afterwards. The MAC is computed using the previous messages shown in Figure 3.2 and the key, k_i. The key, k_{i-1}, is sent so the user can verify the previous MAC, $\mathrm{MAC}_{1:J,i-1}$. Figure 2.5 shows the algorithm for using TESLA to authenticate messages.

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

Figure 3.1: Example of one-way function

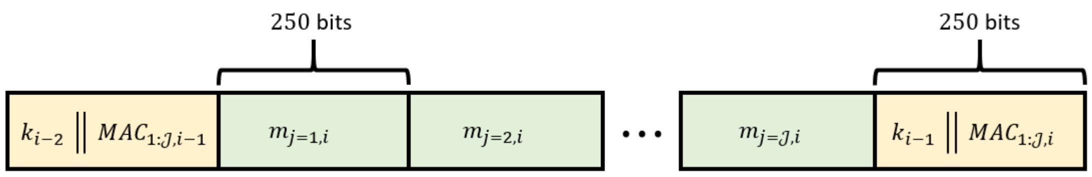


Figure 3.2: Basic authentication message sequence for an L5 SBAS signal

If SBAS messages are to be authenticated using the in-phase L5 channel, multiple messages will need to be authenticated at once as the requirements on the system precludes having every message authenticated individually. In this case there are J total messages per authentication group and in practice this number can vary from authentication to authentication. More treatment is given to this particular aspect of the TESLA design the next chapter.

The SBAS L5 message has 216 bits available in a single message. SHA256 has an output of 256 bits and so the output of the hash will need to be truncated at each step in the creation of the keychain if the key and MAC are to fit in a single message. Using only a hash function and truncation in the creation of keys provides a deterministic function for deriving the keychain. This characteristic of being deterministic offers receivers the ability to recover the keychain in the event that messages are lost, but it also allows attackers the ability to carry out pre-computation attacks. The addition of a "salt", or cryptographic randomness, to the key creation process has been proposed in previous literature [39, 19, 40]. This salt can come in the form of incorporating a time dependent set of bits input to the hash function. Because these bits change as time changes, the hash function becomes hard to precompute unless an attacker knows these salt bits. An example would simply be appending the key with its time of transmission at each key in the keychain before taking the hash and deriving the next key. In order to enable key recovery in the event that a receiver missed messages, these salted hash functions need to be deterministic, which is possible for message structures within GPS or Galileo but may be a challenge for SBAS unless the SBAS service provider adopts a rigid message scheduler. An alternative would be to include a fixed set of bits that are constant with respect to each keychain but are randomly chosen between different keychains so a salt for a future keychain is unpredictable. This salt is appended to each key before it is hashed to find the previous key and the salt would be broadcast and authenticated alongside the root key so that receivers could utilize it once the keychain is in use. If the salt is reasonably large, precomputation attacks on the keychain would be infeasible. Both of these salt techniques are not mutually exclusive and can both be implemented. Further discussion on this salt will be given in Section 3.4.

There are several methods by which an attacker might attempt to circumvent or break an authentication scheme such as TESLA, but the mode of attack this chapter will address is an attack on the TESLA keychain itself. With enough resources, an attacker may try to discover the keychain with a brute force attack by guessing final keys and performing the necessary one-way to discover a valid keychain.

3.3 Probabilistic Attack Model for TESLA Security

A standard L5 SBAS message contains 250 bits: a 4-bit preamble, a 6-bit message type identifier, a 216-bit data field, and a 24-bit CRC. The one-way function that is proposed for use in the creation of the TESLA keychain is the SHA256 algorithm. This Secure Hash Algorithm (SHA) acts as a standard one-way function in cryptography that is capable of taking an arbitrarily sized bit-field and creating a 256-bit output. This function is deterministic and has no known inverse, making it an ideal candidate for computing the TESLA keychain. The keys in the keychain will be derived from this one-way function by truncating the output of the hash to create each key. In [39] it was pointed out that one needs to be careful when truncating the output of the SHA256 function to create the keychain. A pre-image or second pre-image becomes more likely to be found as the amount of truncation increases.

The attack is set up as follows: A keychain is created in secret by the SBAS command segment and stored in secret. An attacker has a window of attack where they plan to broadcast a spoofed SBAS signal and change the time or position of a user. The attacker wishes to discover l keys between k_i and k_{i+l} in order to forge messages and have the user accept these messages as valid. Figure 3.3 shows this window of attack within the keychain.

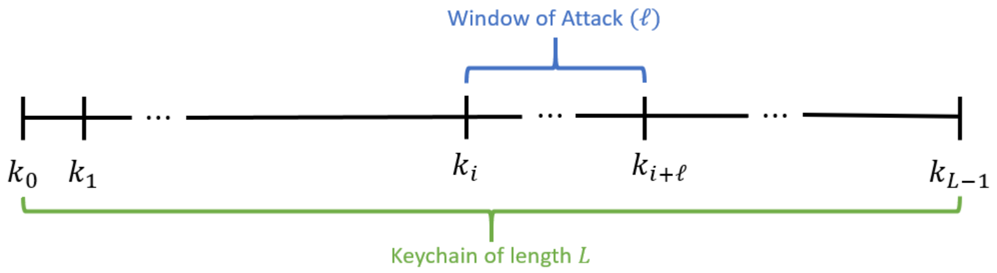


Figure 3.3: Window of attack desired by attacker

In this chapter, k_0 represents the root key and L represents the length of the full keychain. The attacker would guess a value $\hat{k}_{i+l}$, recursively hash and truncate this value l times and check if $k_i = \hat{k}_i$. If the output is true, the attacker has found a keychain that a receiver would accept and begins to spoof, otherwise the attacker guesses another $\hat{k}_{i+l}$ and tries again. If the attacker begins this method before the release of k_i, computed $[\hat{k}_i, \hat{k}_{i+l}]$ pairs are stored, and a search is performed once k_i is released. Let T_k represent the period of time between message authentications, which will be roughly constant in practice and will be treated as constant here. If a salt is added, preventing the attacker from carrying out a precomputation attack, then the time available for the attacker to find a valid keychain is given in Equation 3.1.

$$T = T_k(i + l) \tag{3.1}$$

A probabilistic model was derived in [39] to assess the security of the keychain and a metric known as the probability of a successful attack, P_s, was computed. The probability of success is given in Equation 3.2 for the case where the keys are generated using a combination of hashes, truncations, and padding.

$$P_s = \left(1 + \frac{1}{l}\right)\left(\frac{R_h T}{N}\right) \tag{3.2}$$

In this equation, R_h is the computing power of the attacker expressed in terms of hash/s and N is the total number of possible permutations for the set of keys; if keys are n-bits then $N = 2^n$. For this equation and those presented in the rest of this chapter, it is assumed that $L << N$. And again from [39], the probability of success against an ideal keychain, where the key generation function is a true one-to-one function without collisions, is given in Equation 3.3.

$$P_s = \left(\frac{1}{l}\right)\left(\frac{R_h T}{N}\right) \tag{3.3}$$

As l increases, the probability of a successful attack for the non-ideal method becomes dependent upon the hardware and time available to the attacker and the effect of the length of the keychain diminishes with respect to the ideal key creation method. It was shown [39] that for the assumed case where the keys were truncated to 80 bits, a brute force attack using modern computing hardware on the keychain yielded a 10^{-4} probability of success. Since then, longer keys have been proposed to be used and this thesis will give recommendations to the minimum key length to be used for the TESLA keychain.

The attacker model that was developed in [39] consisted of a malicious actor targeting a specific set of keys within their window of attack. The attacker would know they were successful if their computed keychain reached the same key, k_i, by the time that key was released. The attack occurred under the assumption that a time-varying hash function was used in the creation of the keychain, so an attacker could not carry out a precomputation attack on the keychain. If the attacker is not particular about when they would like to attack, the limiting case for the window of attack is the entire keychain ($l = L$), shown in Figure 3.4. In this case, using the above attack model, the attacker would hash each guess of the final key L times until a guessed root key was calculated. Once the keychain begins to be used, an attacker would only have to hash down to the most recently released key to check if their computed keychain is valid, thereby minimizing the total number of hash functions that they would have to perform. This particular strategy will change the probability of success metric and that change is derived as follows.

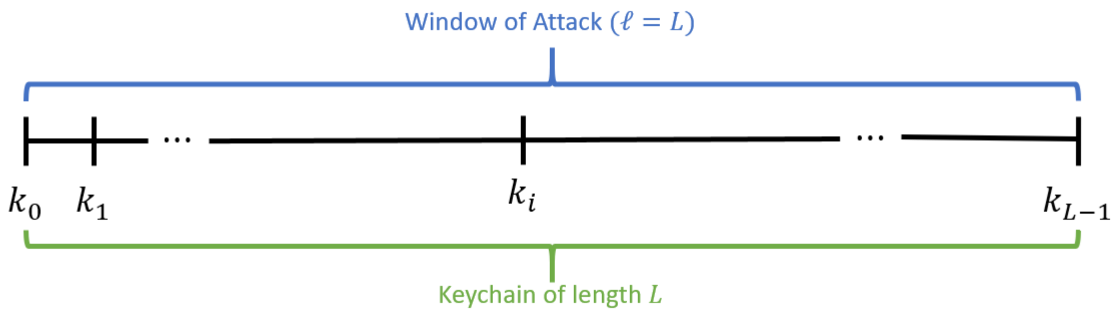


Figure 3.4: Attack on an entire keychain

From [39], the definition of a successful single guess using a brute force attack on the keychain is given as Equation 3.4, where S_j is the condition of success for guess j, i is the index of the disclosed key along the keychain (which may or may not be the root key), l is the length of the keychain the attacker wants to spoof, $\hat{k}_i$ is the guessed key at the end of the keychain computed by the attacker, and k_i is the disclosed key.

$$S_j(i,l) = \left\{ \hat{k}_i^j = k_i \right\} \tag{3.4}$$

For the case of an attack on the entire keychain mentioned above, the success criteria for a single guess is rewritten as Equation 3.5.

$$S_j(0,L) = \left\{ \hat{k}_0^j = k_0 \right\} \tag{3.5}$$

If Equation 3.5 defines the condition of success for a single guess, then Equation 3.6 defines the condition for success for an entire attack, where N_A is the set of all permutations within in the computational scope of the attacker constrained by computing power and time.

$$\mathbb{S}\left(0, L, N_A\right) = \bigcup_{i=1}^{N_A} S_j(0,L) \tag{3.6}$$

The probability of success defined in Equation 3.2 is predicated on the notion that the attacker will perform L key generations (padding $\rightarrow$ hash $\rightarrow$ truncation) at each guess j for an attack on the full keychain. As the attacker is attempting to brute force attack the keychain, however, keys are continuing to be released. It is not necessarily in the interest of the attacker to find a keychain that arrives at the root key, but to find a keychain that arrives at the most recently released key. If an attacker finds a keychain that arrives at the root key but arrives there through a second pre-image that occurs farther down in the keychain than the most recently released key, the attack is not a successful attack as the receiver would not authenticate this false keychain if they had any recently saved keys in memory. Any key generation computations past the most recently released key is also a waste of time and computing resources. For long keychains, it would be unnecessary to compute large sections of the keychain if they have already been released. Therefore, an attacker would likely listen to the SBAS service and continually update the target key, k_i, to match the most recently released key.

As noted in [39], if the assumption holds that $P[S_j(0,L)] << 1/N_A$, the attack is well approximated by its union bound:

$$P_s = P\left[\mathbb{S}\left(0, L, N_A\right)\right] = \sum_{j=1}^{N_A} P\left[S_j(0, L)\right] \tag{3.7}$$

Where the set of attacker permutations is $N_A = R_h T/L$. For a model where the attacker computes no keys past the most recent key, the sum in Equation 3.7 is split into a sum of sums shown in Equation 3.8.

$$P_s = \sum_{m=0}^{L-1} \sum_{j=1}^{N_m} P\left[S_j(m, L-m)\right] \tag{3.8}$$

In Equation 3.8, N_m is the set of all guesses that can be computed by an attacker in between key releases and is now a function of the number of keys that have been released since the start of the attack. In this case, $N_m = \frac{R_h T_k}{L-m}$, where T_k is the interval in seconds between key release. As time moves forward and m increases, N_m increases linearly as one would expect when the length of the keychain that must be computed decreases. Since $\frac{1}{N_m} > \frac{1}{N_A}$, the probability of success is still approximated well by its union bound. The average single attempt success probability derived in [39] is given in Equation 3.9, and remains true for this new model.

$$P\left[S_j(0, L)\right] = \frac{L+1}{N} \tag{3.9}$$

The sum of Equation 3.8 is thus presented as

$$P_s = \frac{L+1}{N}\left(\frac{R_h T_k}{L}\right) + \cdots + \frac{L-m+1}{N}\left(\frac{R_h T_k}{L-m}\right) + \cdots + \frac{2}{N}\left(\frac{R_h T_k}{1}\right)$$

$$P_s = \left(\left[\sum_{m=0}^{L-1} \frac{1}{L-m}\right] + L\right)\left(\frac{R_h T_k}{N}\right) \tag{3.10}$$

Where the sum inside of Equation 3.10 is the L-th harmonic number. There is no closed form solution for the harmonic numbers, but for large values of L, they are well approximated by Equation 3.11 where γ is the Euler-Mascheroni constant.

$$H_L = \sum_{m=0}^{L-1} \frac{1}{L-m} \approx \ln L + \gamma + \frac{1}{2L} - \frac{1}{12L^2} + \frac{1}{120L^4} \tag{3.11}$$

With this relation, Equation 3.10 can be rewritten in the familiar form seen in Equation 3.12 where one notices that the time available to attack the keychain is $T \leq LT_k$ assuming the attack begins when the root key and salt are released.

$$P_s = \left(1 + \frac{H_L}{L}\right)\left(\frac{R_h T}{N}\right) \tag{3.12}$$

As L increases, H_L increases logarithmically, and so probability of success converges to the solution given in Equation 3.2. As a formal exercise, the result found in Equation 3.12 shows that even with this updated attack model, the benefit to the attacker is linear and not exponential.

The equation for the probability of a successful attack given thus far assumes the use of a time-varying hash function or a salt that serves to add entropy to the key generation process and prevent pre-computation attacks. In the case where there is no keychain salt and a time-independent hash function is used in the generation of the keychain, attackers can use precomputation attacks to discover the keychain long before the keychain is released. The probability of success is then dependent upon the attacker's ability to find a collision on the keychain which is derived in [39] and given as

$$E\left[p_c\right] \cong \frac{R_h T}{N} \tag{3.13}$$

Where the attack time T in Equation 3.13 is no longer bounded by the length of the keychain as it has been previously. If an attacker is given years to attack the keychain, the probability of computing a valid keychain increases linearly with precomputation time.

3.4 TESLA Trade Space

There are several variables that are malleable in the design of the TESLA keychain. Each variable is constrained by limits in bandwidth, security, authentication error rate (AER), receiver effort and other factors crucial to the purpose of SBAS signals. It is important to understand how each of these variables effect these performance parameters and so an analysis is carried out using the mathematical models derived in this and previous works. This work will focus primarily on the affects these variables have on the security of the system.

The variables that are considered in this work are thought to have the greatest impact on these performance parameters. The number of keys in a keychain (L) can affect the security of the TESLA scheme along with the length of the keys (n). Time between authentication (TBA), represented in this chapter by the variable T_k, impacts the bandwidth the authentication messages consume in the L5 SBAS signal, affects the AER, and the effort the receiver must exert periodically when authenticating the message. TBA also has implications in the security of the keychain if L is chosen independently. The use of a salt in order to prevent precomputation attacks will affect the security of the TESLA keychain as well.

A model for the computational abilities for an attacker is derived using hardware that is specialized in the mining of cryptocurrencies. One of the most productive cryptocurrency mining facility exists in Ordos, China that uses 25,000 machines 24 hours a day [41]. Some of the best machines that are available today run are capable of running at speeds of up to 1.4×10^{13} hash/sec [42]. A generous model for the computational ability of an attacker can be estimated at $R_h = 25{,}000(1.4 \times 10^{13})$ hash/sec.

Proper selection of the keychain length and key length should be chosen to make the probability of a successful attack negligible. This threshold is set to 10^{-9} for this work and only parameters that render the possibility of a successful keychain attack to this value or lower will be considered. First, the case where a salt is added to the keychain, thereby rendering precomputation attacks infeasible, is considered. Using a TBA of 6 seconds, Figure 3.5 shows a contour plot of the probabilities of a successful brute force attack on the TESLA keychain with various combinations of key length and keychain length. It is clear from the plot that there is a logarithmic relationship between the keychain length and the probability of a successful attack. Figure 3.6 uses the same parameters and plots the keychain length logarithmic in time.

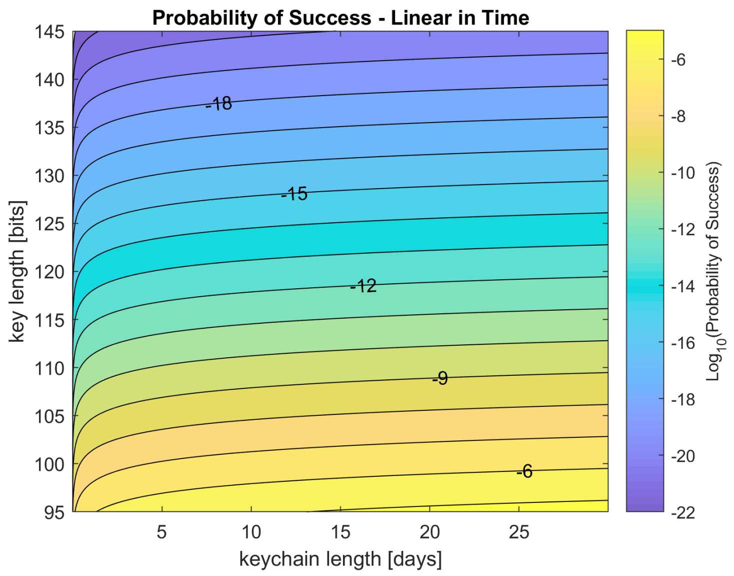


Figure 3.5: Probability of Success with keychain length linear in time. TBA = 6s.

The data field within the L5 SBAS message is 216 bits long, which will be sufficient to fit the key, MAC and extra bits carrying information about the root key and salt. An attack on forging the MAC itself by guess has a one-time opportunity for success. If successive MACs do not authenticate the data, the receiver will be alerted and will not trust the associated SBAS data. The probability of guessing the correct MAC given in Equation 3.14 where m is the number of bits allocated to the MAC. For the probability of a successful single MAC forgery to be $P_{\text{MAC}} \leq 10^{-9}$, the minimum MAC length needs to be is $m = 30$ bits. For a 216-bit data field with a 30-bit MAC, 186 bits remain for the key and other information. For the sake of this chapter we'll assume a 30-bit MAC and the following chapter will focus more on the MAC design for use in SBAS.

$$P_{MAC} = 2^{-m} \tag{3.14}$$

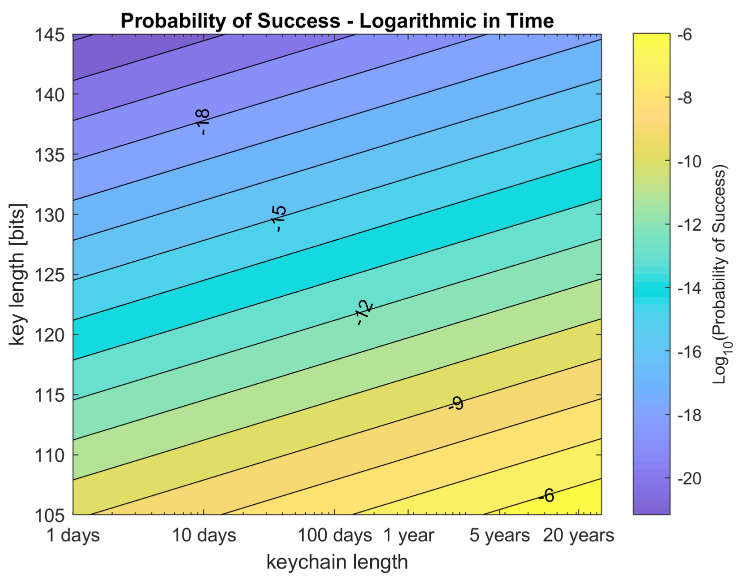


Figure 3.6: Probability of Success with keychain length logarithmic in time. TBA = 6s.

From Figure 3.5 and Figure 3.6, a key length of 115 bits would be sufficiently secure for the assumed TBA and R_h for keychains not lasting more than several years. Smaller key lengths are possible for shorter keychain lengths, but as these smaller key lengths are still at most 105 bits, 115 bits is used for the remainder of this thesis, the extra bits affording exponentially more security than the smallest possible key length. This would leave 71 bits in the L5 message for extra information that could be used for over-the-air rekeying (OTAR), salt information and root key information.

The above figures assume there is salt added to the keychain preventing precomputation attacks. Figure 3.7 below shows the equivalent plot of Figure 3.6, assuming the attacker performs a precomputation attack lasting 30 years with a TBA of 6 seconds. The probability of success is only a function of the keychain length for very long keychains because the attacker is able to precompute a keychain, and in this case a 115-bit key leads to a probability of success slightly under 10^{-8}.

The effects of the chosen TBA and assumed computing power (R_h) of the adversary are shown in Figure 3.8 and Figure 3.9, respectively. In Figure 3.8, a TBA = 1s

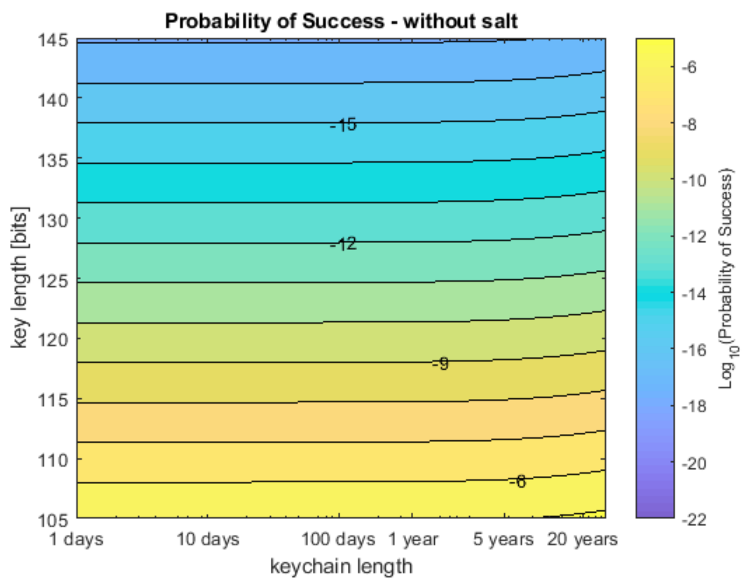


Figure 3.7: Probability of Success with no salt added to the keychain. TBA = 6s and 30 years of pre-computation.

refers to a key/MAC message existing on the quadrature channel and in Figure 3.9, an $R_h = 350{,}000$ Thash/s is equivalent to the attackers computing capabilities assumed throughout the rest of the chapter.

