Chapter 1

GPS SIGNAL AUTHENTICATION USING THE CHAMELEON HASH KEYCHAIN

Yu Han Chu, Sye Loong Keoh, Chee Kiat Seow, Qi Cao, Kai Wen and Soon Yim Tan

Abstract Global Navigational Satellite Systems (GNSS) are used to provide accurate time synchronisation and location services. Satellites transmit navigational messages that can be used by a receiver to compute its current location. However, most of the navigational messages are not protected and can be easily spoofed. Many GPS signals spoofing attacks have been reported in which the attackers are able to transmit spoofed and replayed GPS signals to hijack autonomous vehicles, ships and drones. Non-cryptographic methods that use pseudorange differences, antenna arrays and multi-receivers to detect GPS spoofing can be inaccurate due to environment changes. In this paper, we propose an efficient verification protocol of GPS signals by having a dedicated authentication server to continuously compute Keyed-Hashing for Message Authentication Code (HMAC) of GPS navigational messages received from the satellites using *Chameleon Hash Keychain*. The keychain has a unique property that it is practically unbounded, and this allows the GPS receivers to easily authenticate the authentication server, and verify the GPS signals concurrently by checking the HMACs. A proof-of-concept prototype has been developed to demonstrate the feasibility of this authentication scheme, and our results show that our approach can update the hash key in the keychain every 30 seconds, thus protecting every five GPS message subframes with a different hash key. This makes it difficult for the attacker to compromise the navigational messages.

Keywords: GNSS Authentication, Global Positioning System, Chameleon Hashing, Integrity Protection

1. Introduction

The proliferation of location services have led to the increased use of Global Positioning Systems (GPS) in automotive, maritime and aviation industries. In the maritime industry in particular, navigation in the sea is especially important to ensure security and safety. Other innovative applications such as delivery using unmanned autonomous drones, and aerial surveillance require an accurate GPS service to ensure the safety of its navigation, thus preventing malicious hijacking [16, 15] and diversion of drones. In order to control traffic congestion in London, Singapore and The Netherlands, drivers are being charged a fee during peak hours for driving into the city. There are proposals to charge road tax based on the distance traveled, thus the use of GPS will be prevalent.

However, it is widely known that Global Navigation Satellite Systems (GNSS), e.g., GPS, can be easily spoofed [7]. Firstly, the GPS signals received on a GPS device have no authentication or integrity protection, therefore the authenticity of the signals cannot be determined, and the signals can be replayed to spoof a GPS receiver. Secondly, an adversary can easily make use of low-cost Software-Defined Radio (SDR), i.e., *HackRF-One* to transmit historical GPS signals to the receivers with an aim to divert and hijack autonomous vehicles, ships [1], and drones [20] to another location. Thirdly, users will be motivated to spoof the location of their vehicles if they were being taxed based on the distance traveled. The in-vehicle GPS device must not only be hardened to prevent hardware tampering, but must also be able to verify the authenticity of the GPS signals received to prevent location spoofing.

This paper proposes a novel and efficient approach to verify the authenticity of GPS signals by continuously computing Keyed-Hashing for Message Authentication Code (HMAC) of GPS navigational messages received from the satellites using the *chameleon hash keychain*. The generated HMACs serve as fingerprints for GPS receivers on vehicles, drones and mobile devices to authenticate their GPS signals in real-time. In our scheme, we use a new key to protect every frame of navigational message. When each frame is HMAC-ed using a different key on the keychain, this makes tampering and spoofing very difficult. With this, we eliminate the need to set up a Public-Key Infrastructure (PKI) to provide authentication. GPS receivers can easily verify the key on the keychain, and subsequently use the authenticated key to compute the HMACs of GPS subframes to perform signal authentication. The main contributions of this paper are as follows:

- Near real-time secure GPS authentication protocol using an unbounded one-way *Chameleon Hash Keychain.*
- Fast and efficient authentication of GPS signals by synchronising the keychain, without needing PKI and modification of the navigational message.

