Quantum Resistant Authentication Algorithms for Satellite-Based Augmentation Systems

Andrew Neish, Todd Walter, Per Enge
Stanford University

Abstract

Cryptography in the form of digital signatures can be a part of the solution to the threat of spoofing and is going to be implemented on Galileo and other Global Navigation Satellite Systems (GNSS). Digital signatures incorporated into the data stream authenticate the integrity of the data as well as the origin of the message. A multitude of signature algorithms already exist. Most are designed for use over the internet where there are fast data rates, high computing power and the option for call and response. Implementing a digital signature on a system such as a Satellite-Based Augmentation System (SBAS) for use in aviation will require the signature to be short, one-way and secure for the next 30 or more years. With the advent of quantum computing, many state-of-the-art authentication schemes are no longer viable, so an authentication scheme implemented in SBAS will need to be quantum secure. This paper introduces the cryptographic primitives (foundational problems) necessary to understand the vulnerabilities in modern day cryptography due to quantum computing and investigates the use of TESLA (Timed Efficient Stream Loss-Tolerant Authentication) and EC-Schnorr algorithms in broadcast systems. A brief introduction to quantum computing and how it will change the field of cryptography is presented followed by attacks on the aforementioned authentication schemes. State of the art quantum resistant authentication algorithms are introduced and compared with the earlier classical cryptographic methods. Finally, recommendations are put forward for the selection and implementation of authentication schemes for SBAS that withstand the threat of quantum computing.

Introduction

Satellite-based augmentation systems have been tremendously successful in enhancing the accuracy, integrity, and availability of global navigation satellite systems. There are, however, exploitable weaknesses in the broadcast design that render these SBAS signals vulnerable to malicious attacks. The signals received on the ground arrive at very low power levels making them susceptible to interference and they are also broadcast unencrypted leaving them open to manipulation and forgery. The GNSS community has been developing strategies for mitigating spoofing attacks and one of these strategies is to authenticate the data broadcast by GNSS and SBAS satellites. While data authentication does not prevent all possible attacks, it does bring resiliency to these systems against false signal generation. Data authentication uses cryptographic techniques to create digital signatures that are appended to the SBAS messages. These signatures not only authenticate the origin of the message generation, but they also authenticate the content of the message as well.

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 [1]. Data authentication algorithms designed for SBAS will need to be secure for at least 30 years, so it is imperative that all potential risks to these algorithms are addressed.

Both symmetric [2] and asymmetric [3] 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. For asymmetric algorithms, the primitive is the discrete logarithm and for symmetric algorithms it is the collision of a secure hash function. Both problems are believed to be hard for classical computers, but there already exist algorithms for quantum computers that greatly diminish the security of these primitives.

This work aims to be both pedagogical for those new to cryptography as well as informative for those already intimate with the design of GNSS authentication schemes. It introduces key cryptographic primitives and how they are utilized to create two different authentication schemes proposed for GNSS. These schemes are then scrutinized under the lens of quantum computing and recommendations are put forth for the design of future SBAS authentication algorithms based on the findings from this research.

Cryptography primitives

Cryptographic primitives denote the building blocks used to build cryptographic algorithms. The following section introduces primitives from cryptography that will be used throughout this paper.

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 paper, is assumed to be a one-way function, meaning they are pre-image 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 $H(x) = H(y) \text{ for some } x \neq y$. Secure Hash Algorithm (SHA) 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 secure hash algorithms developed by NIST: SHA-1 [4]. Over the years, SHA algorithms have been improved with SHA256 being todays standard in modern cryptographic schemes. SHA256 is a part of the SHA-2 series and provides a fixed 256-bit output that is pre-image resistant and collision resistant. These SHA functions are thought to be perfectly pre-image resistant and so the security of SHA is bounded by the probability of finding a collision. There is a classic, attack known as the birthday attack, where it is possible to find a collision on any secure hash function in $O\left(2^{n/2}\right)$ time where n is the pre-image security. This means for SHA256 that it would require $\sim 2^{128}$ computations to break the scheme, which is another way of saying it has a security level of 128-bits. (See Errata)

Security level is a means of comparing the strength of cryptographic schemes. A security level of m-bits implies that it would take an adversary on the order of 2^m computations before the scheme would be

broken. Table 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 $1.4 \cdot 10^{13}$ hash/sec [5].