These plots show iso-lines for $P_s = 10^{-9}$ for varying key and keychain lengths as TBA and R_h are changed. Due to the exponential effect the key length has, for fixed keychain lengths both the TBA and R_h change the key length by several bits. In this way, the risk of being too conservative in the key length has a limited effect on the bandwidth of the L5 SBAS message.

3.5 Conclusions on TESLA Keychain Security

A model for the security of the TESLA keychain that was derived in an earlier paper [39] was modified to accommodate a new form of attack. A security-based trade analysis was carried out to look at the role different variables played in the effectiveness of the TESLA keychain. The data field for an L5 SBAS message is 216 bits that will include a MAC, key, and other authentication information, bandwidth

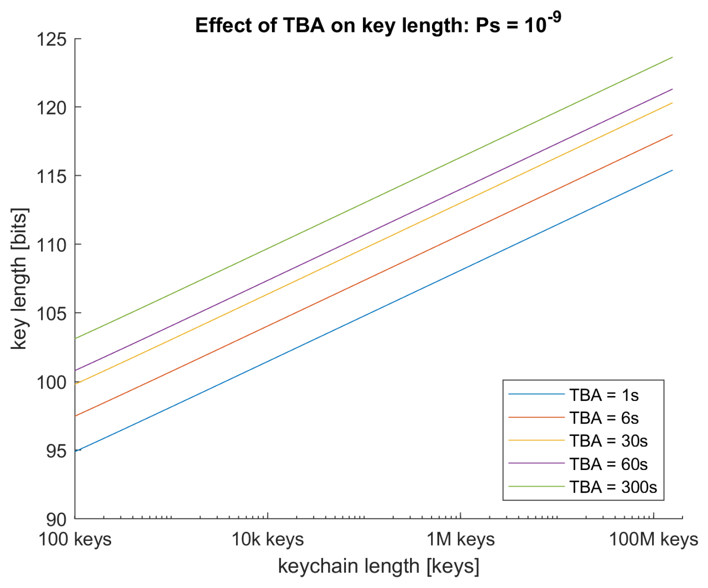


Figure 3.8: Effect of TBA on requisite key length for $P_s = 10^{-9}$

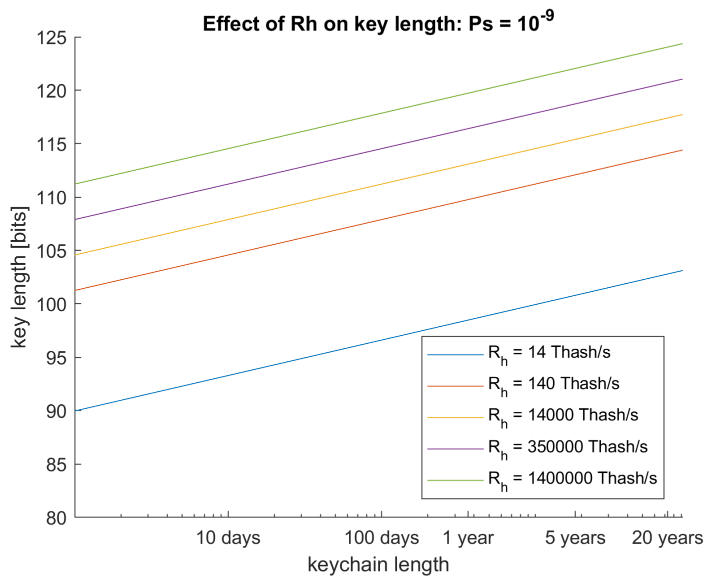


Figure 3.9: Effect of R_h on requisite key length for $P_s = 10^{-9}$. $R_h = 350{,}000$ Thash/s is equivalent to the computational ability assumed throughout the chapter.

permitting.

Contour plots where created looking at the effect that key and keychain length combinations had on the security of the keychain. The security of the keychain scales exponentially with linear increases in the key length. The addition of salt proved to be a robust method to prevent precomputation attacks on the keychain. If a 30-bit salt is used, an attacker would have a $P_s = 10^{-9}$ of correctly guessing a future salt, similar to the length of the MAC.

To look at the consequence that time between authentication and computational ability played on the security of the keychain, plots containing iso-lines were created to illustrate the magnitude of the change each variable had on the security. In both cases, large changes in the independent variable only resulted in small changes in the required key lengths, giving confidence that if a system is designed with sufficient margin in the key length, the security of the keychain increases exponentially.

From this work, for keychains lasting less that 1 year in duration, 115-bit keys appear to be sufficient for security against an attacker with the above assumed capabilities.

Chapter 4

SBAS Authentication: Integrity and Security

4.1 Introduction

Authenticating the data streaming from SBAS satellites has the ability to protect SBAS users from most malicious SBAS targeted attacks. Due to the limited bandwidth available in the SBAS broadcast, most schemes used for internet protocol (IP) authentication are not suitable. There have been a number of papers over the years looking into various adaptions of cryptographic schemes that can be used to authenticate GNSS signals [20, 43, 44, 16, 45]. This research has guided the authentication scheme search to three main candidates: The Timed Efficient Stream Loss-Tolerant Algorithm (TESLA) [38], Elliptic Curve Digital Signature Algorithm (ECDSA) [46], and Elliptic Curve Schnorr (EC-Schnorr) [47], as introduced in chapter 2. There are other schemes that are being reviewed, namely post-quantum cryptographic schemes that are introduced in chapter 8, that will be looked at more closely in future designs for GNSS authentication. In addition to research into schemes, there has also been work done to implement a key management structure that would allow these SBAS users to update important cryptographic key information using messages broadcast by the SBAS satellites themselves [48]. This work into the key management structure will be covered in detail in the following chapter.

The research field is getting closer to the point where actual operational decisions can be made concerning whether these authentication methods should be implemented. This chapter introduces a method of authentication that minimizes authentication impact to SBAS users. In order to carry out an evaluation of the impact to users, a concept of operations (CONOPS) is defined. This thesis uses the term CONOPS as a catch-all term for receiver operations related to authentication procedures. Listed here are just a couple examples of the CONOPS questions that must be answered:

- What should a receiver do with information that hasn't been authenticated yet?
- What should a receiver do with information that can't be authenticated, due to a corrupted message or signature?
- How should a receiver react in the event that an authentication has failed, i.e. the data received does not match the signature that was received?

All these questions and more must be addressed before a nominal impact to users can be assessed. This chapter provides some answers to these questions and gives justification for the CONOPS presented here. The chapter is split into the following sections: Following this introduction, a short section dedicated to the description of the schemes employed is given. Then, the Key Performance Indicators (KPI) used to measure the impact of these schemes along with a discussion on CONOPS is given that builds the framework around how a receiver should handle the authentication service. Following this discussion is a brief introduction to the receiver simulation methodology employed and then an impact analysis is carried out. The chapter concludes that careful consideration of CONOPs and authentication design can yield an authentication solution that preserves integrity against spoofing threats and message loss.

4.2 Authentication Scheme Details

This section gives details of the schemes introduced in the previous chapters. Two variants of TESLA are introduced here along with an ECDSA/EC-Schnorr design. For the purposes of this chapter, the ECDSA design is equivalent to the EC-Schnorr design and so only ECDSA is referred to. The ECDSA variant includes a parallel message stream in the Quadrature channel (Q-channel) of L5. A more detailed description of the implementation of ECDSA is given in [48]. Figure 4.1 shows an example message sequence where s[] in the Q-channel represents a signature of the corresponding messages in the I-channel and OTAR stands for Over-The-Air-Rekeying bits that deliver key management information through the SBAS data stream. Figure 4.2 depicts the contents of the signature message for the Q-channel. The data field for an L5 message is 216 bits. With a 448-bit signature, ECDSA obtains a security level of 112-bits and in this case is strung across two 250-bit message fields to create one Q-channel message. The remaining 52 bits are used for key management.

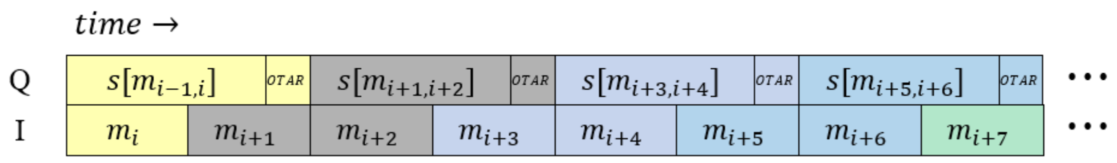


Figure 4.1: Q-Channel ECDSA Implementation

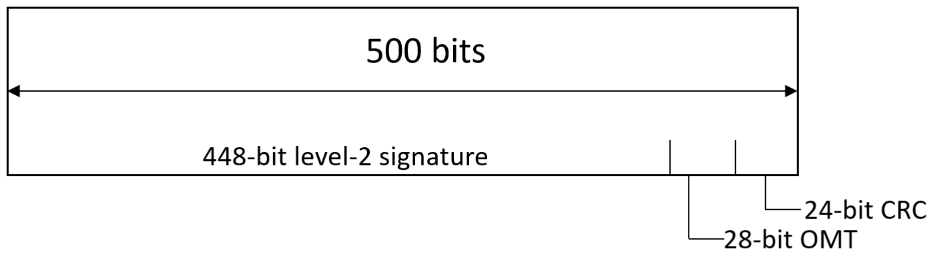


Figure 4.2: ECDSA Authentication Message Contents

It is desirable that the I-channel implementation signature content be contained within one message. To accomplish this, the signature must be less than 216 bits and still retain a high enough security level. In this case TESLA is found to be a viable candidate as the combination of the message authentication code (MAC) and the TESLA key can fit within these bounds.

An example message structure of an I-channel authentication message is shown in Figure 4.3. The typical implementation of TESLA creates a single MAC for a set of previous messages. The bandwidth of the I-channel is crowded as it is, however, and so an inclusion of an authentication message limits the frequency at which these messages can be sent. In this design, a nominal Time Between Authentication (TBA) is designed to be 6 seconds. This implies that in order to properly carry out the TESLA signature protocol for a set of messages, all messages must have been received correctly. The converse of this means that if a single message in the set is not demodulated correctly, then all messages that were a part of that authentication group can no longer be authenticated.

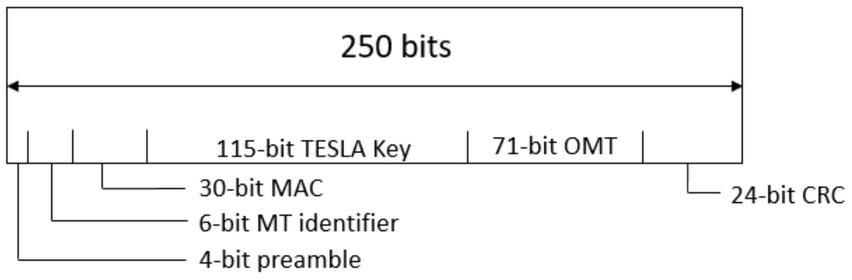


Figure 4.3: TESLA - BigMAC Authentication Message Contents

In order to mitigate this vulnerability, a new design is put forward. Instead of signing all previous messages in a set with a single MAC, individual MACs are instead delivered, and all signed using the same key. An example authentication message for this implementation is given in Figure 4.4. From here on, the original implementation shown in Figure 4.3 is referred to as TESLA-BigMAC and the authentication message depicted in Figure 4.4 is referred to as TESLA-LittleMACs. The 30-bit MAC for the TESLA-BigMAC case was designed to limit the probability of any successful guess of the MAC to less than 10^{-9} as we can recall from the previous chapter. In the case of TESLA-LittleMACs, this probability is $\sim 3 \times 10^{-5}$ for each 15-bit MAC, but as described later in the CONOPS section of this chapter, the security of the scheme is retained through the approach to a failure to authenticate scenario. The clear advantage of the TESLA-LittleMACs design is that it mitigates the effects of a single missed message on a batch of messages intending to be authenticated.

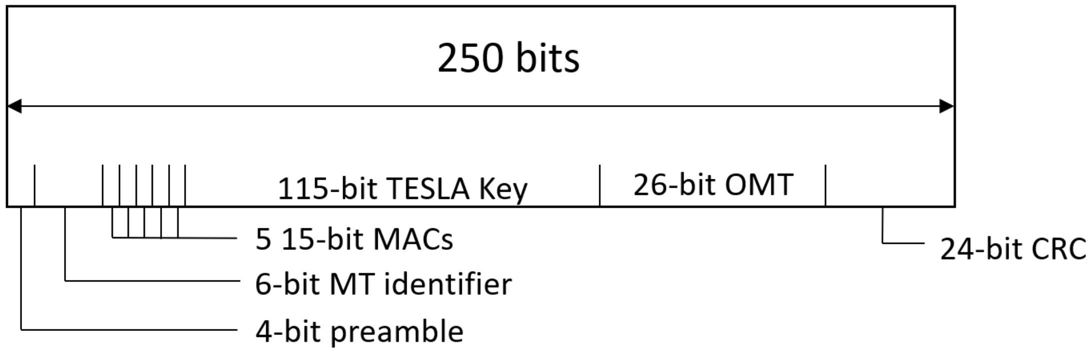


Figure 4.4: TESLA - LittleMACs Authentication Message Contents

4.3 Key Performance Indicators and Definitions

There are three possible outputs from an authentication attempt. The first, defined as "Authentication Passed", occurs when all messages have been received correctly by the receiver (all CRCs pass), and the signature algorithm output on these messages is true. The second output is defined as "Authentication Failed". In this case, all messages have been demodulated correctly, and the signature algorithm output is false. This indicates that either an error has occurred at the service provider level or that the receiver is receiving unauthorized broadcasts of SBAS data. The final output that can occur from a signature algorithm is defined as "Authentication Unavailable". In this case, at least one of the messages to be authenticated or the authentication message itself has been demodulated incorrectly (CRC does not match), which may render the receiver unable to authenticate other information surrounding the missed messages. The CONOPS plays an important role in how a receiver deals with authentication outputs, as will be seen in later sections.

During the development of authentication implementations for SBAS, several key performance indicators (KPIs) have been established that quantify the impact to users. There are many that are associated with the implementation of any scheme, but this chapter highlights a few that are used to quantify the impact of the schemes set forth here. All KPIs referenced throughout the thesis are summarized in Appendix A.

The first two KPIs are already defined in the context of non-authenticated SBAS operations. The first is Availability, which denotes the probability that the SBAS service will be available to the user at a given time. The second is Continuity, which is defined as the probability that the service will remain available during a phase of operation (given to be 15 seconds here), given that the service was initially available at the beginning of said phase.

Three more KPIs are defined that are more specific to the products of authentication. The Authentication Error Rate (AER) is defined as the rate at which authentications fail or are unavailable. The Time Between Authentications (TBA) is defined as the duration of time between authentications. Nominally this is a constant value, but in practice it may vary when authentication messages cannot be processed by the receiver due to incorrect demodulation of the incoming data or alert scenarios that may upset the normal cadence of SBAS messages. The final KPI that we will examine pertaining to authentication is the Authentication Latency (AL) and in this particular case, the median AL. In aggregate, this simply looks at the median time that it takes to authenticate messages once they have been received. These KPIs, along with the defined receiver outputs, now allow us to dive into the CONOPS design.

4.4 Concept of Operations (CONOPS)

A CONOPS sets forth how a receiver reacts in all possible scenarios. The goal of implementing an authentication service to SBAS is to minimize any impact to users while delivering a secure service. The CONOPS is where compromise between these often-competing goals is found. This section is organized as a series of operational considerations posed as questions which are then addressed in the paragraphs that follow.

How does a receiver treat unauthenticated data? Does a receiver use data that has not yet been authenticated?

SBAS users rely on the service to provide vital information that must be protected from potential hampering. If a receiver uses data before it has been authenticated and that data has been spoofed, it may be exposing itself to misleading information before it can verify the data's authenticity. For these reasons, with the CONOPs presented here, receivers do not use most information that is received before that information has been authenticated. All messages received from the SBAS GEOs have a period of validity. In the most stringent operations, receivers cannot miss two of the same message types in succession. For the most critical information, such as the (Dual Frequency Range Errors) DFREs delivered in the MT35 and MT32 messages, the time of validity is 12 seconds [31]. The Q-channel scheme introduced here has a nominal TBA of 2 seconds, while the I-channel schemes have a nominal TBA 6 seconds. With this in mind, receivers can wait to apply the messages received from the SBAS satellites until that information has been authenticated. There are other proposals that would allow users to use unauthenticated and potentially spoofed SBAS data before it can be verified. This creates too much risk for receivers if the authentication latency is large and so is not supported as a viable concept in this thesis.

One caveat to this is that SBAS systems must meet time to alert (TTA) requirements. These alerts are delivered to users through an increase in (DFRE Indicators) DFREIs associated with faulted measurements. In order to meet the TTA requirements, our recommended CONOPS stipulates that users immediately incorporate increases to DFRE information when that information is available, but not incorporate decreases to DFREs until that information has been authenticated. Data that cannot be authenticated due to "Authentication Unavailable" events are never incorporated in the integrity estimation except for the aforementioned increases to DFREIs.

How does a receiver react to an "Authentication Failed" scenario?

In the event that an authentication has failed, either as a failure in the data authentication or key management delivery, the receiver makes two different decisions depending on the authentication scheme employed. In the case of Q-channel ECDSA and I-channel TESLA-BigMAC, the receiver reverts using data from another GEO as long as that data remains authenticated and only returns to the original GEO once it has been established that the service is authenticated once again. In the case of I-channel TESLA-LittleMACs, all data used from the GEO is cleared, the receiver reverts to using data from another GEO if it is available and authenticated, and only returns to the original GEO once the authentication service is available once again.

This is a key feature of the TESLA-LittleMACs implementation that allows for smaller, distributed MACs. If any, of the five MACs delivered in a single authentication message outputs as a failure, all data from that GEO is purged from the receiver. If an attacker were to attempt to forge messages in this case, they have a higher chance of successfully replacing a single message than in the TESLA-BigMAC case, but if the attacker wished to spoof multiple successive messages, the probability of detection of such an attack increases dramatically. In this way, TESLA-LittleMACs can mitigate the impact to other messages when CRCs don't pass for a single message while still delivering a secure authentication service. In all cases, if another GEO cannot be authenticated, then the service is no longer available, and the user must resort to other means of navigation until an authenticated service is available once again.

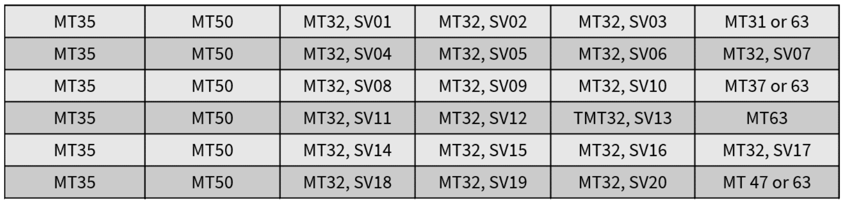
What is the impact to the service if a message times out and is no longer available?

Different messages within the L5 broadcast stream have different impacts to user performance. In the case of MT35, which carries the crucial DFRE information, if that message is not available, then the service is no longer available. In the case of MT32 which sends corrections specific to each satellite, if the message is not available, then the satellite corrected by that message is not available. Whether this leads to a loss of service or only a degradation depends on the geometry of the satellites and the flight operation currently being executed by the aircraft.

4.5 Analysis - Simulator and Scenario Setup

Once a CONOPs and authentication scheme has been defined, the KPIs can be estimated and the impact to the users can be measured. Here, only nominal, non-spoofed and alert free, scenarios are considered.

Two different message streams were generated to deliver SBAS messages on the L5 frequency. The first, shown in Table 4.1, depicts a message set for the I-channel authentication cases. Table 4.2 shows the message structure for the Q-channel implementation. It should be noted that both charts read the same way; time increases from left to right and from top to bottom. These are all rigid and repeating message structures as well. For the I-channel implementation, the message "MT50" is defined as the authentication message type that carries the information shown in Figure 4.3 and Figure 4.4.



MT35	MT50	MT32, SV01	MT32, SV02	MT32, SV03	MT31 or 63
MT35	MT50	MT32, SV04	MT32, SV05	MT32, SV06	MT32, SV07
MT35	MT50	MT32, SV08	MT32, SV09	MT32, SV10	MT37 or 63
MT35	MT50	MT32, SV11	MT32, SV12	TMT32, SV13	MT63
MT35	MT50	MT32, SV14	MT32, SV15	MT32, SV16	MT32, SV17
MT35	MT50	MT32, SV18	MT32, SV19	MT32, SV20	MT 47 or 63

Table 4.1: I-Channel message stream with I-channel authentication

A receiver architecture was emulated in MATLAB that incorporated the CONOPS discussed in the section above and these receivers were simulated in aggregate in order analyze these operations. Figure 4.5 gives a pictorial description of the basic simulation architecture. First, inputs are given that define the number of receivers to be tested and for how long each is tested for. The word error rate (WER) that is incorporated in the simulation is also defined in this initial configuration stage. Then a series of messages is generated and delivered to the receivers through an emulated environment that at times causes messages to be demodulated incorrectly, leading to "Authentication Unavailable" events. Finally, all receivers keep track of the validity of the received messages and at the end of the simulation statistics are gathered to produce the KPIs.