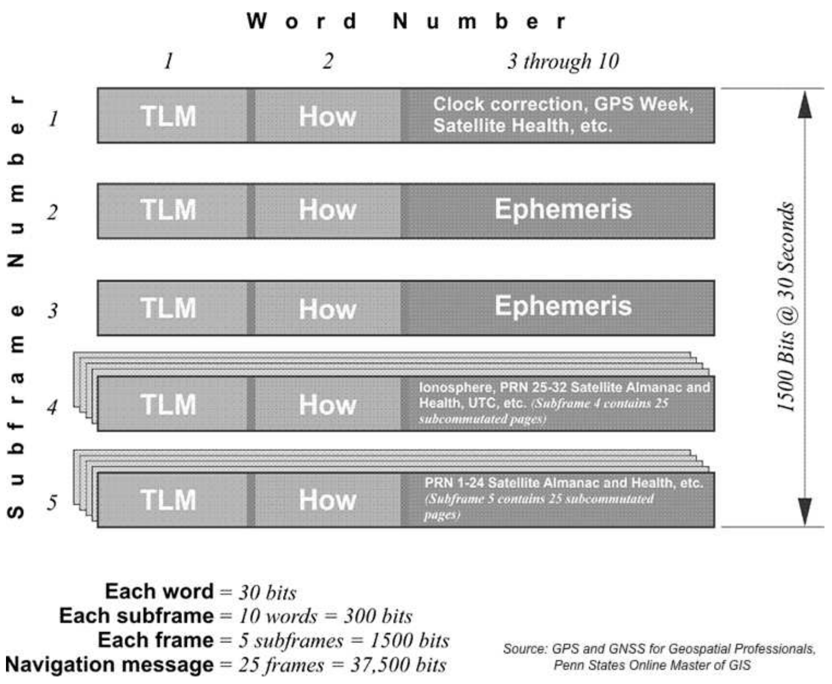
Word Number
1
2
3 through 10
Subframe Number
1 TLM How Clock correction, GPS Week, Satellite Health, etc.
2 TLM How Ephemeris
3 TLM How Ephemeris
4 TLM How Ionosphere, PRN 25-32 Satellite Almanac and Health, UTC, etc. (Subframe 4 contains 25 subcommutated pages)
5 TLM How PRN 1-24 Satellite Almanac and Health, etc. (Subframe 5 contains 25 subcommutated pages)
1500 Bits @ 30 Seconds
Each word = 30 bits
Each subframe = 10 words = 300 bits
Each frame = 5 subframes = 1500 bits
Navigation message = 25 frames = 37,500 bits
Source: GPS and GNSS for Geospatial Professionals, Penn States Online Master of GIS

Figure 1. GPS Navigational Message Structure.

- Easy deployment of GPS signal authentication on existing infrastructure and Internet Protocol (IP) networks without requiring modification to the navigational messages or deployment of additional satellites.

This paper is organised as follows: Section 2 provides some background information on GPS signals, chameleon hash functions and related work. Section 3 describes the proposed GPS authentication scheme using *chameleon hash keychain*. Section 4 presents the implementation and evaluation of the protocol. Finally, we concludes the paper in Section 5.

2. Background and Related Work

This section provides background on GPS navigational message [22] and then introduces chameleon hash and its usage. It also describes several related work on location spoofing detection techniques.

2.1 GPS Signals

Figure 1 shows the GPS L1 C/A Navigation Message structure. Each satellite transmits a continuous stream of data to earth at 50 bits per second. These data contain the system time, the clock correction values, the satellite's orbital data (*ephemeris*), the orbital data of all other satellites (*almanac*) and the satellite's system health. These data are grouped into units known as frames or pages. A complete navigational message consists of 25 frames. Each frame contains 1500 bits and divided into five subframes. Each subframe contains 300 bits, and it takes 6 seconds to transmit from the satellite to the device. The time taken to transmit the entire navigational message is 12.5 minutes. Currently, this navigation message is not protected using any cryptography and therefore susceptible to spoofing attacks.

2.2 Chameleon Hashing

Chameleon hashing [12] is a type of trapdoor collision-resistant function that is associated with a pair of public and private keys. The function is easy to compute in one direction but difficult to compute in the reverse direction without the private key also known as the "*trapdoor*". The holder of the private key can easily detect the collision for every input and it has the power to change the input value while being able to compute the same output hash value. Chameleon constructs can be based on Discrete Logarithm and Elliptic Curve Cryptography [11].

Chameleon Hashing has been adopted to secure the integrity of energy usage data in Advanced Metering Infrastructure (AMI) [21] and industrial control systems [9, 10] to enable aggregated energy readings to be protected, so that any tampering of the data can be detected by the energy provider. The aim is to provide an end-to-end data integrity protection between the smart meter and the backend service provider such that when the concentrator in AMI tampered with the energy readings, this can be detected by checking the chameleon hash value.