Table 1: Security Level and Computation time using $1.4 \cdot 10^{13}$ hashes per second

Security Level (bits)	Time to compute (upper bound)
40	0.0785 seconds
64	15.25 days
80	2,738.2 years
128	$7.7 \cdot 10^{17}$ years (~6 million times the age of the universe)

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 1) [6]. This definition is general for all cyclic sets and includes the use of elliptic curves as the vehicle for discrete logarithms. The most efficient classical algorithm known today for solving the discrete logarithm problem works in sub-exponential time for classical computers. This algorithm is known as the general number field sieve and works in time $O\left(\exp\left\{c(\log n)^{1/3}(\log\log n)^{2/3}\right\}\right)$ where c is a constant and "log" is $\log_2$ [7]. If 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.

- 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 1: Discrete Logarithm Problem

Cryptographic algorithms are often used to encrypt and authenticate information and can generally be split into to two categories: symmetric and asymmetric. Symmetric algorithms are most often used in encryption, where both users have what is referred to as a "secret key" to both encrypt and decrypt a message. Only those with the secret key can decrypt the message, ensuring that any eavesdroppers listening to the communications will not learn any information. An example of this is the sending of a transaction from a bank customer to their bank. Asymmetric cryptography, on the other hand, involves the creation of a set of two keys: a "secret key" and a "public key". In contrast to symmetric algorithms, asymmetric algorithms do not encrypt the information, but instead ensure its integrity. The entity with the secret key can broadcast plaintext messages in the open and sign them with a digital signature using their secret key and a known signing algorithm, s. A digital signature is a bit string that is capable of both confirming the integrity of the message as well as the origin of its generation. Anyone who has the public key can verify the digital signature and ensure the integrity of the message.

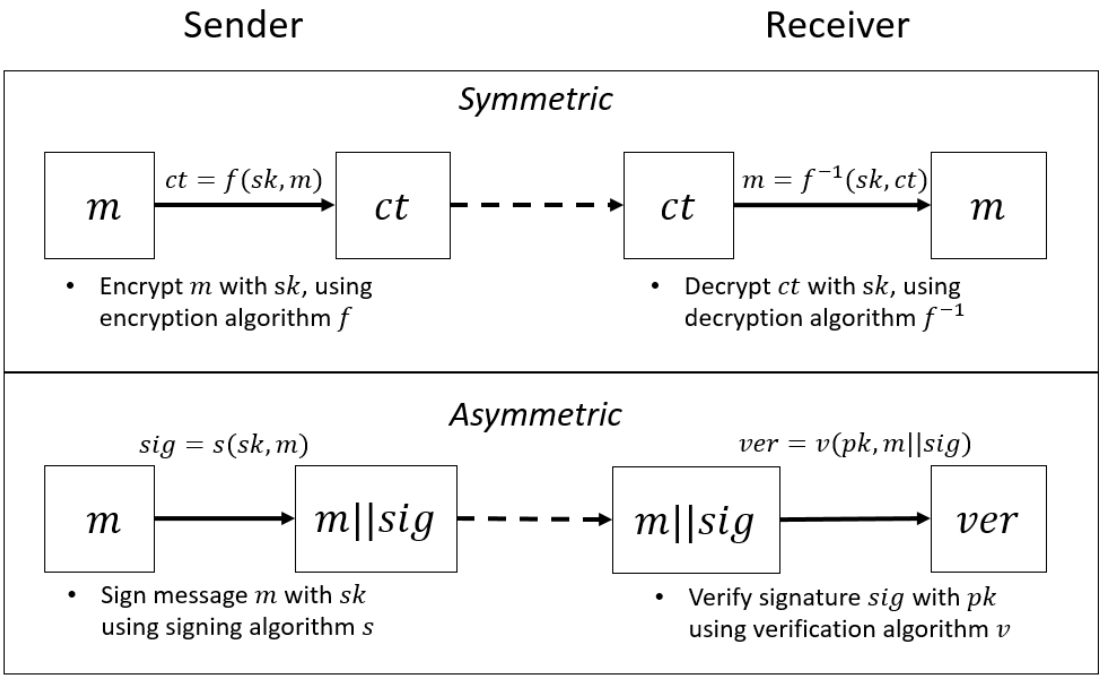

Figure 2: Symmetric and Asymmetric cryptography example

An example of this is the initial contact between a bank customer and their bank. In a simplified world, when the web browser is connecting to the bank's web address, the bank sends a message and signs it with its secret key. The customer can verify that they are talking to the bank by verifying the signature with the public key before setting up an encrypted line of communication. Only those with the secret key can create a digital signature and anyone with the public key can verify its authenticity. Figure 2 depicts an example of encryption and authentication.