One assumption made for these simulations was that the loss of two consecutive

	0	1	2	3	4	5
0	MT35	$MT32_{01}$	$MT32_{20}$	$MT32_{06}$	$MT32_{15}$	MT31
6	MT35	$MT32_{02}$	$MT32_{19}$	$MT32_{07}$	$MT32_{14}$	MT63
12	MT35	$MT32_{03}$	$MT32_{18}$	$MT32_{08}$	$MT32_{13}$	MT63
18	MT35	$MT32_{04}$	$MT32_{17}$	$MT32_{09}$	$MT32_{12}$	MT63
24	MT35	$MT32_{05}$	$MT32_{16}$	$MT32_{10}$	$MT32_{11}$	MT37
30	MT35	$MT32_{01}$	$MT32_{20}$	$MT32_{06}$	$MT32_{15}$	MT63
36	MT35	$MT32_{02}$	$MT32_{19}$	$MT32_{07}$	$MT32_{14}$	MT63
42	MT35	$MT32_{03}$	$MT32_{18}$	$MT32_{08}$	$MT32_{13}$	MT63
48	MT35	$MT32_{04}$	$MT32_{17}$	$MT32_{09}$	$MT32_{12}$	$MT47_{1}$
54	MT35	$MT32_{05}$	$MT32_{16}$	$MT32_{10}$	$MT32_{11}$	MT63
60	MT35	$MT32_{01}$	$MT32_{20}$	$MT32_{06}$	$MT32_{15}$	MT63
66	MT35	$MT32_{02}$	$MT32_{19}$	$MT32_{07}$	$MT32_{14}$	MT63
72	MT35	$MT32_{03}$	$MT32_{18}$	$MT32_{08}$	$MT32_{13}$	$MT47_{2}$
78	MT35	$MT32_{04}$	$MT32_{17}$	$MT32_{09}$	$MT32_{12}$	MT63
84	MT35	$MT32_{05}$	$MT32_{16}$	$MT32_{10}$	$MT32_{11}$	MT63
90	MT35	$MT32_{01}$	$MT32_{20}$	$MT32_{06}$	$MT32_{15}$	MT31

Table 4.2: I-Channel message stream with Q-channel authentication

MT32s for a specific satellite would lead to a loss of service 10% of the time. For most users it is common to have 10 satellites in view and in this case, it is assumed that one of those 10 is crucial to the availability of the service due to geometry.

Two WER models were used in producing the results. The first was a uniform random distribution of word errors in the messages. This is equivalent to sporadic losses of data where each message loss is independent of the reception of all other messages. In practice, it has been shown that this loss may not be completely independent, and that in some cases these errors are correlated in time [49]. These messages may be dropped due to interference events that last for several seconds or due to occlusions of the SBAS GEO which may be from a wing or other structure on the airplane. This correlated WER is modeled as a Markov chain and is depicted in Figure 4.6. Here π represents a transition probability from one state to another. From [49], these values are $\pi_{RR} = 0.999$ and $\pi_{LL} = 0.9078$.

Two different assumptions of the WER are considered for the Q-channel. These assumptions are that the word errors present on the I-channel and Q-channel are either correlated or uncorrelated. For the correlated case, it is assumed that if an error is present in a given I or Q-channel message, then it is also present in the other. In the uncorrelated case, errors on the I and Q-channel are treated as independent.

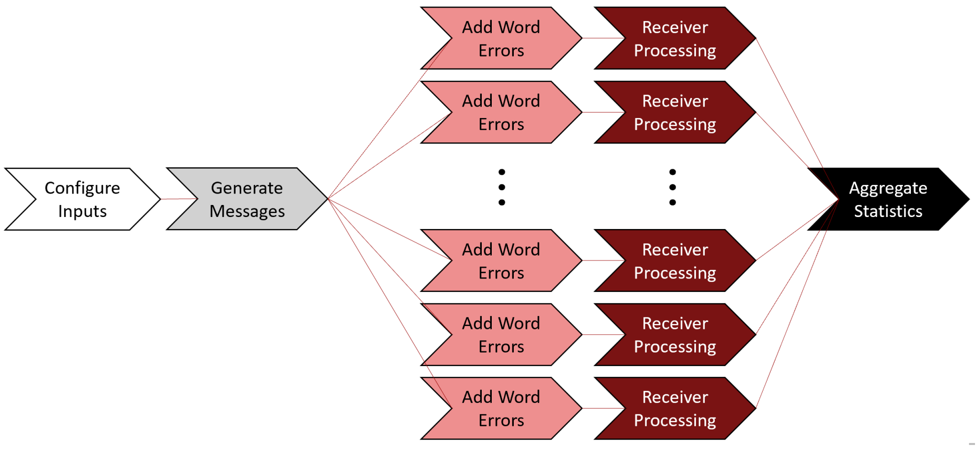
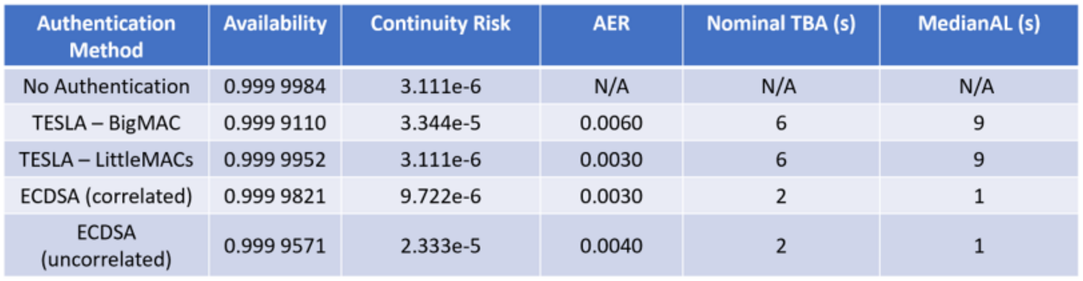


Figure 4.5: Simulator Architecture

4.6 Results

The simulation was configured to emulate 10000 receivers for 1 hour of SBAS messages, summing to a total of 36 million received messages for each authentication configuration. For the uniform WER distribution, an error rate of 10^{-3} was simulated, reflecting the requirement that todays SBAS users must be able to operate through word error rates of this magnitude. Table 4.3 shows the resulting KPIs for these candidate schemes along with a benchmark "No Authentication" case reflecting the level of performance available today.



Authentication Method	Availability	Continuity Risk	AER	Nominal TBA (s)	MedianAL (s)
No Authentication	0.999 9984	3.111e-6	N/A	N/A	N/A
TESLA – BigMAC	0.999 9110	3.344e-5	0.0060	6	9
TESLA – LittleMACs	0.999 9952	3.111e-6	0.0030	6	9
ECDSA (correlated)	0.999 9821	9.722e-6	0.0030	2	1
ECDSA (uncorrelated)	0.999 9571	2.333e-5	0.0040	2	1

Table 4.3: Simulation results for uniform WER of 10^{-3}

Several important insights can be gleaned from these results. The first is in the comparison between the availability and continuity of the no authentication case versus the cases with authentication. All show a degradation in performance, which

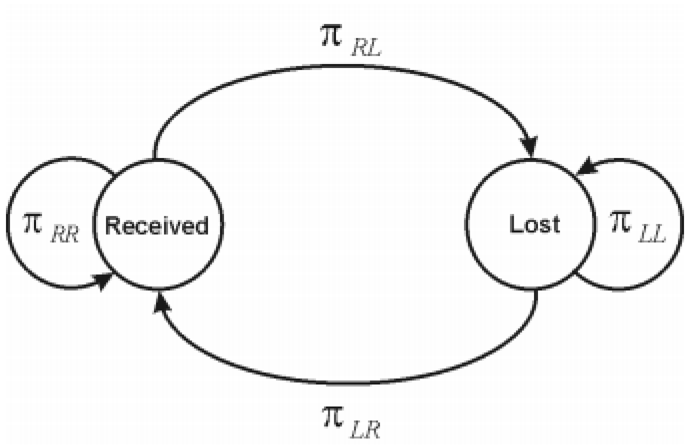


Figure 4.6: Markov chain model for burst errors

is to be expected, but the case of TESLA-LittleMACs, which was explicitly designed to mitigate the impact message loss, offers a service with minimal degradation in performance. Even though the authentication error rate (AER) is shown to be on the order of 3/1000, the service is robust to losing certain messages sporadically. Table 4.4 shows the results of the Markov chain WER model with significantly deprecated results in all cases.

Authentication Method	Availability	Continuity Risk	AER	Nominal TBA (s)	MedianAL (s)
No Authentication	0.992 9675	6.073e-3	N/A	N/A	N/A
TESLA – BigMAC	0.986 8317	6.514e-3	0.0239	6	9
TESLA – LittleMACs	0.989 7897	6.108e-3	0.0239	6	9
ECDSA (correlated)	0.992 3089	6.613e-3	0.0127	2	1
ECDSA (uncorrelated)	0.988 9997	9.685e-3	0.0231	2	1

Table 4.4: Simulation results for Markov chain WER model

Finally, a sensitivity analysis was run looking at different values of uniform WER distributions ranging from 10^{-4} to 10^{-1}. The results for the availability and continuity are shown in Figure 4.7 and Figure 4.8, respectively. At higher word error rates, the performance of all variants drops off. Availability remains quite high for some variants with WER $< 10^{-2}$. There is a clear trend in the continuity risk and how it is related to the WER. For low WER, the data becomes less reliable since the simulations did not appear to collect enough data to produce precise statistics, but the trends are clear from the higher word error rates. As promisingly shown in Table 4.3 and again seen here in Figure 4.8, the continuity risk of the TESLA-LittleMACs scheme tracks closely to that of the legacy service.

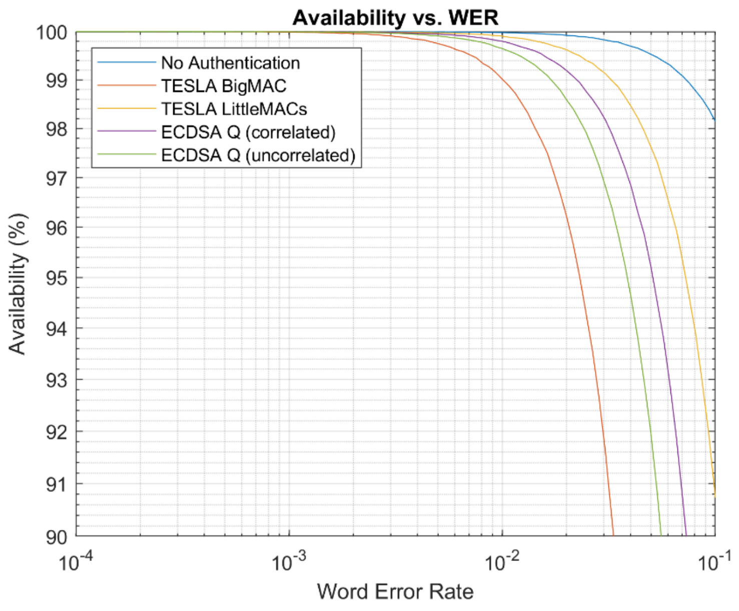


Figure 4.7: Availability vs. WER for a uniform distribution

4.7 TESLA LittleMACs and CONOPS implications

These results show that the impact of SBAS data authentication may be minimal with a proper design of the concept of operations. The concept of operations developed here does not allow for receivers to use any unauthenticated information that could be hazardous or misleading. An important result to note is that a variant of the TESLA scheme has shown promise in delivering a service that meets current performance requirements. The receiver emulators that have been built here also serve as an

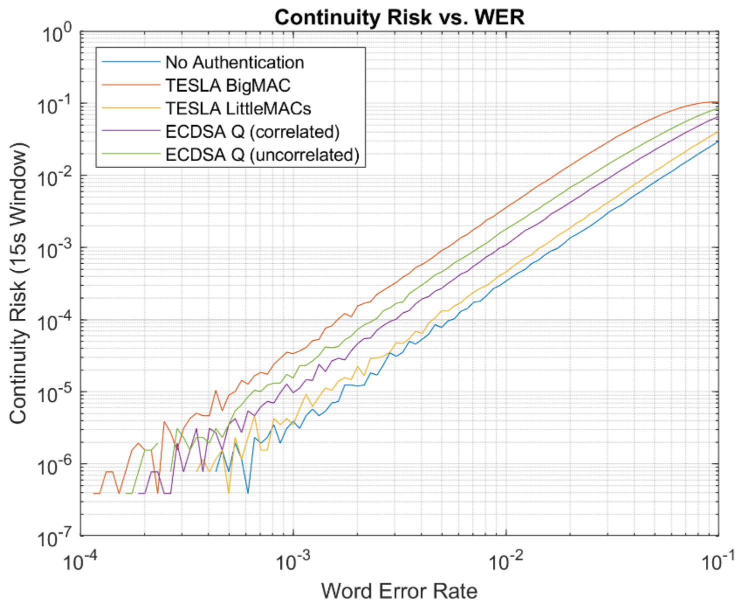


Figure 4.8: Continuity Risk vs. WER for a uniform distribution

instrumental tool in the development of these authentication CONOPS and can be modified and improved upon as more testing scenarios are developed.

One aspect that is not explored in this thesis is the concept of authenticating core constellation information using the SBAS L5 channel. There is still bandwidth that is available in all L5 rigid message schedules presented here. These are currently represented as MT63 and can be replaced with methods to authenticate data from the core GNSS constellations. Moreover, the signatures of these data can use the same keys as those used to sign the SBAS information.

Chapter 5

Public Key Infrastructure for SBAS

5.1 Introduction

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. Public keys that are disseminated to users must be verified to be authentic to defend users from being given false public keys. These private/public key pairs are susceptible to numerous attacks and become more vulnerable the longer they are employed 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 [20, 50, 51, 44, 45, 19, 16, 52, 53] and one publication has offered a potential solution [54]. 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 chapter 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 in nominal conditions. 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.

The chapter introduces a candidate PKI and method of key management for both I-channel and Q-channel authentication schemes. A tool referred to 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 SBAS 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 word error rate (WER)) 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 chapter.

5.2 Public Key Infrastructure and Key Management

In this section, a candidate PKI is introduced for both the L5I and L5Q authentication schemes. As was done in previous chapters, the design of a Q-channel ECDSA for EC-Schnorr authentication scheme is similar except for the number of bits available to broadcast OTAR information. As ECDSA is the more limiting case, it is the scheme that is focused on in this analysis.

The design of a good PKI will need 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 (period for which a key pair is used, typically based on the security of the key pair) and are used for the authentication of SBAS data, while the larger signatures/keys are used to authenticate the smaller keys and the most secure keys are considered the root of trust. A widely accepted international PKI standard for use over the internet is the X.509 standard [55]. Using this standard, X.509 certificates contain information about the public key as well as a host of other information such as time of validity, algorithm information and name of the identity the certificate is issued to. For a broadcast system such as SBAS, some of this information is not needed and so a PKI is designed here that is specifically tailored for use in SBAS.

The PKI of ECDSA consists of two levels of keys. Level-2 private keys are used by the SBAS service provider to authenticate SBAS messages and SBAS enabled receivers use level-2 public keys to check the authenticity of the incoming messages. These level-2 public keys 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 encrypted and authenticated value of the level-1 public key is preinstalled in the receiver by the receiver manufacturer. The level-1 public keys are encrypted using AES-256 (Advanced Encryption Standard with a 256-bit key) [56] in ECB mode [57] with PKCS#7 (Public Key Cryptography Standards) padding [58]. This encryption creates a ciphertext, $c_j = E\left(k_j^e, PK_j\right)$, where $E()$ is the AES-256 encryption function that encrypts the jth public key, PK_j using an encryption key, k_j^e. This public key must also be authenticated so that the receiver can be certain that the public key has been decrypted correctly. In order to minimize the information that must be sent over the air, it is proposed that an authentication tag is prestored in the receiver using the form $\text{Tag}_j = S\left(k_j^s, \left[c_j \left\|k_j^e\right\| PK_j\right]\right)$, where $S()$ is the signing algorithm employed by ECDSA for level-1 keys. This algorithm signs the concatenated element, $\left[c_j \left\|k_j^e\right\| PK_j\right]$, with the jth level-1 secret key, k_j^s, that is paired with the jth public key, PK_j. All pairs $\left[c_j, \text{Tag}_j\right]$ are preinstalled in the receiver. The SBAS service providers will continuously send each current k_j^e and next k_{j+1}^e in an OTAR message to "unlock" the current and next level-1 keys. Upon receiving k_j^e a receiver will decrypt the public key using the AES-256 decryption algorithm $PK_j = D(k_j^e, c_j)$ and authenticate the public key using the ECDSA verification method for level-1 signatures, $V\left(PK_j, \left[c_j \left\|k_j^e\right\| PK_j\right], \text{Tag}_j\right) = \text{pass/fail}$. To further minimize the required bandwidth to unlock level-1 signatures, it is proposed that the service provider only send over k_j^e without any identifying information of which public key is being unlocked. It is up to the receiver to test k_j^e with the different pairs $[c_i, \mathrm{Tag}_i]$ until a pass is found. Future work will examine the tradeoff between sending key pair identifier information and this method of receiver key/tag testing.

These suggested security strengths for the level-1 and level-2 keys are implicitly tied to their intended uses and respective cryptoperiods. As defined by NIST [59], "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 [59], but does provide general guidelines for the specific use case of unclassified federal documents mentioned above.

Details of the level-1 and level-2 keys are outlined in Table 5.1 along with their cryptoperiods. The level-1 keys are expected to have a cryptoperiod of two years and if there are 64 encrypted level-1 public keys preinstalled a receiver would have quadruple the number of keys they would need during an estimated lifetime of 30 years. With 64 ciphertext and tag combinations, the overall storage requirement is roughly 10-15 kB. 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.

It should be noted that the key lengths and OTAR bitrates might be different from those expressed in earlier chapters. These values have changed multiple times over the course of this work and will likely change again after this thesis is printed. Regardless of these changes, the analyses carried out in this thesis are still representative of what is achievable using these cryptographic techniques.

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/intermediate keys of the TESLA keychain. TESLA root/intermediate 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 5.1 and Figure 5.2 show the relative cryptoperiod of the different keys for L5Q ECDSA and L5I TESLA, respectively, and Table 5.1 gives a summary of the parameters for both implementations.

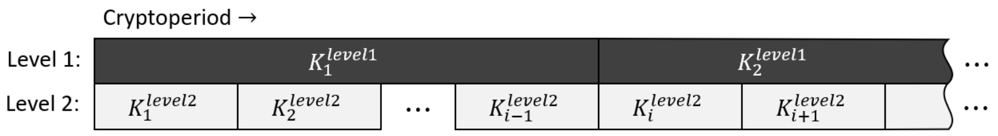


Figure 5.1: Relative cryptoperiods between L5Q keys

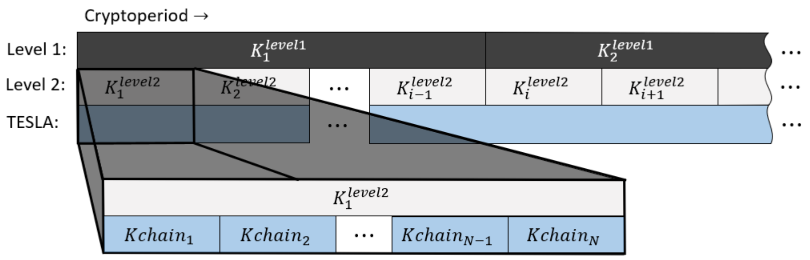


Figure 5.2: Relative cryptoperiods between L5I keys

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 have each SBAS service provider create the level-1 keys. This way, when a new level-2 key is introduced, the SBAS service provider can create the signature for the level-2 key and send it along with its signature using OTAR bits, but there are several challenges that face this implementation. This would require the process of creating, storing, and using the level-1 keys to be carried out in a secure manner. The process of developing the infrastructure to sufficiently secure this information would likely be costly to the SBAS service provider. In addition, these level-1 keys would only be entrusted to the SBAS service provider who created them and so any receiver that wished to use an authentication service offered by another SBAS service provider using a similar PKI would require the receiver to be preinstalled with multiple sets of encrypted level-1 keys.

Summary of Parameters for L5I TESLA and L5Q ECDSA PKI	
L5I TESLA OTAR bitrate	26 bits/6s
L5Q ECDSA OTAR bitrate	52 bits/2s
L5I TESLA Key Length	115 bits
L5I TESLA MAC Length	5x15 bits
L5I TESLA Salt Length	30 bits
L5I TESLA Keychain Nominal Cryptoperiod	1 week
L5I/L5Q ECDSA level-2 Signature Length	448 bits
L5I/L5Q ECDSA level-2 Public Key Length	225 bits
L5I/L5Q Level-2 Nominal Cryptoperiod	6 months
L5I/L5Q ECDSA level-1 Signature Length	768 bits
L5I/L5Q ECDSA level-1 Public Key Length	385 bits
L5I/L5Q Level-1 Nominal Cryptoperiod	2 years
L5I/L5Q # of locally stored ECDSA level-1 public keys	64
L5I/L5Q Storage space required for ECDSA level-1 keys	10-15kB

Table 5.1: Summary of parameters for L5I TESLA and L5Q ECDSA PKI

Another option is to contract the responsibility of securing and using level-1 private keys to a 3rd-party certificate authority (CA) (E.g. ICAO) that is already equipped to protect information with this level of sensitivity. Each SBAS service provider would still manage their own set of level-2 keys. When a new level-2 key needs to be introduced, the CA is sent the new level-2 public key by the SBAS service provider 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 SBAS service providers to implement a secure PKI while minimizing the added cost to their infrastructure and new SBAS service providers would be able to come online without having to change receiver equipment that has already been fielded. In addition to this, if a CA were agreed upon between all SBAS providers, only one set of encrypted level-1 keys would need to be installed on receivers and would greatly 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 a concatenation of the expiration date with its associated key 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 in addition to the current keys. In the case of level-1 keys, the current and next encryption keys used to encrypt the level-1 public keys are transmitted using OTAR bits and both are authenticated with information preinstalled on the receiver. Both levels of keys do not necessarily need to be used until their expirations. They can be superseded at any time by the service providers and users will automatically switch over to the next set of public keys that were broadcast during the life of the previous keys.

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 encrypted 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 revoked. 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 change once every two years.

All level-2 and level-1 keys will have associated expirations, but SBAS service providers will have the option to revoke them sooner if required. Level-2 and level-1 keys are revoked the moment the next keys are used. Once these keys have been revoked, 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. By sending both the current and next keys, SBAS users will be able to smoothly transition from one key to the next with no gap in authentication service. This particular point is important as it allows the service to retain high authentication availability and continuity through the transition of these keys.

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 paragraphs, the 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.

In the case of L5Q, where ECDSA is used, there are 52 bits reserved for OTAR every 2 seconds, referencing Table 5.1. OMTs do not require a preamble. The basic structure of an OMT for L5Q ECDSA is shown in Figure 5.3. For clarification, Figure 5.3 depicts the contents of the “OTAR” boxes seen in Figure 4.2. The first 3 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 B shares detailed information on each individual OMT. Referencing Table B.1, OMT1 is the current level-2 public key concatenated with its associated level-1 signature. 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 SBAS messages. Therefore, OMT1 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 chapter. When a receiver starts from a cold start, it will go through the steps outlined in Figure 5.5 to begin authenticating data.

In the case of L5I, level-2 keys are used to authenticate the root/intermediate keys the TESLA keychain and the OMT design for L5I is similar to the L5Q OMT. The number of bits transmitted per authenticated message is different, however. Figure 5.4 shows an example structure of an L5I OMT. Appendix C 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 along with its associated level-2 signature. This message is broadcast most frequently as it is the minimum amount of information needed to begin authenticating the SBAS message assuming the receiver has the correct level-1 public key. From a cold start there is one more step that precedes step 1 in Figure 5.5. 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 checks the TESLA keychain parameters against the current broadcast of OMT1.

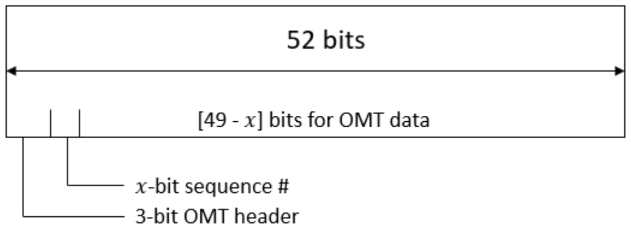


Figure 5.3: L5Q ECDSA OMT message structure

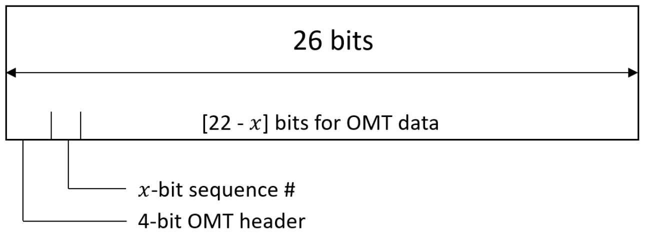
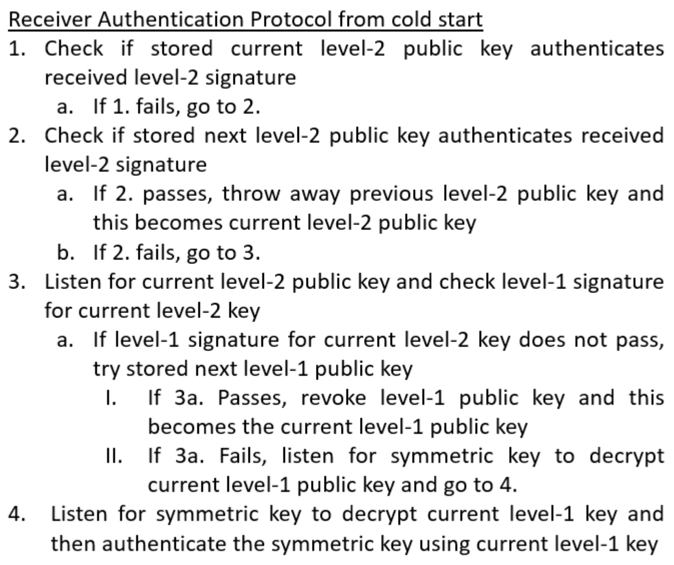


Figure 5.4: L5I TESLA OMT message structure

5.3 MCOS: Monte Carlo OTAR Simulator

A simulator known as the Monte Carlo OTAR Simulator (MCOS) was developed to analyze the PKIs described above. MCOS is broken up into 3 stages as shown in Figure 5.6. The first stage is the definition of the OMTs 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 through an SBAS 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.

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 allow 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



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. Listen for current level-2 public key and check level-1 signature for current level-2 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, revoke level-1 public key and this becomes the current level-1 public key
      II. If 3a. Fails, listen for symmetric key to decrypt current level-1 public key and go to 4.
4. Listen for symmetric key to decrypt current level-1 key and then authenticate the symmetric key using current level-1 key

Figure 5.5: Receiver authentication protocol from cold start

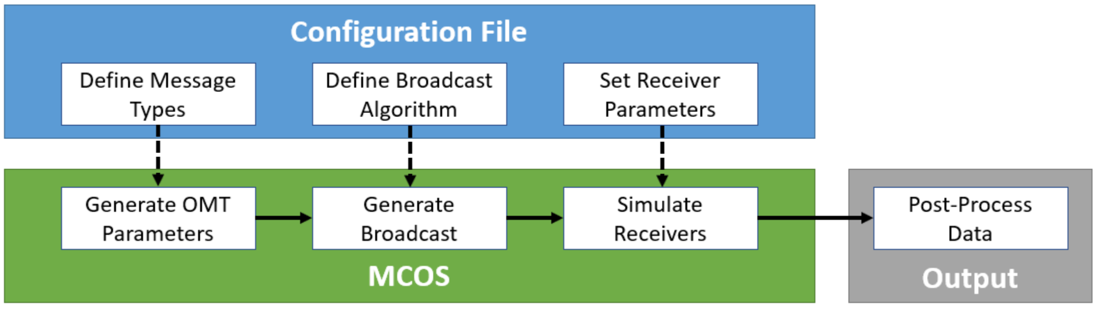


Figure 5.6: MCOS high-level configuration

such algorithm similar to packet fair queueing (PFQ) [1] 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. Figure 5.7 from [1] depicts the difference between a PFQ (a) and simple broadcast example (b). Both broadcasts have 50% of the bandwidth dedicated to channel 1 and 25% dedicated each to channel 2 and 3. Unlike the simple broadcast example, PFQ generates a message cadence that is proportional to the desired channel bandwidth allocation, and so it delivers channel 1 messages with a shorter maximum time between transmissions than that of channels 2 and 3. This is useful if channel 1 is the most important channel and the intent is to decrease the time for a receiver to receive messages from channel 1.

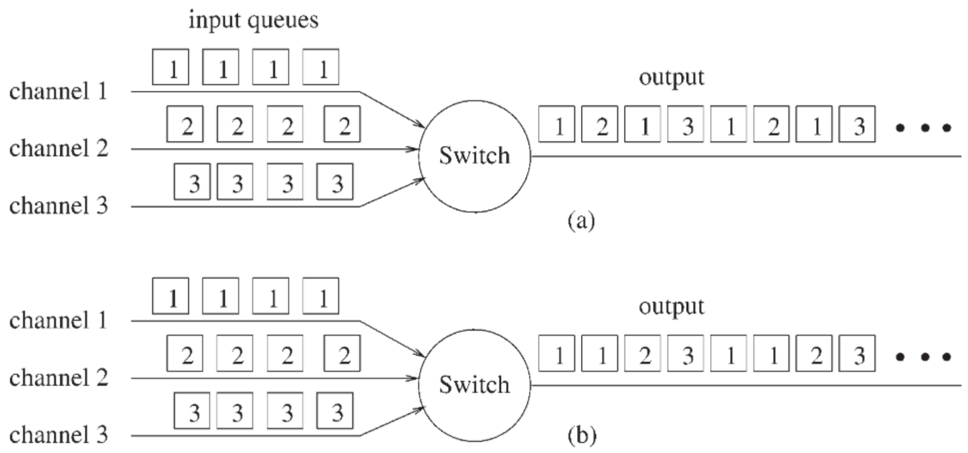


Figure 5.7: Broadcast outputs from PFQ (a) and an example broadcast (b) [1]

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. It should be noted that in its current form, MCOS only produces randomized results if the WER is nonzero. Otherwise, the broadcast algorithm and distribution of the receiver start times is deterministic. 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 environment that may or may not lead to the loss of parts of the OMTs. This leads to a Word Error Rate (WER) that effects their ability to receive all messages in the broadcast stream. Example outputs can be seen in Figure 5.8 and Figure 5.9 in the analysis section of the chapter. The derived results, such as mode and average time to receive a message can then be used to iterate on the broadcast algorithm and OMT parameters. The simulator is capable of running simulations involving hundreds of thousands of receivers in a number of seconds on an i7 processor. This code is open-source and made available through the Stanford GPS Lab GitHub page: `https://github.com/stanford-gps-lab/mcos`.

It should be noted that MCOS does not currently simulate the time from cold start to frame synchronization for a receiver. All time to receive values reported in this analysis represent the time to receive after successful frame synchronization.

5.4 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 5.1.

Variations of weights for the messages broadcast by the two PFQ broadcast algorithms were analyzed. The resulting weights chosen for the results shown in this chapter was a ratio of 100:1 in favor of OMT1 with respect to the remaining messages for L5Q ECDSA and for the case of L5I TESLA, a weight ratio of 100:50:1 was used favoring OMT1 to OMT2 to the remaining messages. Table 5.2 shows several performance parameters related to the chosen weighting ratio.

	Average time to receive all messages	**Average time to receive current authenticated level-2 key**	**Average time to receive current TESLA keychain and salt**
ECDSA L5Q	16.6 minutes	55.0 seconds	N/A
TESLA L5I	3.0 hours	10.8 minutes	7.0 minutes

Table 5.2: Broadcast results for L5I and L5Q

An example simulation output for the L5Q ECDSA design is shown below in Figure 5.8 and Figure 5.9. These figures are normalized histograms showing the number of receivers that received the message at the times shown in the horizontal axis. Figure 5.8 shows the time to receive OMT1 and Figure 5.9 shows the time to receive all messages for all receivers simulated for the case where WER=0.

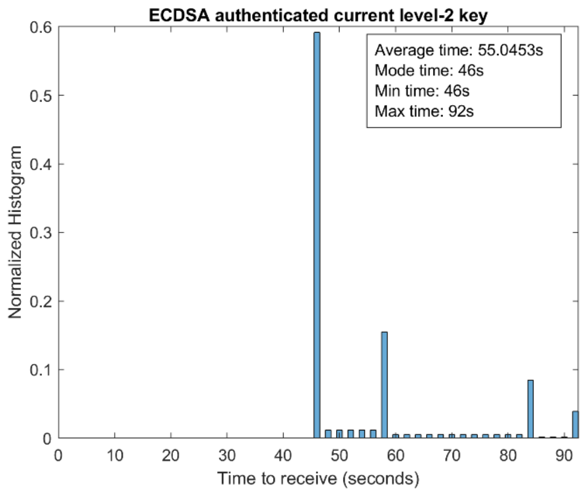


Figure 5.8: Times to receive authenticated current level-2 keys for ECDSA

5.4.1 Effect of WER

An analysis was performed over a range of word error rates for each case. For each WER, a full simulation is run, producing histograms similar to the results shown Figure 5.8 and Figure 5.9. These results are then aggregated to produce Figure 5.10. 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 WER 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 decoding all parts of a message deteriorates as the WER increases past 10^{-3}. Since this is a Monte Carlo simulation, it does not show all possible outcomes, and so jumps 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.

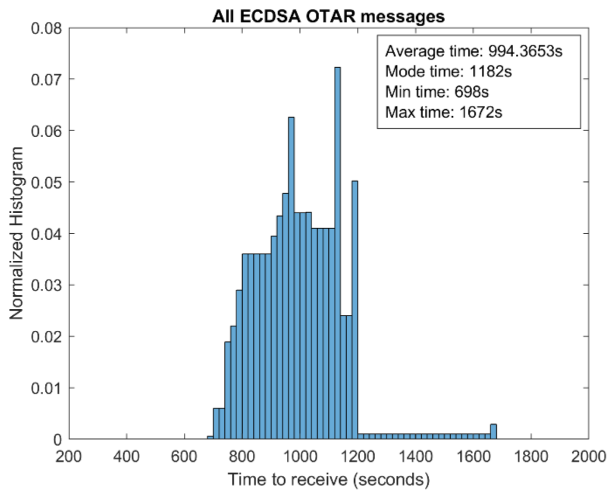


Figure 5.9: Times to receive all OMTs for ECDSA

5.5 Broadcast Algorithm Analysis

This broadcast algorithm used up until this point has been using a standard form of the PFQ algorithm proposed in [1]. One drawback to this standard implementation is that there is no guarantee on the minimum time between transmission of certain message types. For instance, if there was a requirement that OMT1 be transmitted at least every 60s, the PFQ algorithm would need to be modified. In order to address this issue with 'PFQ-Standard', a second algorithm, referred to as 'PFQ-Semi-Rigid', is implemented. 'PFQ-Semi-Rigid' acts similarly to 'PFQ-Standard' with the only difference being that the maximum time between delivery of important OMTs can be set. For instance, in the case where OMT1 is the most important message, this algorithm allows the designer to specify the maximum time interval between their transmission, effectively ensuring an upper bound on time between transmission of these messages. This algorithm allows the broadcast to interrupt other messages temporarily while OMT1 is sent. For this study, a maximum interval time of 60 seconds for OMT1 was set and the resulting comparison is shown in Table 5.3.

There are some advantages to 'PFQ-Semi-Rigid' when compared with 'PFQ-Standard'. 'PFQ-Semi-Rigid' achieves better performance for receivers starting from a cold start with no level-2 key information. The penalty is a much longer time

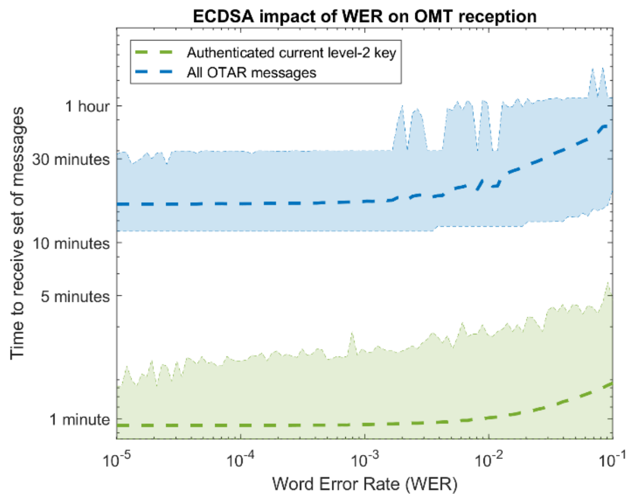


Figure 5.10: ECDSA impact of WER on OMT reception

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	55.0	92 sec	16.6 min
PFQ-Semi-Rigid	48.6 sec	60 sec	41.4 min

Table 5.3: Broadcast algorithm analysis

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.

5.6 Chapter Concluding Remarks

A PKI was developed around the two schemes that allows public keys to be updated solely over the air in nominal conditions. This PKI was developed in a way that allows the SBAS service provider to distribute and allocate levels of security in order to more firmly safeguard against potential attacks. Central to both PKIs was the use of a trusted level-1 source, 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 encrypted and authenticated 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 allowing the receivers to authenticate any level-2 keys introduced by the SBAS service provider. In the case of ECDSA/EC-Schnorr, a two-level system was introduced, allowing for the SBAS service provider to generate their 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 used to authenticate the root/intermediate keys of these keychains. The motivation for this being that level-2 signatures have bit lengths significantly shorter than level-1 signatures, allowing a receiver on cold start to authenticate the current keychain much quicker than it would be able to otherwise. For both designs, a series of OTAR message types was derived with the intent on delivering all required PKI information in nominal conditions using minimal bandwidth. An important part of these OTAR broadcasts is both the current and next set of public keys are broadcast. This allows for seamless transitions from old to new keys at the discretion of the service provider.

In order to test this PKI architecture, an analysis tool, MCOS, was developed 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 design parameters. Results showed that with an L5Q ECDSA scheme in nominal conditions, receivers would be able to receive all OTAR messages within an average time of less than 17 minutes. This performed far better than the L5I TESLA scheme which had an average time of nearly two hours. That being said, if level-2 and level-1 public keys have cryptoperiods on the order of months to years, two hours make up a small fraction of that time. A series of analyses were performed looking at the effect different PKI input parameters and word error rates had on performance metrics concerning the time to receive important PKI messages. Finally, an analysis comparing different broadcast algorithms was carried out. If requirements are placed on the time between transmission of the level-2 public key information, the broadcast can be adapted to meet these requirements at the cost of extending the time between transmission of other OTAR messages.

Outside of solely using the extra bits in the signature messages for OTAR, there is potential to broadcast a new key management message that effectively replaces 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.

Two important contributions are presented in this chapter. The first is that an SBAS data authentication system can perform key management solely over the air in nominal conditions with cryptographic information preinstalled in authentication enabled receivers. The second is the contribution of a strong candidate PKI that is designed to be standardized internationally for current and future SBAS service providers.

Chapter 6

Receiver States

6.1 Introduction

This chapter introduces authentication states for SBAS receivers describing all processes ranging from pre-authentication verification to nominal authenticated operations. Most importantly, the secure transitions between these states is defined and receiver operations are proposed to better define what the receiver's role is in delivering a secure service. Different authentication techniques also require tailored modifications for the receiver CONOPS. As in previous chapters, this chapter focuses on an I-channel design for TESLA and a Q-channel design for ECDSA/EC-Schnorr. An important distinction between the designs discussed here and those discussed in the rest of the literature is that an "authenticate-then-use" approach is taken with the SBAS data [60] as was introduced in chapter 4. This protects the SBAS-enabled receiver from using potentially spoofed data while ensuring that receivers use alerts immediately to preserve integrity.

6.2 Information Content Categories

As was introduced in chapter 4, there are two categories of SBAS message content that dictate how the received information is processed. The first category consists of integrity information that communicates degraded integrity data compared with previously received data, and the second category contains all other SBAS message content. Using this authentication process, all information received in the second category is buffered and not used until it is authenticated. In contrast, the integrity information for each satellite is used immediately by the receiver when this information conveys a decrease in integrity compared with the current integrity information. In this case, the received data is used prior to authentication. For the purposes of this work, the Dual Frequency Range Error (DFRE) sigma derived from the Dual Frequency Range Error Indicator (DFREI) or the Dual Frequency Range Error Change Indicator (DFRECI) is considered the main source of integrity information. The Do Not Use message (MT-0) also provides a means to protect the integrity of users and also requires immediate action. Figure 6.1 shows the two message content categories described above.

The DFRE communicates confidence information on the range to each satellite with available corrections. When the DFREI for a satellite is increased, it indicates that the variance on the range to that satellite is increased, and therefore the confidence has decreased. When the DFREI of a satellite is broadcast as the highest value, it is an indication that the satellite should not be used in the position, velocity and time (PVT) and protection level (PL) calculations. By allowing the receiver to use increased DFREI information before it has been authenticated, the Time-To-Alert (TTA) is preserved for the SBAS service. In contrast, when a satellite's DFREI decreases, that change in DFREI is buffered (similar to the second SBAS message category) and is only incorporated after it is authenticated. If the DFREI remains the same it is buffered similar to the decreased DFREI case. Generally, the receipt of a DFREI resets the timeout of the DFREI while the receiver waits to authenticate it before its use. The loss of four consecutive messages still results in all DFREI information timing out. This authentication concept prevents a window of vulnerability that would occur if a receiver were allowed to start using SBAS data prior to authenticating the data.

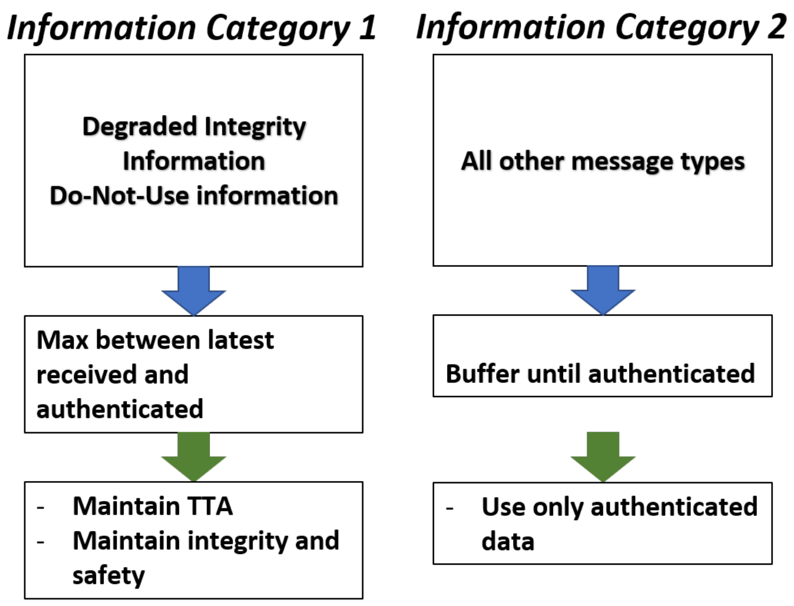

Information Category 1
Information Category 2
Degraded Integrity Information Do-Not-Use information
All other message types
Max between latest received and authenticated
Buffer until authenticated
- Maintain TTA
- Maintain integrity and safety
- Use only authenticated data

Figure 6.1: Information Categories

6.3 Receiver States

Receiver states refer to states held internally within the receiver regarding SBAS authentication. As an important note, these states are notional and do not necessarily need to be known to the receiver. They serve as an important tool for those developing how authentication should be incorporated in SBAS receivers. These states are designed to be as clear as possible to understand how a receiver operates through all conceivable scenarios that might occur during its lifetime. How these states are arrived at and what they signify is discussed here. How these states translate into alerts for pilots in potential spoofing situations and service outages is left for future work. There are three internal receiver states: “Initialization”, “Authenticating” and “Authentication Failed”. These states and their transitions are shown in Figure 6.2. The over-the-air-rekeying (OTAR) and key management process is not shown as a part of the receiver states. It is instead a process that is constantly running in the background. More information concerning OTAR and key management can be found in [48] and in the previous chapter.

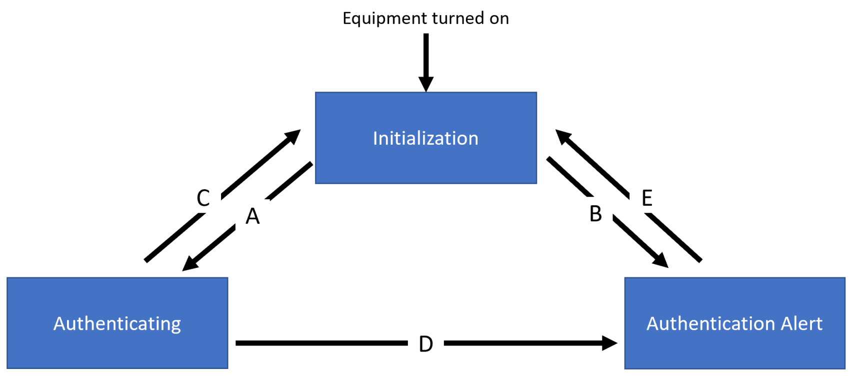


Figure 6.2: Internal States of an authenticated SBAS receiver

The “Initialization” state occurs during receiver start up and is required to establish trust in the SBAS broadcast. This state covers time synchronization and validation which is important for the TESLA I-channel scheme. Typically, initialization is a short, first step as the receiver enters into service from a warm or cold start. There are some circumstances, however, when a receiver would need to enter the initialization state from the other receiver states. These are mentioned briefly in Table 6.2.

“Authenticating” is the state at which the receiver has begun to authenticate SBAS messages. This state occurs irrespective of whether there is enough data to deliver an SBAS corrected PVT and protection levels (PLs). This is the nominal working state of the receiver.