Now that the necessary cryptographic primitives have been introduced, the asymmetric scheme candidates being considered for SBAS and GNSS authentication are presented.

TESLA and EC-Schnorr for SBAS Authentication

Recently, several papers have been published in the GNSS literature outlining different methods to facilitate data authentication [8] [9] [10]. Many of these papers have focused on the implementations of these cryptographic systems on SBAS, GPS and Galileo, but have not presented the cryptographic workings of the schemes themselves. This paper will present both symmetric and asymmetric cryptographic schemes and several quantum algorithm attacks that they are vulnerable to. TESLA will be used as a central example because of its prevalence in the GNSS literature on authentication. Another trait that makes TESLA pedagogically advantageous is that even though the protocol itself is a symmetric scheme, the suggested over the air rekeying (OTAR) method involves an asymmetric protocol, EC-Schnorr [10]. The rest of this section outlines the origins and design of TESLA and EC-Schnorr.

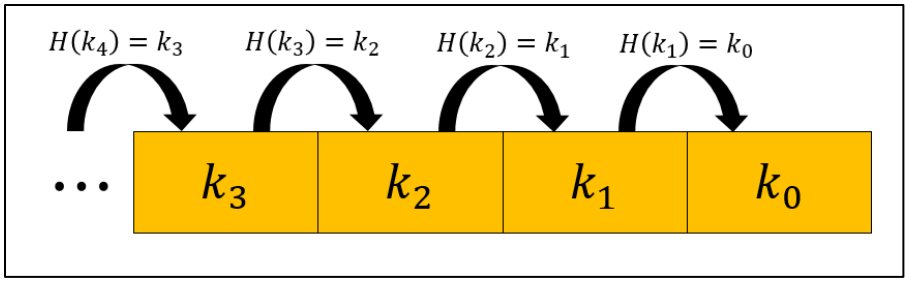

Figure 3: Keychain for TESLA

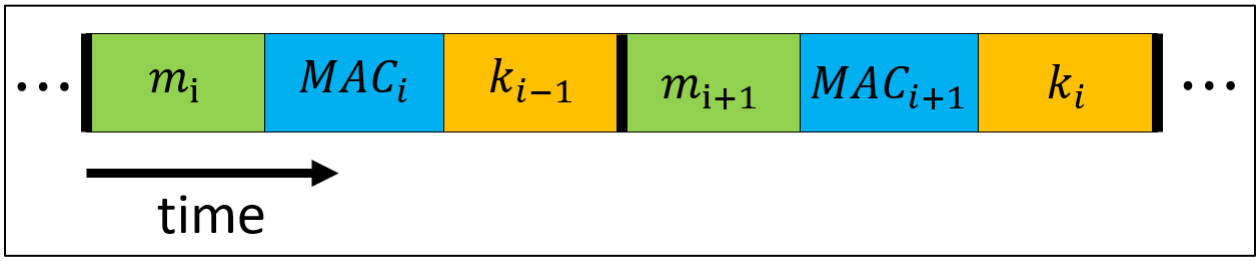

Figure 4: Basic Authenticated Message Sequence

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 5: TESLA Algorithm Implementation Outline

TESLA was developed as an integrity and authentication method for data packets in multicast and broadcast data streams [11]. 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 "key chain" which is a sequence of keys that can be reproduced with a series of one-way functions, typically secure hash functions. Since the key chain is constructed using a one-way function and these keys are released in the order opposite which they are created, only previous keys can be reproduced with the most recent key, and no future keys can be discerned from the key chain. Figure 3 shows an example key chain and

Figure 4-Figure 5 provide an outline for how TESLA is implemented. The details concerning the creation of the key chain, message sequence and receiver implementation have been covered extensively in the literature and are simplified here for the sake of understanding the vulnerabilities of these cryptographic schemes.

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, $H(\cdot)$, 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 key chain is recovered, allowing the receiver to regain message authentication. For the purposes of this article, $H(\cdot)$ will be assumed to be the hash function SHA256. 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 security level of at most 128-bits [12].

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 were to be spoofed into believing 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 $(m^*_{i-1} \neq m_{i-1})$ 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 collision of the hashed keychain, the spoofer could then use their false keychain to authenticate different messages to provide a false position, velocity, and time (PVT) solution.