If there is an epoch at which an authentication fails, the receiver internal state moves to the “Authentication Alert” state. There is an ongoing discussion into what this state means for the receiver and what implications this would have on other users in the same airspace. What is clear is that the only way a receiver would be able to transition from an “Authentication Alert” state back into an “Authenticating” state is by first going back through the “Initialization” state to reestablish trust in the service.

Table 6.1 gives a general summary of what occurs during each of the receiver states. The state transitions internal to the receiver are shown in Table 6.2 along with rough examples of when these transitions might occur. The letters associated with the transitions are shown in Figure 6.2.

State	Description	PVT Outputs	Authentication Status Output Value
Initialization	Tracking satellites, have not yet validated all keys required to authenticate messages	According to legacy requirements when SBAS is not available	None
Authenticating	SBAS messages are authenticated	According to legacy requirements	True
Authentication Alert	SBAS messages fail authentication, service is not available, and an alert is raised	No computed data – or – output but flagged consistent with legacy requirements when misleading information is detected, integrity alert	False

Table 6.1: General description for SBAS receiver authentication states

Transition	States transitioning	Transition description
A	Initialization to Authenticating	Trust is established in the received messages, all required keys have been validated, and these messages have been authenticated. With these messages, the SBAS service is available according to legacy requirements.
B	Initialization to Authentication Alert	Trust cannot be established in the received messages. The receiver has received keys, but these keys either fail to be validated or the incoming messages fail authentication.
C	Authenticating to Initialization	Corner cases where keys are revoked, or keys are changed shortly after initialization such that the receiver no longer has a valid current set of keys. The receiver needs to reestablish trust in the broadcast once more.
D	Authenticating to Authentication Alert	The receiver has the correct keys and the authentication fails.
E	Authentication Alert to Initialization	After an Authentication Alert state, the receiver attempts to reestablish trust in the SBAS broadcast. This transition will be better defined once the Authentication Alert state is also better defined.

Table 6.2: Example receiver internal state transitions

6.4 Internal operations of the receiver states

This section gives a high-level description of the internal operations undertaken by the receiver in each of the 3 states mentioned above. It is important to note that these state processes pertain to a single SBAS channel. SBAS receivers are required to track more than one SBAS channel during operation and these processes will be parallelized between these channels.

6.4.1 Initialization

The "Initialization" state governs the process by which a receiver gains confidence in the authenticity of the incoming SBAS data. If the receiver has been in an Authentication Alert state, it will transition into the Initialization state once more before transitioning to the Authenticating state. Figure 6.3 shows a high-level flow chart of how the receiver progresses through this state.

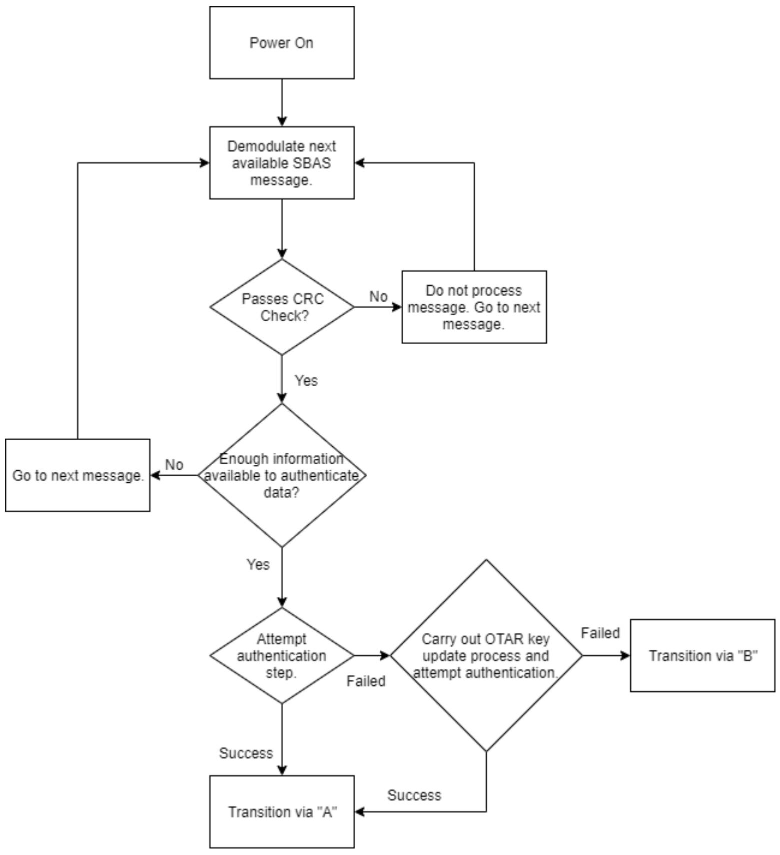

Power On
Demodulate next available SBAS message.
Passes CRC Check?
No
Do not process message. Go to next message.
Yes
Enough information available to authenticate data?
No
Go to next message.
Yes
Attempt authentication step.
Failed
Carry out OTAR key update process and attempt authentication.
Failed
Transition via "B"
Success
Success
Transition via "A"

Figure 6.3: Initialization state processes

If the receiver is powering on, it goes through its normal acquisition procedures until it is able to track and demodulate the incoming SBAS data. Figure 6.3 demonstrates the logical path a receiver takes as it attempts to authenticate the data. The authentication attempt includes ensuring loose-time synchronization in the case of TESLA [13]. If the attempt is deemed a success, the receiver proceeds to the "Authenticating" state. If the attempt fails, the receiver goes through the over-the-air-rekeying update process to update the keys. If during that process it is found that the keys fail their authentication as well, the receiver transitions to the "Authentication Alert" state. The process by which the keys are verified has been outlined in [48] and a more detailed OTAR receiver based process is left for future work. As mentioned in Table 6.1, the receiver output while in the "Initialization" state is according to legacy requirements when SBAS is not available. A major benefit to this design is that if the SBAS service is being spoofed, none of the forged data will be used in a PVT/PL solution.

6.4.2 Authentication Alert

This state is the least defined out of all the states presented here. There are many questions that still need to be answered: If an authentication has failed, how should a receiver communicate that information? Should an alert be raised to the pilot? Should air traffic controllers be informed directly? Should the receiver not burden the pilot with this information in potentially hazardous situations and keep the knowledge internal? Should the receiver produce a PVT solution? And then finally, after a receiver has entered into an "Authentication Alert" state, how should a receiver transition safely out of that state, back into an eventual "Authenticating" state once more? These questions are more than technical in nature and will need to be addressed in cooperation with the aviation community and are out of the scope for this thesis.

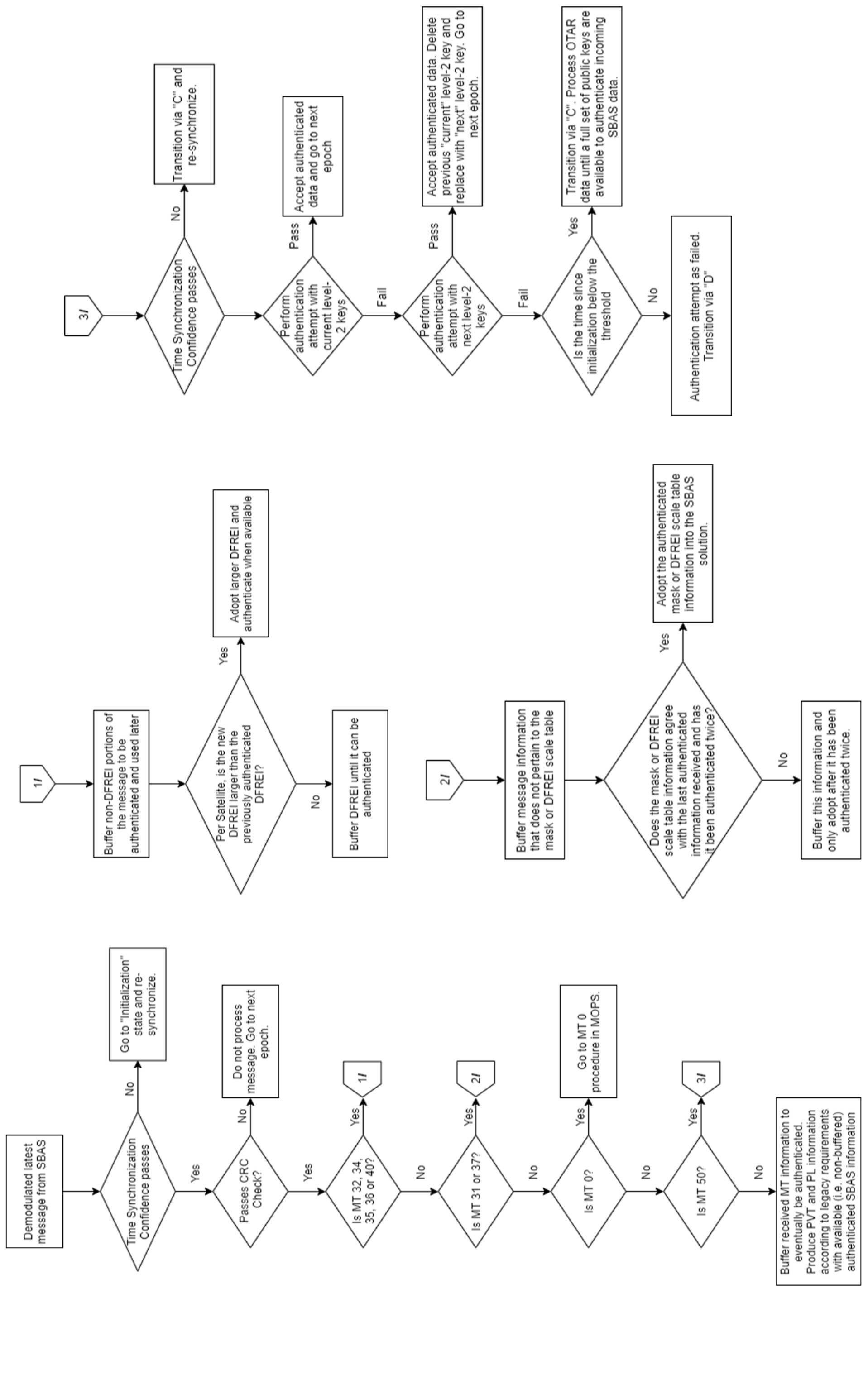

Demodulated latest message from SBAS
Time Synchronization Confidence passes
No
Go to "Initialization" state and re-synchronize.
Yes
Passes CRC Check?
No
Do not process message. Go to next epoch.
Yes
Is MT 32, 34, 35, 36 or 40?
Yes
1I
No
Is MT 31 or 37?
Yes
2I
No
Is MT 0?
Yes
Go to MT 0 procedure in MOPS.
No
Is MT 50?
Yes
3I
No
Buffer received MT information to eventually be authenticated. Produce PVT and PL information according to legacy requirements with available (i.e. non-buffered) authenticated SBAS information
1I
Buffer non-DFREI portions of the message to be authenticated and used later
Per Satellite, is the new DFREI larger than the previously authenticated DFREI?
Yes
Adopt larger DFREI and authenticate when available
No
Buffer DFREI until it can be authenticated
2I
Buffer message information that does not pertain to the mask or DFREI scale table
Does the mask or DFREI scale table information agree with the last authenticated information received and has it been authenticated twice?
Yes
Adopt the authenticated mask or DFREI scale table information into the SBAS solution.
No
Buffer this information and only adopt after it has been authenticated twice.
3I
Time Synchronization Confidence passes
No
Transition via "C" and re-synchronize.
Perform authentication attempt with current level-2 keys
Pass
Accept authenticated data and go to next epoch
Fail
Perform authentication attempt with next level-2 keys
Pass
Accept authenticated data. Delete previous "current" level-2 key and replace with "next" level-2 key. Go to next epoch.
Fail
Is the time since initialization below the threshold
Yes
Transition via "C". Process OTAR data until a full set of public keys are available to authenticate incoming SBAS data.
No
Authentication attempt as failed. Transition via "D"

Figure 6.4: I-channel TESLA Authenticating State

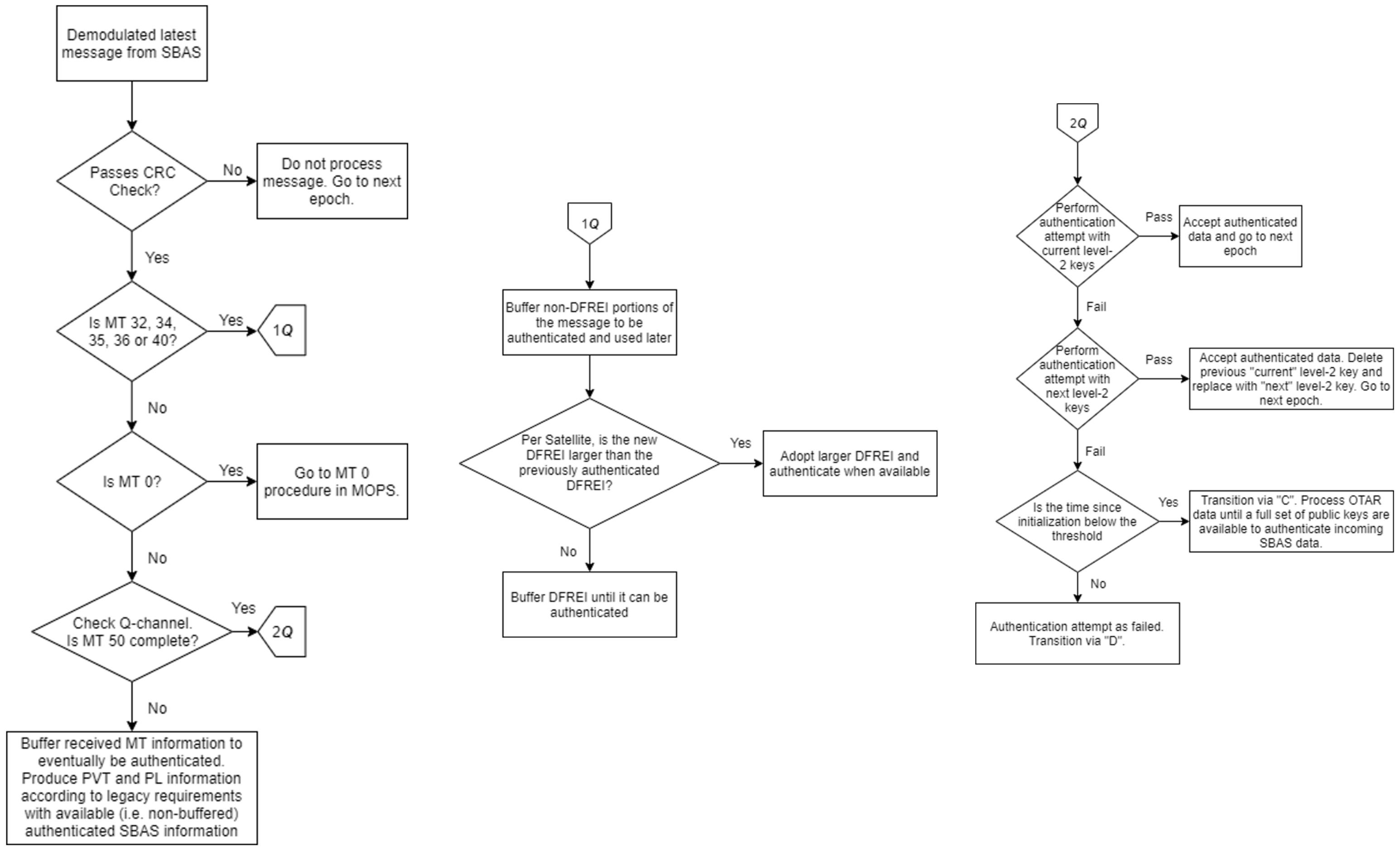

Demodulated latest message from SBAS
Passes CRC Check?
No
Do not process message. Go to next epoch.
Yes
Is MT 32, 34, 35, 36 or 40?
Yes
1Q
No
Is MT 0?
Yes
Go to MT 0 procedure in MOPS.
No
Check Q-channel. Is MT 50 complete?
Yes
2Q
No
Buffer received MT information to eventually be authenticated. Produce PVT and PL information according to legacy requirements with available (i.e. non-buffered) authenticated SBAS information
1Q
Buffer non-DFREI portions of the message to be authenticated and used later
Per Satellite, is the new DFREI larger than the previously authenticated DFREI?
Yes
Adopt larger DFREI and authenticate when available
No
Buffer DFREI until it can be authenticated
2Q
Perform authentication attempt with current level-2 keys
Pass
Accept authenticated data and go to next epoch
Fail
Perform authentication attempt with next level-2 keys
Pass
Accept authenticated data. Delete previous "current" level-2 key and replace with "next" level-2 key. Go to next epoch.
Fail
Is the time since initialization below the threshold
Yes
Transition via "C". Process OTAR data until a full set of public keys are available to authenticate incoming SBAS data.
No
Authentication attempt as failed. Transition via "D".

Figure 6.5: Q-channel ECDSA and EC-Schnorr Authenticating State

Chapter 7

Quantum Computing

7.1 Introduction

Modern data authentication has been around for nearly half a century, and there exist robust algorithms that have been proven to be secure. These algorithms use cryptographic primitives that assume the hardness of certain problems, such as the discrete logarithm problem, but these assumptions only hold for classical computers. Quantum computers are being developed all over the world today, and their future development will render many of these popular schemes vulnerable to attacks. Several corporations have announced programs developing quantum computers, and a few have already taken products to market [61]. While there are many that are chasing this technology, it is unclear when quantum computers at large scale will become a reality with some predictions ranging from 10s of years to never. If large scale quantum computers come to fruition, data authentication algorithms designed for SBAS will need to be secure against them so it is imperative that all potential risks to these algorithms are addressed.

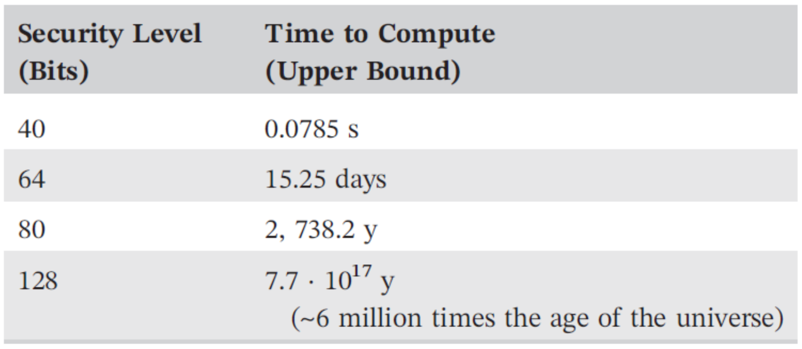
Both symmetric [39] and asymmetric [26] algorithms have been proposed as digital authentication methods for GNSS and SBAS, and both sets of algorithms are built upon primitives that are assumed to be hard to break. For asymmetric algorithms, the primitive discussed here is the digital signature based on the discrete logarithm, and for symmetric algorithms, it is the one-way hash function. Both problems are believed to be hard to break for classical computers, but there already exist algorithms for quantum computers that greatly diminish the security of these problems.

This chapter introduces key cryptographic primitives and how they are utilized to create the two different authentication schemes proposed earlier in this thesis, TESLA and ECDSA/EC-Schnorr. These schemes are scrutinized under the lens of quantum computing, and recommendations are put forth for future iterations of SBAS authentication in the next couple decades.

7.2 Cryptographic Primitives

Cryptographic primitives denote the building blocks used to build cryptographic algorithms. A hash function is a primitive used in many applications within, and outside of, cryptography. Hash functions are deterministic functions that are capable of mapping data of any size to a fixed size. For use in cryptography,the hash functions must be secure, which requires additional definitions. A secure hash function, denoted by h($\cdot$) in this chapter, is assumed to be a one-way function, meaning it is preimage resistant and there is no inverse function, $h^{-1}(\cdot)$, that can be easily computed. Secure hash functions are also assumed to have an infinitesimal probability of collision, where $h(x) = h(y)$ for some $x \neq y$. Secure hash algorithms (SHAs) are algorithms approved by the National Institute of Standards and Technology (NIST) that are assumed to have these traits. However, collisions are known to exist, and recently, Google found such a collision on one of the first SHAs developed by NIST: SHA-1 [62]. Over the years, SHA algorithms have been improved with SHA256 being today's standard in modern cryptographic schemes. SHA256 is a part of the SHA-2 series and provides a fixed 256-bit output that is preimage resistant and collision resistant. These SHA functions are thought to be perfectly preimage resistant and so the security of SHA is bounded by the probability of finding a collision using the birthday attack. Using this attack, it is possible to find a collision on any secure hash function in $\mathcal{O}(2^{n/2})$ time, where n is the preimage security if the attacker has the ability to choose the input. $\mathcal{O}(*)$ in this chapter denotes the limit in order of operations necessary to accomplish a task. The birthday attack is derived from the birthday problem, which asks: “If there are N people a room, what is the probability that at least 2 of them share the same birthday?” This problem is relevant for hashes in that an attacker wants to find any two inputs that hash to the same output, thus creating a collision. The probability curve for the birthday problem is shown in Figure 7.1. In the case of hashes used in TESLA, the birthday attack is not a relevant attack as the attacker does not have the ability to choose the input to the hash function. In this case, the keychain is already established, and the attacker must find the preimage or second preimage of the previous key. Hash algorithms such as SHA256 are said to have a security level of 128 bits as it would require $\sim 2^{128}$ computations to break the scheme using the birthday attack. In the case of hash functions used in GNSS authentication, the birthday attack is not a valid attack and so SHA256 retains its preimage security of 256 bits.

Security level is a means of comparing the strength of cryptographic schemes. A security level of n-bits implies that it would take an adversary on the order of 2^ncomputations before the scheme would be broken. Table 7.1 shows the upper bound on the computation time required to break different security level schemes with a single processor. This processor is based on publicly available bitcoin mining hardware that can perform up to 1.4×10^{13} hash/second [42].



Security Level (Bits)	**Time to Compute (Upper Bound)**
40	0.0785 s
64	15.25 days
80	2, 738.2 y
128	$7.7 \cdot 10^{17}$ y (~6 million times the age of the universe)

Table 7.1: Security level and computation time using 1.4×10^{13} hashes per second

Another cryptographic primitive is the discrete logarithm. This is the problem of solving for the logarithm of an integer over a finite cyclic set (see Figure 7.2) [28]. This definition is general for all cyclic sets and includes the use of elliptic curves as the

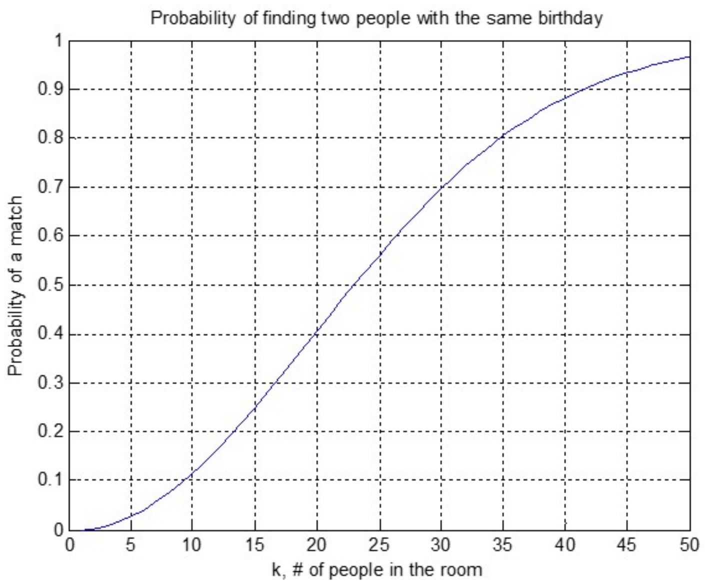
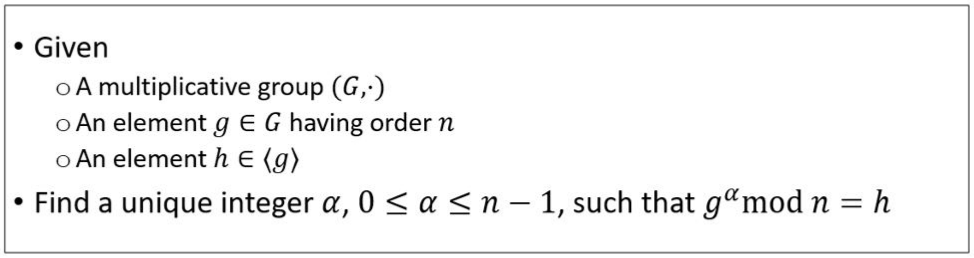


Figure 7.1: Birthday attack

vehicle for discrete logarithms. The most efficient classical algorithm known today for solving the discrete logarithm problem works in subexponential time for classical computers. This algorithm is known as the general number field sieve and works in time $\mathcal{O}(\exp\{c(\log n)^{1/3}(\log\log n)^{2/3}\})$, where c is a constant and "log" is $\log_2$ [63]. If 2^n, the size of the set, is large, then the computation time for solving the discrete logarithm problem becomes infeasible. The discrete logarithm is a powerful primitive because discrete exponentiation can be computed easily with the square and-multiply algorithm but going in the opposite direction is hard for classical computers, making it an effective one-way function. In the case of discrete logarithms on elliptic curves, the best known algorithms are the Pollard's ρ and Shank's baby-step, giant-step method [64]. These algorithms both work in time $\mathcal{O}(\sqrt{n})$.



- Given
  - A multiplicative group $(G,\cdot)$
  - An element $g \in G$ having order n
  - An element $h \in \langle g \rangle$
- Find a unique integer α, $0 \leq \alpha \leq n-1$, such that $g^{\alpha} \bmod n = h$

Figure 7.2: Discrete logarithm problem

7.3 Quantum Computing and Quantum Algorithms

Quantum computers are being developed today in both the public and private sector because of their advantages in computing problems difficult for classical computers. The quantum computer was first theorized by Richard Feynman in 1982 as an inspiration to efficiently simulate quantum phenomena [65]. Quantum computers were shown to be theoretically possible, and since then, many of the design issues have transitioned from the realm of theory to engineering. IBM has been in the development and research of quantum computers for the past 35 years with a program known as IBM Q [66]. D-Wave systems in Canada are developing computers that use quantum annealing and in 2017 sold a 2000 qubit machine known as the D-Wave 2000Q to Temple defense systems [61]. Other well-known companies, such as Google, Microsoft, Intel, and Alibaba, also have projects that are pushing the frontier of quantum computing [67, 68, 69, 70]. Hardware for quantum computers is complex and expensive, but that does not imply that only governments and large corporations will have access to quantum computing. IBM, Google, and several others already offer access for common users to quantum computers using web services. The threat to modern cryptographic architectures is not exclusively controlled by those that own the physical computers.

Quantum computers differ from classical computers in several ways. Classical computers store information in bits that must be in either of two states: 0 or 1. Quantum computers store information in qubits that can exist as 0, 1, or a superposition of both states at one time. Before a qubit is observed, it exists in a probability distribution between 0 and 1, but the moment it is observed, the wave function collapses and the state is read as either 0 or 1. The wave function of a qubit is denoted as $\bar{\Psi}$. The power of superposition and its implications for computing cannot be overstated. Superposition not only allows a qubit to be in multiple states at once but also enables the computation of all possible transitions from the qubit states simultaneously. In the case of a single qubit, the wave function is defined in Equation 7.1, where $\|\cdot\|$ is the L2 norm and $\Psi_0 |0\rangle$ represents an amplitude, Ψ_0, associated with state 0 according to Dirac notation. $|\Psi_0|^2$ and $|\Psi_1|^2$ are the probabilities associated with states 0 and 1, respectively, and the unity of the L2 norm ensures a proper probability distribution. While the square of the amplitudes gives the probability of observing a particular state, the amplitudes themselves are allowed to be negative, which leads to another important trait: quantum algorithms can use constructive and destructive interference of the states to perform algorithms.

$$\begin{aligned} \bar{\Psi} &= \Psi_0|0\rangle + \Psi_1|1\rangle \\ \|\bar{\Psi}\| &= 1 \end{aligned} \tag{7.1}$$

Quantum computers perform Hamiltonian transformations on the qubit wave functions to "move" the quantum particles from state to state. In the theoretical sense, a Hamiltonian is a unitary operation that transforms the state of the qubits. In a physical sense, a Hamiltonian operation on a machine can be thought of as shining a laser on an electron or allowing two electrons to interact. Once the quantum computer has run the necessary amount of Hamiltonian transformations on the system, the state is observed, the wave functions collapse, and the final output of the system is reported. In general, for n qubit machines, the state can be represented by Equation 7.2.

$$\bar{\Psi} = \sum_{w\in\{0,1\}^n} \Psi_w|w\rangle \in \mathbb{C}^{2^n} \tag{7.2}$$

The dimension of this quantum state is 2^n, and observing the system gives output $w \in \{0,1\}^n$ with probability $|\Psi_w|^2$. Algorithms transform the state with a series of Hamiltonians and work by cancelling outputs that are undesirable and amplifying those that are desirable. After applying a series of m Hamiltonians, $\bar{\Psi}_m = H_m \cdots H_2 H_1 \bar{\Psi}_0$, the state is observed, and the answer lies where the largest square of the amplitude (highest probability) rests. However, these Hamiltonian transformations contain noise when processed in physical systems, $\bar{\Psi}_m = \widetilde{H}_m \cdots \widetilde{H}_2 \widetilde{H}_1 \bar{\Psi}_0$ and so quantum error corrections are needed to recover lost information, which is just one example of the complexity of realizing these systems. The rest of this section will introduce two algorithms that have an enormous impact on the cryptographic primitives introduced in the previous sections.

The first is known as Shor's algorithm, which solves the discrete logarithm problem in polynomial time. A comparison of the order of operations of classical versus quantum algorithms against elliptic curve discrete logarithms is seen in Figure 7.3. The advantage of Shor's algorithm is its ability to find the periodicity of functions with the use of the quantum Fourier transform (QFT). This is important since it can be shown that computing the discrete logarithm is equivalent to finding the period of a function (see Figure 7.4). The QFT can be represented as a series of Hamiltonian transformations and requires $\mathcal{O}(n\log n)$ gates to perform the discrete Fourier transform on 2^n amplitudes [71]. The advantage is that the QFT can be computed in exponential dimensional space, whereas the discrete Fourier transform is performed on a vector. This gives quantum computers the ability to solve the discrete logarithm problem in polynomial time [72]. The consequence of this is that any asymmetric cryptographic scheme that utilizes the discrete logarithm problem can be broken in polynomial time, effectively making the method obsolete.

The second algorithm is Grover's algorithm, which can find the inverse of black box functions in less time than classical algorithms [73]. One example of finding the inverse of a black box function is finding the secret key of the encryption scheme shown in Figure 2.2. If a key is n bits, then the worst case (brute force) method of finding this key will take time $\mathcal{O}(2^n)$ using a classical computer. A quantum computer using Grover's algorithm (Figure 7.5) can find that same key in time $\mathcal{O}(\sqrt{2^n})$, effectively halving the assumed security of the cipher. Quantum computers can use this technique to find preimages of hash functions quadratically faster than classical computers, changing the security level of SHA256 from 256-bits to 128 bits.

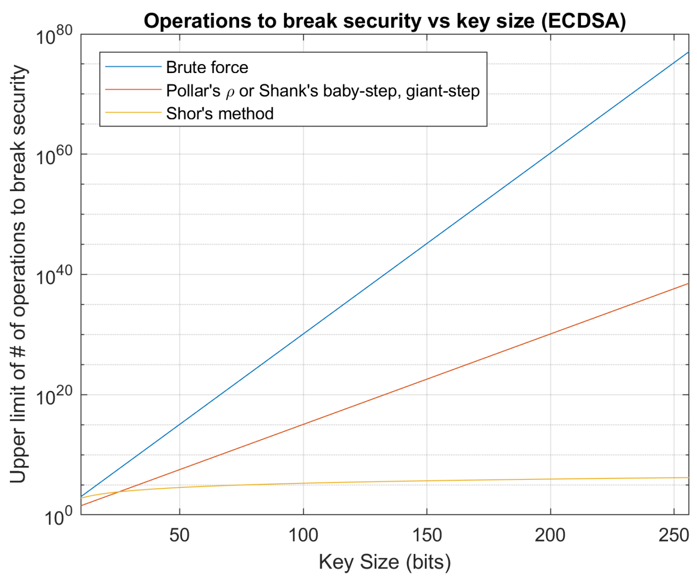


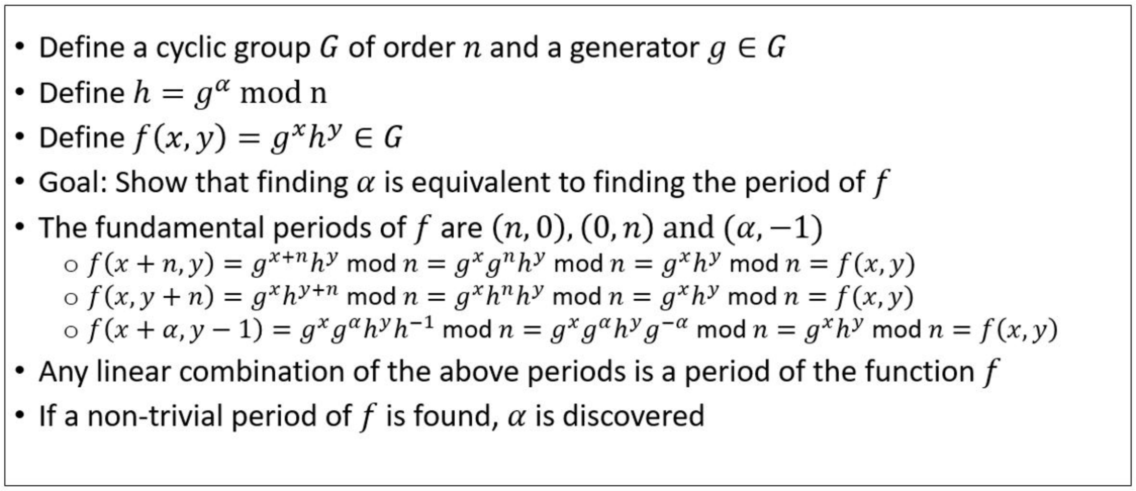
Figure 7.3: Time to break Elliptic Curve Digital Signature Algorithm

This is still a good security level, but there is caution against truncating the SHA256 outputs for use as keys in the TESLA keychain once quantum computers become a reality.

7.4 TESLA and ECDSA/EC-Schnorr Vulnerabilities

In the previous chapters, cryptographic primitives were introduced as well as two schemes built using these primitives. The most recent section presented quantum computing algorithms that fundamentally challenge the security arguments for the primitives that these schemes are built upon, and now, this section will explore how TESLA and ECDSA/EC-Schnorr are vulnerable to quantum computing attacks.

ECDSA and EC-Schnorr derive their security argument from the claim that the discrete logarithm problem is hard. Indeed, a subset of the public keys that are freely given are computed using the secret key as shown in Figure 2.6. If an attacker had access to a quantum computer, either through owning the hardware or by renting time

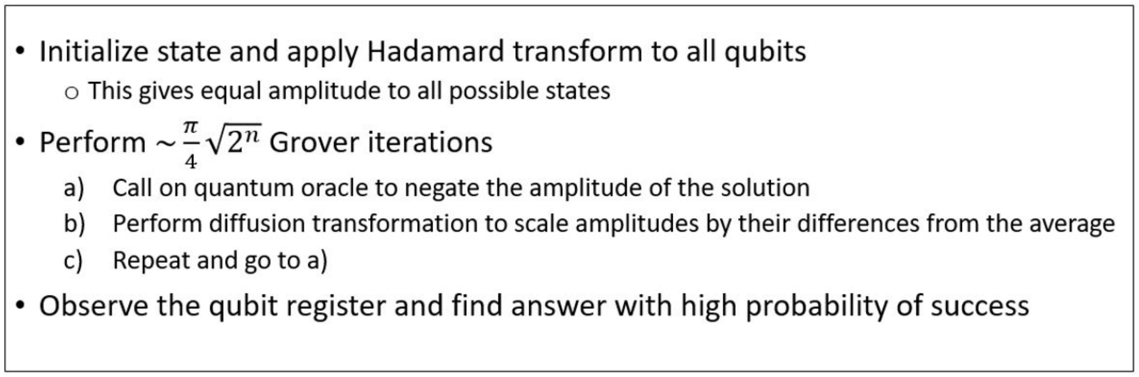


- Define a cyclic group G of order n and a generator $g \in G$
- Define $h = g^{\alpha} \bmod \mathrm{n}$
- Define $f(x, y) = g^x h^y \in G$
- Goal: Show that finding α is equivalent to finding the period of f
- The fundamental periods of f are $(n, 0), (0, n)$ and $(\alpha, -1)$
  - $f(x+n, y) = g^{x+n} h^y \bmod n = g^x g^n h^y \bmod n = g^x h^y \bmod n = f(x, y)$
  - $f(x, y+n) = g^x h^{y+n} \bmod n = g^x h^n h^y \bmod n = g^x h^y \bmod n = f(x, y)$
  - $f(x+\alpha, y-1) = g^x g^{\alpha} h^y h^{-1} \bmod n = g^x g^{\alpha} h^y g^{-\alpha} \bmod n = g^x h^y \bmod n = f(x, y)$
- Any linear combination of the above periods is a period of the function f
- If a non-trivial period of f is found, α is discovered

Figure 7.4: Discrete logarithm as a periodic function

on a quantum platform, the secret key could be discovered in polynomial time using Shor's algorithm. This is true for all cryptographic algorithms that rely on the discrete logarithm problem. Other proposals that suggest the use of discrete logarithm–based signatures, such as those in previous studies [26, 16] are also vulnerable to the same attack. (It is also important to note that RSA and other popular cryptographic schemes are broken using quantum computing. This thesis does not go into detail about them here as they are not relevant to the discussion of GNSS and SBAS signature schemes.) In the case where EC-Schnorr or ECDSA is used as the digital signature algorithm for verifying the keychain used in TESLA, an attacker would use quantum computing to discover the secret key, and then create a false keychain authenticated by a forged digital signature. If the digital signature that signs the root key of the TESLA keychain is quantum-resistant, this attack would be much more difficult to carry out.

The attack on TESLA is slightly more nuanced than the attacks on the discrete logarithm. TESLA uses a symmetric security primitive with the delayed release of keys from an authenticated keychain. The attack is to "discover" the full keychain or a false keychain that collides with the true one through a second preimage, before that full keychain is released, as was presented in chapter 3. With an authenticated keychain, a spoofer would be able to write and sign any message without fear of



- Initialize state and apply Hadamard transform to all qubits
  - This gives equal amplitude to all possible states
- Perform $\sim\frac{\pi}{4}\sqrt{2^n}$ Grover iterations
  a) Call on quantum oracle to negate the amplitude of the solution
  b) Perform diffusion transformation to scale amplitudes by their differences from the average
  c) Repeat and go to a)
- Observe the qubit register and find answer with high probability of success

Figure 7.5: Grover algorithm for inverting black box functions

being discovered since the keys being released would be fully authenticated by the discovered or forged keychain.

The attack proceeds as follows. First a new root key is released that is authenticated by an asymmetric scheme. The TESLA protocol is then carried out in the standard way, releasing keys at a standard pace on the way up the keychain. The attacker's job is to find a keychain that hashes down to the most recent key that has been released. For the following examples, we will assume that the keychain is sufficiently long with N keys derived from a hash function that is susceptible to quantum attacks in order to give the attacker a chance to find the keychain or a keychain that collides through a second preimage. In this case, the attacker finds a keychain that is connected to key k_i. Several possible outcomes are shown for a successful attack on the TESLA keychain in Figure 7.6, where k^* denotes the attacker keychain. In outcome (a), the attacker finds a keychain that collides after key k^*_{i+1} that hashes to key k_i where $k^*_{i+1} \neq k_{i+1}$; in (b), the attacker finds a keychain that collides after key k^*_{j+1} that hashes to key k_j where $k^*_{j+1} \neq k_{j+1}$ and $i < j < N-1$; and, in (c), the attacker finds a keychain that is identical to the true keychain.

All that is known to the attacker is that key k_i has been reproduced by a keychain discovered by the attacker. The attacker has no way of knowing any of the true keys higher up in the keychain that have yet to be released. It is uncertain whether the attacker found the actual chain or a chain based on a second preimage of a key. Assuming that there is a time Δt between each key release, the attacker would need

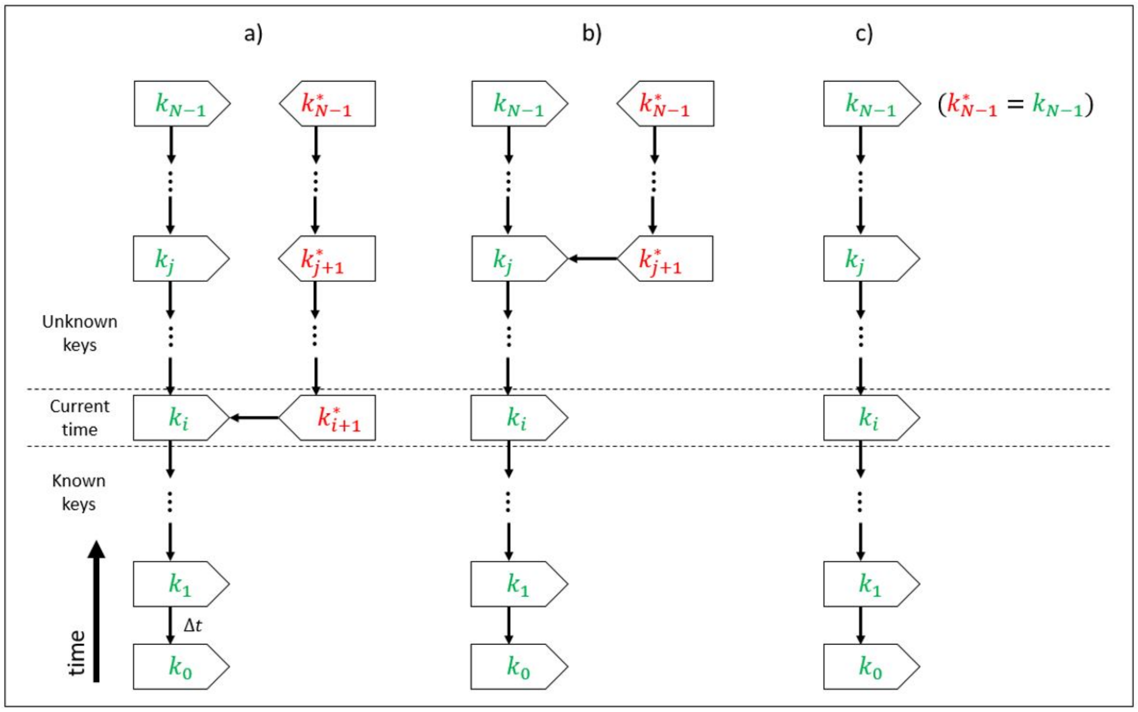


Figure 7.6: Possible outcomes for a successful Timed Efficient Stream Loss-tolerant Algorithm (TESLA) keychain attack

to begin spoofing the receiver within at least time Δt, or at most time $(N-1-i)\Delta t$ of the successful attack depending on where in the keychain the second preimage was found. It would be in the interest of the spoofer to begin spoofing the target receiver immediately, but the attacker has no way of telling if they are in case (a), (b), or (c). The attacker knows they now have control of the keychain, but they do not know if they have the full keychain or a separate one that connects through a second preimage between key k_i and k_{N-1}. There is margin for the spoofer if they accept the risk of a missed spoof attempt and assume they have found a preimage higher up the chain. Another case not mentioned here is the case where the attacker finds a second preimage in the keychain beyond the current epoch. This will not be accepted by receivers and so this does not break the keychain. An efficient attacker would stop computing hashes beyond the current epoch to save computation power and so these second preimages may not be found in that case.