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. It has been proposed that an alternative asymmetric algorithm, EC-Schnorr, be used to authenticate the root key while leaving the authentication of the message sequences to TESLA. EC-Schnorr is a provably secure algorithm that has 128-bit level security with as little as 512-bits transmitted in its signature [13], 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.

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 introduced in Figure 1. They are, however, beyond the scope of this paper and the reader is pointed to [6] for details of their mechanics. For the sake of simplicity, the Schnorr algorithm is presented below without the use of elliptic curves. The conclusions found in this paper due to quantum computing are the same for both Schnorr and EC-Schnorr.

- **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 $\gamma = H(m \,||\, \alpha^k \bmod p)$ and $\delta = k + a\gamma \bmod q$
  - The signature is (γ, δ) for message m using k
- Verification algorithm for message m
  - Compute $y = H(m \,||\, \alpha^{\delta}\beta^{-\gamma} \bmod p)$
  - If $y = \gamma$, then the message is authenticated

Figure 6: Schnorr signing and verification algorithm

Figure 6 above 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 operators and the public key is known to all users. This public key is either hardcoded into the receiver, or has been received by a trusted source and encoded using either OTAR through the SBAS channel or through some other trusted network.

Since the discrete logarithm problem is assumed to be a hard problem, the algorithm is assumed to be asymmetric. Unlike TELSA, Schnorr does not require loose time synchronization, but the modular exponentiations, or modular elliptic curve adding 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 6 that if the discrete logarithm problem were not hard, one would only need to compute a from β and α, which are both publicly known parameters, to break the integrity of Schnorr.

All assumptions of security that have been presented so far have been made in the classical world, excluding quantum algorithms. In the next few sections, an overview of quantum computing is presented along with the status of contemporary machines known to the public. Algorithms will then be presented that break the hardness of the discrete logarithm problem and reduce the computation time required to find collisions of hash functions.

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 [14]. 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 [15]. 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 [1]. Other well-known companies, such as Google, Microsoft, Intel, and Alibaba also have projects that are pushing the frontier of quantum computing [16] [17] [18] [19]. 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 are 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's 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 understated. 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 1, where $||\cdot||$ is the L_2 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 state 0 and 1, respectively and the unity of the L_2 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.

$$\bar{\Psi} = \Psi_0|0\rangle + \Psi_1|1\rangle \qquad 1$$
$$\left||\bar{\Psi}|\right| = 1$$

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 2.

$$\bar{\Psi} = \sum_{w\in\{0,1\}^n} \Psi(w)|w\rangle \in \mathbb{C}^{2^n} \qquad 2$$

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, $\overline{\Psi}_m = \widetilde{H}_m \cdots \widetilde{H}_2 \widetilde{H}_1 \overline{\Psi}_0$, and so quantum error corrections are needed to recover lost information which is just one example of the complexity of realizing these systems. 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. The most efficient algorithm for computing the discrete logarithm in classical computers runs in sub-exponential time, but Shor's algorithm runs in time $O\big((\log n)^2(\log\log n)(\log\log\log n)\big)$ [20]. 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). The QFT can be represented as a series of Hamiltonian transformations and requires $O(n \log n)$ gates, where n is the number of qubits. The advantage is that the QFT can be computed in exponential dimensional space, giving quantum computers the ability to solve the discrete logarithm problem in polynomial time [21]. The consequences 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.

- Define a cyclic group G of order n and a generator $g \in G$
- Define $h = g^{\alpha} \bmod n$
- 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: Discrete Logarithm as a periodic function

The second algorithm is Grover's algorithm which can find the inverse of black box functions in less time than classical algorithms [22]. One example of finding the inverse of a black box function is finding the secret key of the encryption scheme shown in Figure 2. If a key is n bits, then the worst case (brute force) method of finding this key will take time $O(2^n)$ using a classical computer. A quantum computer using Grover's algorithm (Figure 8) can find that same key in time $O\big(\sqrt{2^n}\big)$, effectively halving the assumed security of the cipher. Quantum computers can use this technique to find collisions on hash functions quadratically faster than classical computers, changing the security level of SHA256 from 128-bits to 64-bits. It has been advised by the NSA to use SHA384 for protection levels up to top secret to ensure security against future quantum computer attacks [23].

- 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 8: Grover Algorithm for inverting black box functions

TESLA and EC-Schnorr vulnerabilities

In the previous sections, 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 on and now this section will explore how TESLA and EC-Schnorr are vulnerable to quantum computing attacks.

EC-Schnorr derives its 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 6. If an attacker had access to a quantum computer, either through owning the hardware or by renting time 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 [3] and [24], are also vulnerable to the same attack. In the case where EC-Schnorr 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 EC-Schnorr signature.

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, before that full keychain is released. With an authenticated keychain a spoofer would be able to write and sign any message without fear of being discovered since the keys being released would be fully authenticated by the 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 attackers 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 a collision. In this case, the attacker finds a collision after key k_i is released. Several possible outcomes are shown for a successful attack on the TESLA keychain in Figure 9.

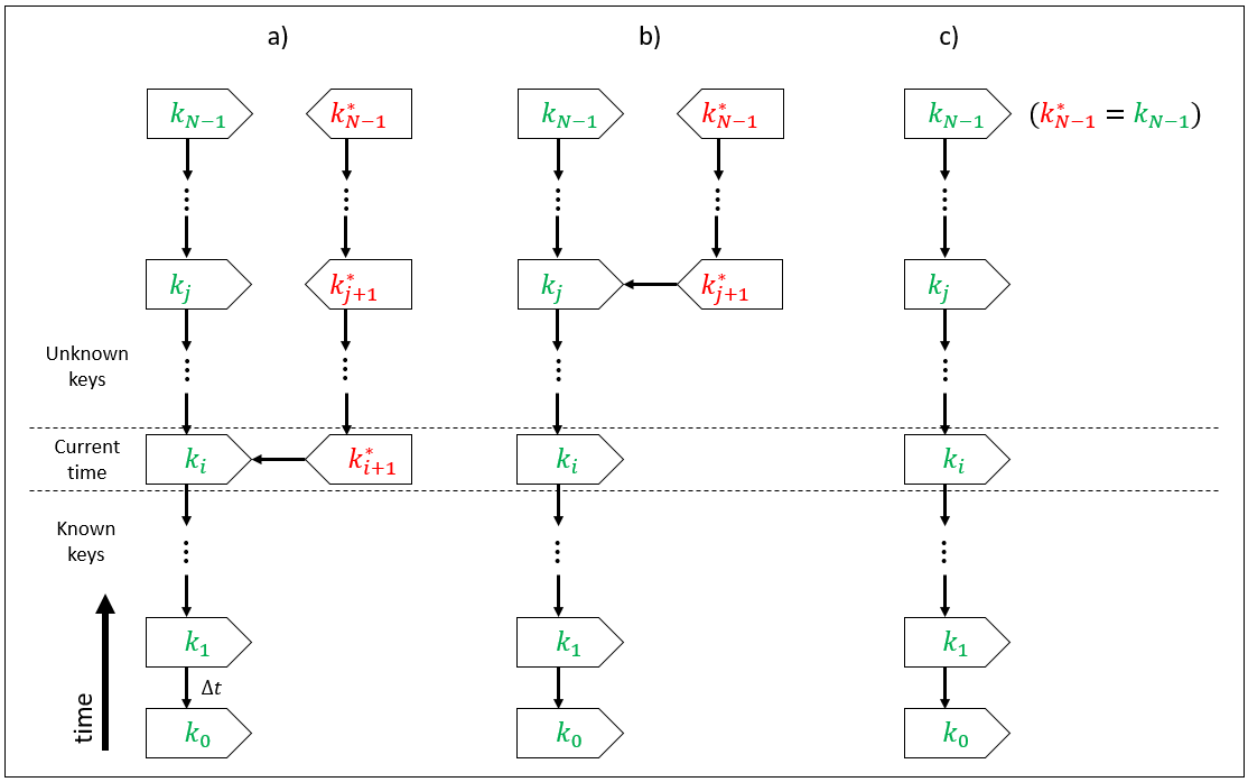

Figure 9: Possible outcomes for successful TESLA keychain attack. k^* denotes attacker keychain. a) attacker finds keychain that collides after key k^*_{i+1} that hashes to key k_i where $k^*_{i+1} \neq k_{i+1}$. b) attacker finds keychain that collides after key k^*_{j+1} that hashes to key k_j where $k^*_{j+1} \neq k_{j+1}$ and $i < j < N-1$. c) attacker finds keychain that is identical to the real 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 so it is uncertain where the actual collision took place. Assuming that there is a time Δt between each key release, the attacker would need to begin spoofing the receiver within at least time Δt, or at most time $(N-1-i) * \Delta t$ of the collision depending on where in the keychain the collision took place. 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). There is margin for the spoofer if they accept the risk of a missed spoof attempt and assume they've found a collision higher up the chain.