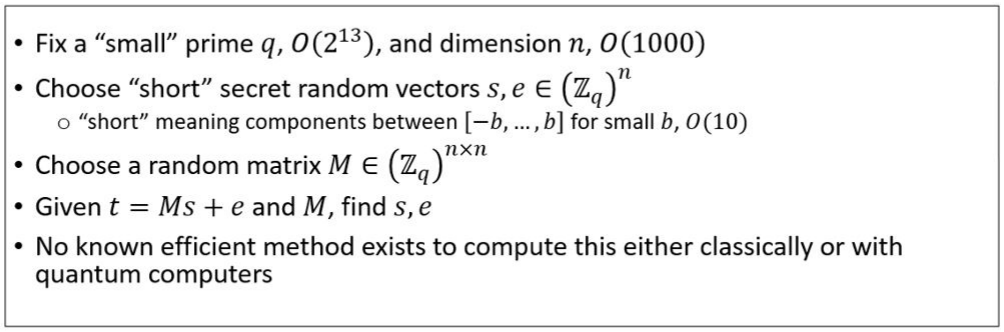
In this exercise it is assumed that the keychain is long enough for an attack to be feasible and the question remains if this is a reasonable assumption. It turns out the length of the keychain is not the most decisive factor in determining the security level. Lengthening the keychain increases the amount of hashes the attacker must perform linearly while also giving the attacker longer amounts of time to find a preimage or second preimage of an element in the keychain. The size of the key and hashing function, however, scales the amount of hashes required to find a preimage exponentially. Assuming the SHA256 hash is used to create the keychain, the security level of the keychain is ~ 256 bits using classical computers. With Grover's algorithm and a quantum computer, the keychain's security is reduced to ~ 128 bits, which is still considered to be very secure. If, however, the keys are truncated to save bandwidth in the authentication message, depending on the extent of the truncation, this attack becomes more feasible. Citing Table 7.1 as a rough estimate for time to find a preimage of an element in the keychain, if a keychain is used for a period of a month or greater and the keys are truncated to be 128 bits or less, it is reasonable to assume that a preimage can be found. A more formal analysis of key and keychain lengths has already been shown in Chapter 3

7.5 Quantum-Resistant Algorithms and Recommendations for SBAS

Today, many companies and organizations that require secure communications are future proofing their security schemes in response to higher amounts of computing power and quantum computers. For symmetric hash–based cryptography, it has been recommended to use the SHA2 or SHA3 series hash functions with bit lengths of 256 bits or more as this is sufficient against preimage attacks [74]. For replacements to key exchange protocols and other cryptographic schemes that use the discrete logarithm problem as the foundation of security, there has been a push within the academic community to develop what are known as postquantum cryptographic functions, which will be secure against quantum computing attacks. This section will give an overview of the field of modern post-quantum cryptography and give recommendations on the replacement of ECDSA/EC-Schnorr as an asymmetric cryptographic scheme for use in SBAS. The schemes presented here are still under scrutiny and post-quantum cryptographic schemes are not expected to be public for several years. NIST is currently in the selection process for these post-quantum solutions [75].

One field within postquantum cryptography that is showing promise is the field of lattice-based cryptography using learning with errors (LWEs). LWE has the advantage of being provably secure and having a (relatively) small public key and signature size. The security behind LWE is found in the difficulty to invert an affine function as seen in Figure 7.7 [76]. A signature scheme that uses LWE named GLYPH is used as an example for signature and public key size. These values are shown in Table 7.2 along with the security level of the scheme.

Another form of postquantum cryptography is multivariate cryptography. This method uses difficult to invert multivariate functions as a basis of security. The signature is verified by solving systems of multivariate polynomial equations, and while the public and private keys are large for this system, the signature size is small with values on the order of those given in Table 7.2. These values are derived from the patented Rainbow Signature Scheme found in a previous study [77].



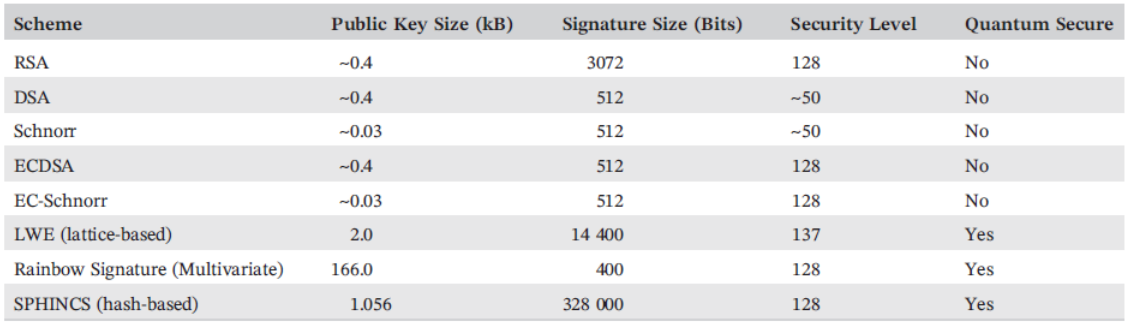
- Fix a "small" prime q, $O(2^{13})$, and dimension n, $O(1000)$
- Choose "short" secret random vectors $s, e \in (\mathbb{Z}_q)^n$
    - "short" meaning components between $[-b, \ldots, b]$ for small b, $O(10)$
- Choose a random matrix $M \in (\mathbb{Z}_q)^{n \times n}$
- Given $t = Ms + e$ and M, find s, e
- No known efficient method exists to compute this either classically or with quantum computers

Figure 7.7: Learning with errors

There are also hash-based digital signatures, such as the SPHINCS signature [78], that are based on the Merkle tree. A limitation with these signatures is that there are only a finite number of messages that can be signed, although this number resides somewhere in the millions. Another disadvantage of this signature scheme is that the signatures are very long and infeasible for any SBAS or GNSS.

Symmetric algorithms, such as TESLA, still provide the best security for a given key size and so it is recommended that TESLA serve as the authentication process for SBAS messages in a post-quantum world.

The asymmetric algorithm for authenticating these keychains will need to have a security level sufficient for the lifetime of the system. It is recommended that SBAS use a scheme with a security level of 128 bits or higher. The three algorithms mentioned above serve as potential replacements for ECDSA/EC-Schnorr to authenticate TESLA keychains. Rainbow Signature in particular stands out as a potential replacement that provides a small signature size comparable to other signature schemes being proposed today. Its major drawback is the public key size, which would make OTAR for this asymmetric scheme infeasible so alternative methods of installing and using public keys would need to be investigated for future revisions of an SBAS authentication service.



Scheme	Public Key Size (kB)	Signature Size (Bits)	Security Level	Quantum Secure
RSA	~0.4	3072	128	No
DSA	~0.4	512	~50	No
Schnorr	~0.03	512	~50	No
ECDSA	~0.4	512	128	No
EC-Schnorr	~0.03	512	128	No
LWE (lattice-based)	2.0	14 400	137	Yes
Rainbow Signature (Multivariate)	166.0	400	128	Yes
SPHINCS (hash-based)	1.056	328 000	128	Yes

Table 7.2: Prequantum and postquantum asymmetric cryptographic schemes

7.6 SBAS Authentication in a post-quantum world

Quantum computers are currently being developed and may someday have the ability to break classical cryptographic schemes used throughout the world. While predictions vary widely for quantum computers that could break cryptographic schemes [79], it is prudent to start thinking of solutions to this problem today and be prepared when the time comes. This thesis offers two examples of primitives, the discrete logarithm and the hash function, that will witness decreased security when quantum computers of sufficient size come online. With low data rates and minimal computational ability in the receiver, SBAS systems should use asymmetric schemes with small signature sizes and low computational power if they wish to defend against these threats. TESLA offers a secure and reasonably sized signature for future revisions of SBAS authentication if it is designed with a sufficiently large key size ($\sim 256 bits$). A first look into postquantum secure algorithms also points to the Rainbow Signature scheme as a lightweight, secure algorithm for keychain authentication. The results of NIST's search for post-quantum secure digital signature algorithms will likely be the largest driver for future SBAS authentication algorithm choices.

Chapter 8

Conclusions and the Road Ahead

8.1 Thesis Contributions

This thesis makes several contributions on the path towards development of an authentication solution for SBAS. This chapter will recap these contributions here and then discuss more about the road ahead and what it will take to eventually make this design a reality.

8.1.1 Security and integrity preserved through message loss

Developed at length in chapter 4, an authentication scheme for Space Based Augmentation Systems (SBAS) was introduced that preserved security and integrity through message loss. Prior to this thesis, there were serious issues with how receivers would deal with messages that the receiver was unable to authenticate. Typical SBAS authentication schemes would bundle groups of messages together and produce a single signature for that batch of messages. The issue here was that if a single message was lost due to a small amount of interference, the whole batch of messages could no longer be authenticated. This created large continuity issues for authenticated SBAS. To mitigate this issue, designs up until this point had stated that messages that could not be authenticated due to a bad message would still be used by the SBAS receiver, even though its authenticity could never be verified. Of course, the problem with this action then, is the fact that an attacker could simply add in a ”bad” message within a batch of spoofed messages and a receiver, not being able to authenticate that batch of messages, would incorporate the spoofed information. To solve the problem of heightened continuity risk due to missed messages, another problem arose whereby an attacker could circumvent the whole authentication process altogether, making SBAS authentication a moot point.

This thesis addressed these two major problems. The first problem of continuity risk due to message loss was alleviated with the design of TESLA-LittleMACs. TESLA-LittleMACs produces an individual Message Authentication Code (MAC) for each and every SBAS message. This way, if a single SBAS message is lost, the data sent in the surrounding messages can still be verified, alleviating the continuity risk introduced through authentication considerably. To tackle the problem of attackers being able to introduce spoofed messages into the SBAS receiver, this thesis produced guidelines for how a receiver should use or not use messages before they have been authenticated. Importantly, these guidelines allowed receivers to preserve integrity in the event that a heightened Dual Frequency Range Error Indicator (DFREI) were broadcast while mitigating risks of Hazardous and Misleading Information (HMI) being introduced by a spoofer. These guidelines also allowed receivers to preserve the Time To Alert (TTA) mandate of the SBAS service.

8.1.2 Public key infrastructure and key management for SBAS authentication

Prior to this thesis, there was only a light touch on how Public Key Infrastructures (PKI) might work for GNSS architectures. Beyond GNSS, there was even less treatment on how a PKI might work for the internationally standardized SBAS.

This thesis provided solutions to 3 major challenges with developing a PKI for a system like SBAS. The first challenge was that the users of the service have no access to external, potentially higher bandwidth, networks. Without access to these external networks, a PKI designs need to figure out a way to securely deliver keys to users. The second challenge was an extension of the first, in that if a provider was to broadcast key updates over the air, it was unclear if this was even possible given the bandwidth that was available. And finally the third challenge was creating a PKI that worked for all parties involved. Each sovereign state is responsible for maintaining and operating their respective SBAS, but all 'SBAS enabled' receivers should be able to use any SBAS service. This thesis addressed the challenge of how the PKI could be created in a way that preserved the sovereignty of these systems as well as the universal standards that allows all receivers to work globally.

Introduced in this thesis is a message set and corresponding delivery mechanism for public keys. This design is able to deliver crucial public keys to users in a secure and timely manner. An analysis was done to prove out the efficacy of this design and showed that for the lowest level of keys, the keys used to verify the data signatures, time on the order of minutes was required to retrieve these keys using the over-the-air-rekeying concept. This concept, which uses minimal bandwidth that is left over in the signature messages, proved up to the task in delivering all levels of keys. Moreover, the full public key infrastructure that was designed allows each service provider to manage their own set of data level signature keys, while no added processes are required for receivers to receive and use new authenticated signals from upcoming service providers. Because level-2 data keys are delivered solely over the air and these keys are verified using universally held level-1 SBAS public keys, an SBAS receiver that is designed to use authentication is still forward compatible to any new service providers that may emerge. Current and next keys are also published for users so that seamless transitions between keys can take place without any impact to authentication availability and continuity. This tiered public key solution also preserves the autonomy and sovereignty of each SBAS service provider to provide their own data authentication service, while maintaining the International Civil Aviation Organization (ICAO) as the collective organization ensuring the standardization of SBAS.

8.1.3 Quantum considerations for SBAS authentication

Quantum computing is poised to change the way many challenging computational problems are addressed. This is particularly true for the field of cryptography. While quantum computers as of 2020 still measure their computing power in 10's of qubits, technologies that reduce qubit errors and enable qubits to be operate at warmer temperatures are being developed all over the world. Modern day cryptography in the form of encryption and authentication can best be described as a collection of very difficult math problems. Quantum computing threatens to make some of these hard problems feasible.

While many have been designing authentication schemes for both GNSS and in particular SBAS for many years, this thesis is the first to take a closer look at the mathematical underpinnings of these schemes through the lens of quantum computing. Two attacks are at the center of these quantum computing attacks: Grover's algorithm and Shor's algorithm. Grover's algorithm is a method of solving the "black box" problem in exponentially faster time than classical computers. This attack has been proven to be an upper bound on quantum computing attacks for this particular problem, but it will require that the security level of all symmetric cryptographic schemes be doubled to maintain the same theoretical security level. Shor's algorithm takes advantage of the cyclical nature of most asymmetric cryptographic problems. By utilizing the quantum fourier transform, Shor proved that quantum computers would make many asymmetric cryptographic schemes obsolete. While Grover makes symmetric cryptography moderately more cumbersome, Shor completely wipes away much that is available today. The National Institute of Standards and Technology (NIST) is exploring symmetric and asymmetric schemes that are post-quantum secure, but these solutions are likely not to be standardized for the next couple years. This thesis was the first to explore attacks on several authentication schemes that are being considered for SBAS today. While the first generation of SBAS authentication algorithms need not be post-quantum resistant, new asymmetric algorithms will likely need to replace current algorithms in future iterations of SBAS authentication designs.

8.2 The road ahead

While this thesis endeavoured to tackle large pieces of how an SBAS authentication system may be designed, there are still many smaller pieces that need to be polished and worked on in future work. This section lists some of those important future steps.

- TESLA keychain and MAC function definitions

  This thesis has covered many of the higher level principles of how the Timed Efficient Stream Loss-tolerant Algorithm (TESLA) can be implemented in the I-Channel to provide authentication of SBAS data. In chapter 3, truncation of the TESLA keys was examined, but the underlying one-way function that creates the TESLA keychain is still yet to be officially defined. While that was not cogent to the analysis presented in this thesis, it will be important to take a look at the computational loads different one-way functions can have in receiver architectures. Cancela et al. [14] has already taken a look at CPU loads for different authentication algorithms and this work can be leveraged to make final design choices for the TESLA keychain one-way function.

  In addition to defining the algorithm to be used for the TESLA keychain, a similar analysis must be done on the specific algorithm to be used to generate the TESLA MACs. While TESLA itself is not standardized, elements that make up TESLA such as the MAC do have some standardized methods and so these can be leveraged when choosing the proper MAC algorithm.

- Loose-time synchronization for TELSA

  As discussed in previous chapters, TESLA requires loose-time synchronization because the MAC algorithm used to verify the data is a symmetric algorithm. Some work has already looked at how to ensure loose-time synchronization in GNSS receivers [13] and methods will need to be put in place specifically for aviation users that allows them to be sure they are within the timing requirements to successfully authenticate SBAS data.

- Higher fidelity simulations

  While the simulations that were carried out in this thesis have been crucial in gaining knowledge of how authentication and key management algorithms perform, higher fidelity simulations will be needed in the future as SBAS authentication gets further in its development. Using a tool such as the Stanford Matlab Algorithm Availability Simulation Tool (MAAST) that is capable of doing quick availability analyses for SBAS services can be beneficial in this case. Currently, the tool is being enhanced to work with individual SBAS messages. Having this type of fidelity available for simulation will help in creating reference implementations for how receivers process SBAS authentication messages. It will also be important in helping to highlight corner cases that may have been overlooked in the course of this work.

- <u>MT63 and Over-The-Air-Rekeying</u>
  The public key infrastructure presented in chapter 5 used only left over bits in the signature messages to deliver Over-The-Air-Rekeying (OTAR) information to receivers. While it was shown that this method can give reasonable performance in delivering keys to users, there exist other bandwidth in the SBAS message stream that may be allocated towards OTAR as well. In particular, the message type 63 messages are currently sent when no information needs to be sent from the SBAS satellites. These messages take up a non-negligible fraction of the bandwidth in both the L1 and L5 broadcast. This bandwidth could be used to deliver OTAR information if they were replaced by a different dedicated OTAR message. One note of caution on relying too heavily on the use of MT63 replacements to deliver OTAR information is that not all service providers have equal amounts of bandwidth available for this type of solution. If an authentication algorithm is to be accepted and standardized by all SBAS providers, it will have to take this into account.

- <u>Procurement of level-1 key services</u>
  In this thesis it was assumed that an entity such as ICAO could take up the mantle of signing level-2 keys used by individual service providers. While it is one thing to presume that ICAO can provide this service, it is another to verify and procure the service itself. As these types of infrastructure can take some

time to properly come online, it would be wise to nail down some of these types of specifics sooner rather than later.

- Future authentication innovations and revisions
  As was covered in depth in chapter 7, quantum computing is on the horizon and it is cause for concern for many of the authentication algorithms available today. As new quantum-resistant algorithms are published by NIST, there will likely be a need in the future to upgrade the authentication capabilities of SBAS services to keep up with the changing landscape. Whether new algorithms that are used are included as a different message type and how these newer algorithms will be phased in should be thought of today to properly design in the ability to revise the system in the future.

- SBAS L1 authentication
  While most of the work presented here as focused on the authentication of L5 SBAS data, many of these same algorithms and concepts can also be used to authenticate L1 data as well. This thesis lays the ground work for how that may be accomplished and integrated as an L1/L5 data authentication scheme.

The work presented in this thesis lays the groundwork for how data authentication can be implemented on operational SBAS services. The design was shown to be backwards compatible with existing receivers, forward compatible to future service providers, and have the ability to be ratified and standardized by all SBAS. As the threats to aviation evolve, SBAS authentication can meet these challenges so that service providers can continue to ensure the safe use of GNSS for the millions of people that fly everyday.

Appendix A

Summary of Key Performance Indicators

This appendix summarizes the Key Performance Indicators (KPIs) used throughout this work. Many of these definitions are paraphrased from Working Paper 36 presented to the Navigation Systems Panel at the fifth joint working groups meeting in Montreal, October 2019.

- Authentication Error Rate (AER)
  The Authentication Error Rate identifies the failure rate of an authentication protocol in non-adversarial conditions. This outcome can be influenced by error receiving the authentication information, limitations of the authentication scheme, and aircraft dynamics. Transmission errors have strong impact on data authentication, since decoding errors for the authentication data can lead to unavailable authentication event due to the miss detection of the data signature.

- Time Between Authentication (TBA)
  The Time Between Authentication identifies the mean time between message authentication events. For rigid message schedulers, this value can be analytically derived discounting for alerts and other message interruption events. An extension of the TBA is the MaxTBA and MinTBA which provide both the maximum and minimum time between authentication events.

- Authentication Latency (AL)
  The Authentication Latency measures the delay that is experienced at user level between the reception of the data to be authenticated and the authentication verification output. Authentication latency can be computed both in ideal and realistic conditions. In particular, the Authentication Latency in realistic conditions depends on the probability of having an authentication failure (failures that may delay the availability of a reliable authentication verification).

- Availability
  The probability that the SBAS service will be available to the user at a given time. This value is typically denoted as a percentage. Authentication will impact this value if users are not allowed to use the service if it is not authenticated.

- Continuity
  Continuity is defined as the probability that the service will remain available during a phase of operation given that the service was initially available at the beginning of said phase. For SBAS approach procedures, the window for computing continuity is 15 seconds. Continuity is typically presented as its complement, continuity risk, which is a value that is typically less than 1×10^{-5} for any 15 seconds.

- Time To Alert (TTA)
  When an integrity event occurs, an alert is sent through the SBAS broadcast to warn users of the event. The requirement for these SBAS services is to deliver these alerts within 6 seconds of an event causing Hazardous and Misleading Information. With authentication, this TTA must still exist for integrity events regardless of the authentication solution employed.

Appendix B

OMT Details for L5Q ECDSA Implementation

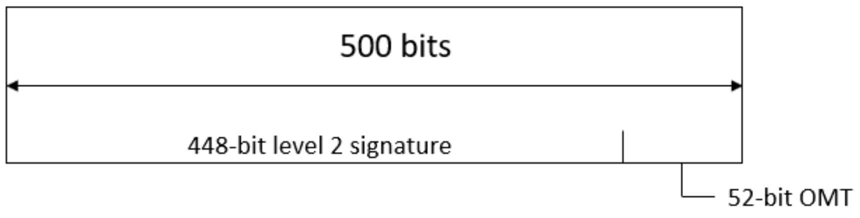
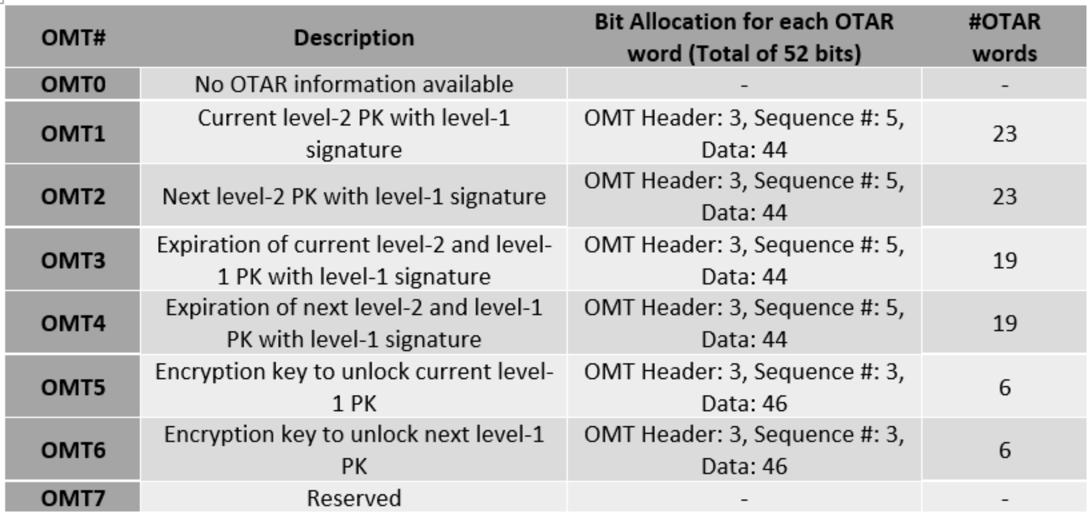


Figure B.1: ECDSA Authentication Message Contents

Notes for Table B.1

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. OMT3 and OMT4 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. OMT3 report the expiration of the level-2 public key first and the level-1 public key second, both concatenated. PK refers to public key.



OMT#	Description	Bit Allocation for each OTAR word (Total of 52 bits)	#OTAR words
OMT0	No OTAR information available	-	-
OMT1	Current level-2 PK with level-1 signature	OMT Header: 3, Sequence #: 5, Data: 44	23
OMT2	Next level-2 PK with level-1 signature	OMT Header: 3, Sequence #: 5, Data: 44	23
OMT3	Expiration of current level-2 and level-1 PK with level-1 signature	OMT Header: 3, Sequence #: 5, Data: 44	19
OMT4	Expiration of next level-2 and level-1 PK with level-1 signature	OMT Header: 3, Sequence #: 5, Data: 44	19
OMT5	Encryption key to unlock current level-1 PK	OMT Header: 3, Sequence #: 3, Data: 46	6
OMT6	Encryption key to unlock next level-1 PK	OMT Header: 3, Sequence #: 3, Data: 46	6
OMT7	Reserved	-	-

Table B.1: OMT description for L5Q ECDSA implementation

Appendix C

OMT Details for L5I Implementation

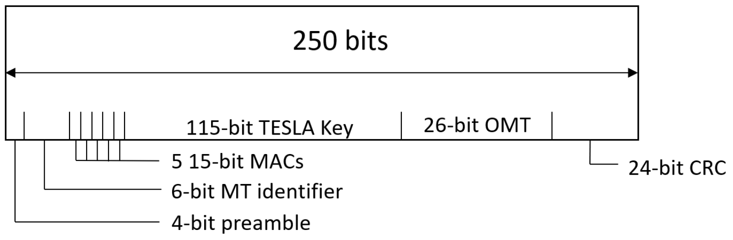


Figure C.1: TESLA - LittleMACs Authentication Message Contents

Notes for Table C.1

All level-1 signatures use the current level-1 key. There are 26 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.