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 collision. The size of the key and hashing function, however, scales the amount of hashes required to find a collision exponentially. Assuming the SHA256 hash is used to create the keychain, the security level of the keychain is ~128-bits using classical computers. With Grover's algorithm and a quantum computer the keychain's security is reduced to ~64-bits. (See Errata) Citing Table 1 as a rough estimate for time to find a collision, if a keychain is used for a period of a month or greater, it is reasonable to assume that a collision can be found.

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 SHA3 series hash functions with bit lengths of 384-bits or more. 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 post-quantum 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 EC-Schnorr as an asymmetric cryptographic scheme for use in SBAS.

One field within post-quantum cryptography that is showing promise is the field of lattice-based cryptography using learning with errors (LWE). 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 10 [25]. 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 2 along with the security level of the scheme.

- 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 $[-b, \dots, b]$ 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 10: Learning with Errors

Another form of post-quantum 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 2. These values are derived from the patented Rainbow Signature Scheme found in [26].

There are also hash-based digital signatures, such as the SPHINCS signature [27], 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.

Table 2: Post-Quantum Asymmetric Cryptographic Schemes

Scheme	Public Key Size (kB)	Signature Size (bits)	Security Level
EC-Schnorr	~ 0.03	512	Not Quantum Secure
LWE (lattice-based)	2.0	14400	137
Rainbow Signature (Multivariate)	166.0	400	128
SPHINCS (hash-based)	1.056	328000	128

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. However, anticipating the near future development of quantum computers, it will be important to select a key size and hash function that are reasonably large to avoid attacks like the one shown in Figure 9. It is recommended that a SHA3 hash function of output 384-bits or higher is used in place of the older SHA256 algorithm. With an attack by Grover's algorithm this will leave the keychain with a security level of 96-bits, a reasonable size that will allow keychains to be longer and avoid frequent OTAR.

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 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 like EC-Schnorr and ElGamal. Its major drawback is the public key size which would make OTAR for this asymmetric scheme infeasible. Still, if the public key were known by the receivers a-priori, either through on the ground software integration or through higher bandwidth rekeying (WIFI), the signature involved in authenticating the root key of the TESLA keychain is a reasonable size and the scheme offers robust security level against all forms of computing attacks.

Conclusions and Future Work

Quantum computers are currently being developed and will someday have the ability to break classical cryptographic schemes used throughout the world. This paper offered two examples of primitives, the discrete logarithm and the hash function, that will witness decreased security when quantum computers of sufficient size come online. All systems using cryptography will be affected by quantum computing and it is imperative that SBAS authentication schemes being designed today are resilient to this future technology. 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. TESLA offers a secure and reasonably sized signature for SBAS authentication if it is designed with a sufficiently large key size (~ 384-bits). A first look into post-quantum secure algorithms also points to the Rainbow Signature scheme as a lightweight, secure algorithm for keychain authentication.

This paper stands as a pedagogical source, using the building blocks of cryptography to construct classical and post-quantum algorithms. Moving forward, an analysis for the bandwidth requirements for these post-quantum algorithms on SBAS will need to be completed as well as the infrastructure requirements for an SBAS authentication system. Moreover, these schemes will require scrutiny from the cryptographic and GNSS community to ensure that a secure method of authentication is designed for future SBAS.

References

[1] [Online]. Available: https://www.dwavesys.com/.

[2] G. Caparra, S. Sturaro, N. Laurenti and C. Wullems, "Evaluating the Security of One-way Key Chains in TESLA-based GNSS Navigation Message Authentication Schemes," in *International Conference on Localization and GNSS*, Barcelona, 2016.

[3] J. M. Anderson, et al. "Chips-Message Robust Authentication (Chimera) for GPS Civilian Signals," in *International Technical Meeting of The Satellite Division of the Institute of Navigation*, Portland, 2017.

[4] M. Stevens, E. Bursztein, P. Karpman, A. Albertini and Y. Markov, "The first collision for full SHA-1," *IACR Cryptology ePrint Archive,* p. 190, 2017.