OMT#	Description	Bit Allocation for each OTAR word (Total of 26 bits)	#OTAR words
OMT0	No OTAR information available	-	-
OMT1	Current salt and root/intermediate key for TESLA keychain with level-2 signature	OMT Header: 4, Sequence #: 6, Data: 16	38
OMT2	Current ECDSA level-2 PK with level-1 signature	OMT Header: 4, Sequence #: 6, Data: 16	51
OMT3	Expiration of current salt, TESLA keychain, level-2 and level-1 PK with level-1 signature	OMT Header: 4, Sequence #: 6, Data: 16	43
OMT4	Next salt and root/intermediate key for TESLA keychain with level-2 signature	OMT Header: 4, Sequence #: 5, Data: 17	29
OMT5	Next ECDSA level-2 PK with level-1 signature	OMT Header: 4, Sequence #: 6, Data: 16	51
OMT6	Expiration of next salt, TESLA keychain, level-2 and level-1 PK with level-1 signature	OMT Header: 4, Sequence #: 6, Data: 16	43
OMT 7	Reserved	-	-
OMT8	Encryption key to unlock current level-1 PK	OMT Header: 4, Sequence #: 4, Data: 18	15
OMT9	Encryption key to unlock next level-1 PK	OMT Header: 4, Sequence #: 4, Data: 18	15
OMT10-15	Reserved	-	-

Table C.1: OMT description for L5I TESLA implementation

Appendix D

Selection of Attacks Relating to SBAS Authentication

D.1 TESLA related attacks

- Preimage or second preimage attack on the TESLA keychain

  This topic is developed in great detail in chapter 3. Since the TESLA keychain is derived from a series of one-way functions, there is a threat that if these one-way functions are not strong enough, attackers may be able to discover future keys. If we use the following terminology for the one-way function: $h(x) = y$ where h is a non-invertible one-way function, a preimage refers to an attacker discovering x such that $h(x) = y$. A second preimage refers to an input $\tilde{x}$ such that $h(\tilde{x}) = y$ and $\tilde{x} \neq x$. If the one-way function is non-invertible, an attacker with sufficient computational resources may be able to compute enough keychains through a series of guesses, $\hat{x}$, and hoping to find a preimage or second preimage of a key in the keychain. In preventing this attack, key sizes have an exponential effect on the security of the keychain while shorter keychain lengths also improve security.

- MAC forgery

  This topic is also addressed in chapter 3. The Message Authentication Code

(MAC) is created through the use of a symmetric key. The attack outlined above addresses attacks on discovering the symmetric keys, but MAC forgery attacks cover cases where an attacker might try to publish false information and hope that the MAC addresses it, or they might try to guess their own MAC for the associated data as well. The MAC is typically some form of truncated hash output. The probability that an attacker successfully guesses the correct MAC with a single guess is $P_s = 2^{-n}$, where n is the bit-length of the MAC. Because SBAS is a broadcast service and messages are sent at fixed intervals, an attacker has only one chance to successfully guess a single MAC for an SBAS message or batch of messages. As long as the MAC is of sufficient bit-length to prevent successful MAC forgeries, this attack will be mitigated.

- TESLA keychain precomputation attacks

  Covered in chapter 3, this attack refers to the action of precomputing one-way function output pairs and storing output values in the hope of precomputing a future TESLA keychain. TESLA keychains are created using a set of recursive one-way functions and if these one-way functions are static, meaning they are the same exact function for all time and in all circumstances, an attacker can simply compute large numbers of one-way function chains and cleverly store some of the outputs to then compare later on when a new TESLA keychain is released. In order to avoid this, it is recommended that the constructing one-way function be perturbed in such a way that makes precomputation attacks intractable. This is done by incorporating a ”salt”. In its simplest form, a salt is the incorporation of data in the one-way function that makes it nearly impossible for a keychain to be discovered before the root key is released. A simple salt that is suggested in chapters 3 and 5 is to append each TESLA key with a 30-bit random sequence that is the same throughout the use of the keychain. For example, instead of using the one-way function $h(x) = y$ recursively to create the keys, use $h([x||z]) = y$ where z is only released when the keychain is released. Another method would be to append keys with time-tags and use these to help generate the future keys. Unless the public is aware beforehand when exactly keys will be released, they will not be able to precompute the keychain.

- TESLA timing attacks

  TESLA generates MACs using a symmetric process, meaning the keys that are used to create the MAC are the same keys used to verify the MAC. TESLA creates asymmetry of information by delaying the release of keys such that users trust that only the SBAS provider had access to the secret keys when the MACs were originally generated. This trust in delay ultimately comes down to how much a user trusts their own knowledge of time. Therefore, it is important that users who authenticate SBAS data using TESLA should have a means of ensuring that they have good knowledge of time and that their time is synchronized with the SBAS service provider's time.

- Grover's algorithm applied to the TESLA keychain

  Addressed in chapter 7, quantum computers built at large scales provide challenges to existing cryptographic algorithms. Grover's algorithm is an attack that solves the "blackbox" function. Since one-way function used to generate TESLA keys are thought to be non-invertible, inverting a one-way function is the same as solving the blackbox function. Grover's algorithm solves the blackbox function in a way that halves the security level of the underlying algorithm. For example, if a SHA-256 algorithm is used to generate keys of 256 bits in length and these keys have a preimage security level of 256 bits, Grover's algorithm halves this preimage security level down to 128 bits. This is still fairly strong, but in the case of using a 115-bit TESLA key with the assumptions of Grover's algorithm, the security level is halved to 57.5 bits which is not strong at all. If quantum computers are built to scale and attacks such as Grover's algorithm are realized, the length of the TESLA keys will need to be increased to avoid such attacks.

D.2 ECDSA/EC-Schnorr related attacks

- Shor's attack on ECDSA and EC-Schnorr

  ECDSA and EC-Schnorr are built on the foundation of the discrete logarithm problem (see chapter 7). Peter Shor developed a method of using quantum fourier transforms to take advantage of the cyclical nature of these discrete logarithm problems and invert them, making secret information such as keys discoverable using public keys and signature algorithms. More so, while Grover's algorithm dramatically decreases the security of one-way functions such as SHA256, these functions still retain exponential security while Shor's algorithm more or less breaks all security for ECDSA and EC-Schnorr. If quantum computers are realized to the scale at which they can carry out attacks using Shor's algorithm, all authentication methods that use the discrete logarithm or elliptic curve cryptography will be obsolete. In this event, SBAS service providers will need to use a different asymmetric algorithm to sign the root key of the TESLA keychain or sign other public key information. NIST is in the process of standardizing a new set of post-quantum secure algorithms and when those are available, SBAS service providers may want to design SBAS authentication solutions keeping future revisions of the service in mind.

D.3 Public Key Infrastructure and Key Management related attacks

- Denial of service attacks

  While introducing data authentication for SBAS does provide security of the SBAS data, it also opens up avenues of potential denial of service attacks depending on how authentication is eventually implemented. While there are a number of ways in which an attacker might take advantage of the design of the SBAS authentication methods discussed in this thesis to carry out these types of attacks, denial of service attacks already exist for all SBAS users and there currently exists no protection for users against spoofing.

- Attacks on SBAS ground infrastructure

  WAAS currently has 3 master stations and if all are expected to be able to sign WAAS data, they will all need to be able to share the same secret key information. How this secret information is shared is still being designed, but passing information like this over networks like the internet might open vulnerabilities in the more commonly known domain of internet security. Great care will need to be taken by service providers to protect this secret information and ensure that it does not fall into the wrong hands.

- Secret information compromised through personel:

  One vulnerability with these systems is that secret key information will be accessible in some form to individuals working within ICAO or one of the SBAS service providers. Although this type of attack isn't mentioned often, sometimes vulnerabilities in systems such as these can stem from vulnerabilities in individuals either through bribery, ransom, blackmail, or phishing.

Bibliography

[1] Sohail Hameed and Nitin H Vaidya. Efficient algorithms for scheduling data broadcast. *Wireless Networks*, 5(3):183–193, 1999.

[2] J Volpe. Vulnerability assessment of the transportation infrastructure relying on the global positioning system.–final report. *National Transportation Center*, 2001.

[3] Edwin L. Key. Techniques to counter gps spoofing. Technical report, MITRE, 1995.

[4] Randolph G Hartman. Spoofing detection system for a satellite positioning system, September 17 1996. US Patent 5,557,284.

[5] Nathan Alan White, Peter S Maybeck, and Stewart L DeVilbiss. Detection of interference/jamming and spoofing in a dgps-aided inertial system. *IEEE Transactions on Aerospace and Electronic Systems*, 34(4):1208–1217, 1998.

[6] Per Enge and M Hellman. Authenticating the waas message. presentation to FAA, 1994.

[7] J Rushanan. A gps iii trade study. Technical report, MITRE, 2001.

[8] Logan Scott. Anti-spoofing & authenticated signal architectures for civil navigation systems. In *Proceedings of the 16th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GPS/GNSS 2003)*, pages 1543–1552, 2001.

[9] Markus G Kuhn. An asymmetric security mechanism for navigation signals. In *International Workshop on Information Hiding*, pages 239–252. Springer, 2004.

[10] Oscar Pozzobon, Chris Wullems, Kurt Kubik, et al. Secure tracking using trusted gnss receivers and galileo authentication services. *Positioning*, 1(08), 2004.

[11] Guenter W Hein, Jeremie Godet, Jean-Luc Issler, Jean-Christophe Martin, Rafael Lucas-RodrigueZ, and Tony Pratt. Status of galileo frequency and signal design. In *in CDROM Proc. ION GPS*. Citeseer, 2002.

[12] Christian Wullems, Oscar Pozzobon, and Kurt Kubik. Signal authentication and integrity schemes for next generation global navigation satellite systems. In *Proceedings of the European Navigation Conference GNSS, 2005*, 2005.

[13] Ignacio Fernández Hernández, Todd Walter, Andrew Neish, and C O'Driscoll. Independent time synchronization for resilient gnss receivers. *ION ITM*, 2020.

[14] Simón Cancela, J David Calle, and Ignacio Fernández-Hernández. Cpu consumption analysis of tesla-based navigation message authentication. In *2019 European Navigation Conference (ENC)*, pages 1–6. IEEE, 2019.

[15] I Fernández-Hernández. Gnss authentication: Design parameters and service concepts. In *Proceedings of the European Navigation Conference*, 2014.

[16] Kyle Wesson, Mark Rothlisberger, and Todd Humphreys. Practical cryptographic civil gps signal authentication. *NAVIGATION: Journal of the Institute of Navigation*, 59(3):177–193, 2012.

[17] DH Arze Pando and D Horacio. Distance-decreasing attack in global navigation satellite system. *Mini-Project Final Rep.*, 2009.

[18] Todd E Humphreys. Detection strategy for cryptographic gnss anti-spoofing. *IEEE Transactions on Aerospace and Electronic Systems*, 49(2):1073–1090, 2013.

[19] Andrew J Kerns, Kyle D Wesson, and Todd E Humphreys. A blueprint for civil gps navigation message authentication. In *2014 IEEE/ION Position, Location and Navigation Symposium-PLANS 2014*, pages 262–269. IEEE, 2014.

[20] Sherman C Lo and Per K Enge. Authenticating aviation augmentation system broadcasts. In *IEEE/ION Position, Location and Navigation Symposium*, pages 708–717. IEEE, 2010.

[21] Andrea Dalla Chiara, Giacomo Da Broi, Oscar Pozzobon, Silvia Sturaro, Gianluca Caparra, Nicola Laurenti, Javier Fidalgo, Miguel Odriozola, J Caro Ramon, Ignacio Fernandez-Hernandez, et al. Authentication concepts for satellite-based augmentation systems. In *ION GNSS*, pages 3208–3221, 2016.

[22] Peter Gutierrez. Galileo to transmit open service authentication. *Inside GNSS*, 2020.

[23] Dee Ann Divis. New chimera signal enhancement could spoof-proof gps receivers. *Inside GNSS*, 2019.

[24] Andrew Neish and Ken Alexander. Authenticate-then-use technique with associated receiver states (gswg/3-wp/15). Technical report, ICAO NSP GSWG/3, 2020.

[25] *Galileo Navigation Message Authentication Specification for Signal-In-Space Testing - v1.0*, November 2016.

[26] Jon M Anderson, Katherine L Carroll, Nathan P DeVilbiss, James T Gillis, Joanna C Hinks, Brady W O'Hanlon, Joseph J Rushanan, Logan Scott, and Renee A Yazdi. Chips-message robust authentication (chimera) for gps civilian signals. In *ION GNSS*, pages 2388–2416, 2017.

[27] *Air Force Research Laboratory Space Vehicles Directorate Advanced GPS Technology, Interface Specification, Chips Message Robust Authentication (Chimera) Enhancement for the L1C Signal: Space Segment/User Segment Interface, IS-AGT-100*, April 2019.

[28] Douglas Robert Stinson and Maura Paterson. *Cryptography: theory and practice.* CRC press, 2018.

[29] Jonathan Katz and Yehuda Lindell. *Introduction to modern cryptography.* CRC press, 2014.

[30] RTCA. Waas minimum operational performance standards (mops) (rtca/do-229e), 2016.

[31] The European Organization for Civil Aviation Equipment. Minimum operational performance standard for galileo / global positioning system / satellite-based augmentation system airborne equipment (mops) (ed-259a), 2020.

[32] Department of Defense, Department of Homeland Security, and Department of Transportation. Federal radionavigation plan, 2008.

[33] Todd Walter, Karl Shallberg, Eric Altshuler, William Wanner, Chris Harris, and Robert Stimmler. Waas at 15. *Navigation*, 65(4):581–600, 2018.

[34] Sherman Lo, David De Lorenzo, Per Enge, Dennis Akos, and Paul Bradley. Signal authentication: A secure civil gnss for today. *inside GNSS*, 4(5):30–39, 2009.

[35] Adrian Perrig, Dawn Song, Ran Canetti, JD Tygar, and Bob Briscoe. Timed efficient stream loss-tolerant authentication (tesla): Multicast source authentication transform introduction. *Request For Comments*, 4082, 2005.

[36] Q Dang. Sp 800-107 (rev. 1). recommendation for applications using approved hash algorithms. Technical report, Technical report, Gaithersburg, MD, United States, 2012.

[37] Claus-Peter Schnorr. Efficient signature generation by smart cards. *Journal of cryptology*, 4(3):161–174, 1991.

[38] Adrian Perrig, Ran Canetti, J Doug Tygar, and Dawn Song. Efficient authentication and signing of multicast streams over lossy channels. In *Proceeding 2000 IEEE Symposium on Security and Privacy. S&P 2000*, pages 56–73. IEEE, 2000.

[39] Gianluca Caparra, Silvia Sturaro, Nicola Laurenti, and Christian Wullems. Evaluating the security of one-way key chains in tesla-based gnss navigation message authentication schemes. In *2016 International Conference on Localization and GNSS (ICL-GNSS)*, pages 1–6. IEEE, 2016.

[40] Ignacio Fernández-Hernández, Vincent Rijmen, Gonzalo Seco-Granados, Javier Simon, Irma Rodríguez, and J David Calle. A navigation message authentication proposal for the galileo open service. *Navigation: Journal of the Institute of Navigation*, 63(1):85–102, 2016.

[41] Joon Ian Wong and Johnny Simon. Photos: Inside one of the world's largest bitcoin mines. *Quartz, August*, 17, 2017.

[42] https://www.bitmain.com/. Bitmain. Online, 2018.

[43] James T Curran, Matteo Paonni, and James Bishop. Securing the open-service: A candidate navigation message authentication scheme for galileo e1 os. In *European Navigation Conference,(ENC-GNSS)*, 2014.

[44] Paul Walker, Vincent Rijmen, Ignacio Fernandez-Hernandez, J Simón, D Calle, O Pozzobon, and G Seco-Granados. Galileo open service authentication: a complete service design and provision analysis. In *Proceedings of the 28th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2015)*, pages 3383–3396, 2015.

[45] Ignacio Fernández Hernández, Vincent Rijmen, Gonzalo Seco Granados, Javier Simón, Irma Rodríguez, and J David Calle. Design drivers, solutions and robustness assessment of navigation message authentication for the galileo open service. In *Proceedings of the 27th international technical meeting of the satellite division of the institute of navigation (ION GNSS 2014)*, pages 2810–2827, 2014.

[46] NIST. Fips pub 186-4: Digital signature standard (dss). Technical report, NIST, 2013.

[47] Gregory Neven, Nigel P Smart, and Bogdan Warinschi. Hash function requirements for schnorr signatures. *Journal of Mathematical Cryptology*, 3(1):69–87, 2009.

[48] Andrew Neish, Todd Walter, and J David Powell. Design and analysis of a public key infrastructure for sbas data authentication. *Navigation*, 66(4):831–844, 2019.

[49] Richard Fuller, Todd Walter, and Per Enge. Burst mode message loss effects on waas.

[50] James T Curran and Matteo Paonni. Securing gnss: An end-to-end feasibility study for the galileo open service. In *International Technical Meeting of the Satellite Division of The Institute of Navigation, ION GNSS*, pages 1–15, 2014.

[51] Ignacio FERNANDEZ HERNANDEZ. Method and system to optimise the authentication of radionavigation signals, February 2 2017. US Patent App. 15/302,356.

[52] Gianluca Caparra, Silvia Sturaro, Nicola Laurenti, Christian Wullems, and Rigas T Ioannides. A novel navigation message authentication scheme for gnss open service. In *ION GNSS*, volume 2016, 2016.

[53] Ignacio Fernández-Hernández, Eric Châtre, Andrea Dalla Chiara, Giacomo Da Broi, Oscar Pozzobon, Javier Fidalgo, Miguel Odriozola, Ginés Moreno, Silvia Sturaro, Gianluca Caparra, et al. Impact analysis of sbas authentication. *Navigation*, 65(4):517–532, 2018.

[54] Gianluca Caparra, Silvia Ceccato, Silvia Sturaro, and Nicola Laurenti. A key management architecture for gnss open service navigation message authentication. In *2017 European Navigation Conference (ENC)*, pages 287–297. IEEE, 2017.

[55] ITU. Itu-t standard x.509, 2016.

[56] NIST-FIPS Standard. Announcing the advanced encryption standard (aes). *Federal Information Processing Standards Publication*, 197(1-51):3–3, 2001.

[57] Morris Dworkin. Recommendation for block cipher modes of operation. methods and techniques. Technical report, National Inst of Standards and Technology Gaithersburg MD Computer security Div, 2001.

[58] R Housley. 5652," cryptographic message syntax (cms), 2009.

[59] Elaine Barker and Quynh Dang. Nist special publication 800-57 part 1, revision 4. *NIST, Tech. Rep*, 2016.

[60] Andrew Neish, Todd Walter, and J David Powell. Sbas data authentication: A concept of operations. In *Proceedings of the 32nd International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2019)*, 2019.

[61] https://www.dwavesys.com/. Online, 2020.

[62] Marc Stevens, Elie Bursztein, Pierre Karpman, Ange Albertini, and Yarik Markov. The first collision for full sha-1. In *Annual International Cryptology Conference*, pages 570–596. Springer, 2017.

[63] Jintai Ding and BY Yang. Post-quantum cryptography. *Multivariate public-key cryptography. Berlin, Heidelberg: Springer. doi*, 10:978–3, 2009.

[64] Christof Paar and Jan Pelzl. *Understanding cryptography: a textbook for students and practitioners.* Springer Science & Business Media, 2009.

[65] Richard P Feynman. Simulating physics with computers. *Int. J. Theor. Phys*, 21(6/7), 1999.

[66] IBM. https://www.research.ibm.com/ibm-q/. Online, 2020.

[67] M Reynolds. Google on track for quantum computer break-through by end of 2017. New Scientist, 2017.

[68] M Branscombe. Why microsoft believes we're on the threshold of quantum computing. TechRadar, 2016.

[69] T Simonite. Intel bets it can turn everyday silicon into quantum computing's wonder material. Technology Review, 2016.

[70] Alibaba Group. Aliyun and chinese academy of sciences sign mou for quantum computing laboratory, 2015.

[71] Lisa Hales and Sean Hallgren. An improved quantum fourier transform algorithm and applications. In *Proceedings 41st Annual Symposium on Foundations of Computer Science*, pages 515–525. IEEE, 2000.

[72] Peter W Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM review*, 41(2):303–332, 1999.

[73] Lov K Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 212–219, 1996.

[74] Matthew Amy, Olivia Di Matteo, Vlad Gheorghiu, Michele Mosca, Alex Parent, and John Schanck. Estimating the cost of generic quantum pre-image attacks on sha-2 and sha-3. In *International Conference on Selected Areas in Cryptography*, pages 317–337. Springer, 2016.

[75] Gorjan Alagic, Gorjan Alagic, Jacob Alperin-Sheriff, Daniel Apon, David Cooper, Quynh Dang, Yi-Kai Liu, Carl Miller, Dustin Moody, Rene Peralta, et al. *Status report on the first round of the NIST post-quantum cryptography standardization process.* US Department of Commerce, National Institute of Standards and Technology, 2019.

[76] Arjun Chopra. Glyph: A new insantiation of the glp digital signature scheme. *IACR Cryptology ePrint Archive*, 2017:766, 2017.

[77] Albrecht Petzoldt, Stanislav Bulygin, and Johannes A Buchmann. Selecting parameters for the rainbow signature scheme-extended version-. *IACR Cryptology ePrint Archive*, 2010:437, 2010.

[78] Daniel J Bernstein, Daira Hopwood, Andreas Hülsing, Tanja Lange, Ruben Niederhagen, Louiza Papachristodoulou, Michael Schneider, Peter Schwabe, and Zooko Wilcox-O'Hearn. Sphincs: practical stateless hash-based signatures. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 368–397. Springer, 2015.

[79] Emily Grumbling and Mark Horowitz. Quantum computing: Progress and prospects, 2019.