[5] Bitmain. [Online]. Available: https://www.bitmain.com/. [Accessed 20 November 2017].

[6] D. R. Stinson, Cryptography: Theory and Practice, Boca Raton, FL: Chapman & Hall, 2006.

[7] D. J. Bernstien, J. Buchmann and E. Dahmen, Post-Quantum Cryptography, Chicago, IL: Springer, 2009.

[8] S. Lo, et al. "Signal authentication: A secure civil GNSS for today," *inside GNSS,* pp. 30-39, 2009.

[9] A. J. Kerns, et al. "A Blueprint for Civil GPS Navigation Message Authentication," in *IEEE/ION*, 2014.

[10] A. D. Chiara, G. D. Broi, O. Pozzobon, S. Sturaro, G. Caparra, N. Laurenti, J. Fidalgo, M. Odriozola, J. C. Ramon, I. Fernandez-Hernandez and E. Chatre, "Authentication Concepts for Satellite-Based Augmentation Systems," in *ION GNSS*, Portland, 2016.

[11] A. Perrig, D. Song, R. Canetti, J. D. Tygar and B. Briscoe, "Timed Efficient Stream Loss-Tolerant Authentication (TESLA): Multicast Source Authentication Transform Introduction," 2005.

[12] Q. Dang, "Recommendation for Applications Using Approved Hash Algorithms," NIST, 2012.

[13] International Standards Organisation, "ISO/IEC 14888-3. Information technology - Security techniques - Digital signatures with appendix - Part 3 - Discrete logarithm based mechanisms - Ammendment 1," International Standards Organisation, 2009.

[14] R. Feynman, "Simulating Physics with Computers," *International Journal of Theoretical Physics,* pp. 467-488, 1982.

[15] [Online]. Available: https://www.research.ibm.com/ibm-q/.

[16] M. Reynolds, "Google on track for quantum computer breakthrough by end of 2017," *New Scientist,* 26 June 2017.

[17] M. Branscombe, "Why Microsoft believes we're on the threshold of quantum computing," *TechRadar,* 20 December 2016.

[18] T. Simonite, "Intel bets it can turn everyday silicon into quantum computing's wonder material," *Technology Review,* 21 December 2016.

[19] Alibaba Group, *Aliyun and Chinese academy of Sciences sign MoU for quantum computing laboratory,* 2015.

[20] D. Beckman, A. N. Chari, S. Devabhaktuni and J. Preskill, "Efficient networks for quantum factoring," *Physical Review A,* vol. 54, no. 2, pp. 1034-1063, 1996.

[21] P. W. Shor, "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer," *SIAM J. Sci. Statist. Comput.,* pp. 1484-1509, 1997.

[22] L. K. Grover, "A fast quantum mechanical algorithm for database search," in *ACM symposium on theory of computing*, Philadelphia, 1996.

[23] D. Goodin, "NSA preps quantum-resistant algorithms to head off crypto-apocalypse," *Ars Technica,* 21 08 2015.

[24] K. Wesson, M. Rothlisberger and T. Humphreys, "Practical cryptographic civil GPS signal authentication," *Navigation,* vol. 59, no. 3, pp. 177-193, 2012.

[25] A. Chopra, "GLYPH: A New Instantiation of the GLP Digital Signature Scheme".

[26] A. Petzoldt, S. Bulygin and J. Buchmann, "Selecting parameters for the Rainbow Signature - Extended Version -," *Post-Quatnum Cryptography,* pp. 218-240, 2010.

[27] D. J. Bernstein et al., SPHINCS: practical stateless hash-based signatures, Berlin: Springer, 2015.

Errata

In the original form of this paper, it is stated that the security of a hash function is defined by its strength against a birthday attack type scenario. This is true for applications where the input AND output of the hash function are of no importance in attacking it. TESLA has a predefined keychain and so an attack to forge the keychain would need to find the pre-image of a key somewhere up the keychain, meaning that the birthday attack is not a valid attack in this case. Therefore, for the case of Grover's "black box" algorithm, the security of the preimage of the hash function is only halved. For example, for a SHA256 hash function, the preimage has a security of 128-bits after the use of Grover's algorithm. This change does not change most of this paper but gives more leniency to the design of the TESLA keychain in terms of the length of the keys. A peer-reviewed version of this article has been accepted to NAVIGATION at

the time of this writing under the same title and provides a more thorough look at quantum computing and its implications with respect to SBAS authentication.