2.3 Related Work

In [13], the authors proposed a non-cryptographic method to compute the pseudorange differences to detect meaconing, simplistic and intermediate spoofing attacks. A signal pseudorange model based on the signal transmission path is built to establish the double-difference of pseudorange of two adjacent epochs. By applying the Taylor expansion to the position relationship between the satellite and the receiver or the spoofer, the authenticity of the signal can be verified by comparing the result of the spoofing detection algorithm with the result of the traditional least squares method. Other spoofing detection approaches using signal processing include the use of antenna array [23, 8], determining the receiver pseudorange or carrier phase difference [2, 19], and correlation method [3, 18]. As the environment changes, these signal-based anti-spoofing methods may not be accurate in detecting spoofing.

Cryptography schemes have also been proposed to protect GNSS signals. In [25], the authors proposed an anti-spoofing scheme of BeiDou-II Navigation Message using the Chinese SM cryptographic standards to encrypt the navigational messages. Integrity of the navigational messages is protected by inserting the Spread Spectrum Information (SSI) between subframe 1 and subframe 2 in the D2 navigation messages. However, this requires modification to the navigational message format, which may be difficult to deploy. Other similar schemes used RSA or Elliptic Curve Digital Signature Algorithm (ECDSA) to secure the navigational messages and then broadcast the authenticated signals using QZSS L1SAIF navigation message [14] or GPS Civil Navigation Message (CNAV) [24].

The effectiveness of broadcast authentication had led to the use of TESLA [17] for securing navigational messages. In [5], TESLA was used to enable the sender to encrypt the navigational message, while the receivers only need to wait for the sender to reveal the key in order to authenticate the signals. In [6], TESLA was applied to the navigation message of GPS L1 C/A, but this requires the protocol to be implemented on Civil Navigation signals (CNAV-2) of GPS L1C. [26] used ECSDA in combination with TESLA to protect BeiDou civil navigational signals, using ECDSA to ensure reliability of the signals as TESLA is very efficient for broadcast authentication. Although One-way keychain is very efficient, TESLA is a bounded keychain and requires a new keychain to be set-up once all the keys have been exhausted. Additionally, TESLA relies on loosely synchronised-time between the server and the receivers to ensure data authenticity. Our scheme improves these weaknesses with an unbounded keychain.

3. GPS Signal Authentication

This section presents the threat model, system assumptions and introduces *Chameleon Hash Keychain.* We then describe the proposed GPS authentication protocol using an unbounded *Chameleon Hash Keychain.*

3.1 Threat Model and Assumptions

An adversary is motivated to spoof the location of vehicles, drones or mobile devices by transmitting spoofed or replayed GPS signals. The following lists the assumptions in this work:

- Adversary has access to SDR or low cost transmitter to broadcast spoofed GPS signals to take over the real GPS signals from the satellites.
- Adversary must execute the SDR with a transmitter that is in close physical proximity to the victim's devices in order to spoof the location.
- Adversary does not need to have access to the hardware or software of the victim device's GPS receiver in order to tamper with the signals.
- Adversary does not need to physically capture the victim's device in order to launch a location spoofing attack.

3.2 Chameleon Hash Keychain

This section introduces the unbounded one-way keychain [4] that can be used to provide fast and efficient authentication between two parties. Table 1 shows the notations used for the construction of a one-way keychain (known as *Chameleon Hash Keychain*) using Chameleon Hashing.

Table 1. Notations used for constructing a chameleon hash keychain.

Notation	**Description**
$\mathbb{CH}$	Chameleon Hash Function
$\mathbb{CH}'$	Trapdoor Chameleon Hash Function
$\mathbb{K}$	Trapdoor Key
HK	Chameleon Hash Value or Hash Key
m	Input Message for $\mathbb{CH}$
m'	Message where $m' \neq m$
r	Random prime Input for $\mathbb{CH}$
r'	Collision resulting from $\mathbb{CH}'$

To generate a one-way unbounded *chameleon hash keychain*, i.e., $HK_0 \rightarrow HK_1 \rightarrow HK_2 \rightarrow ... \rightarrow \infty$, a random message m_0 and a random prime number r_0 are first chosen, to generate HK_0 using Eq. 1 where $n = 0$:

$$HK_n = \mathbb{CH}(m_n, r_n) \tag{1}$$

The main property of chameleon hashing is that it is easy to find a hash collision if the trapdoor key, $\mathbb{K}$ is known. This means that it is feasible to derive a separate pair of message (m', r') such that when they are hashed using Chameleon Hash Function, it results in the same Chameleon Hash Value (or Hash Key) using the (m, r), where $m \neq m'$ and $r \neq r'$, as shown in Eq. 2.

$$\mathbb{CH}(m_n, r_n) = \mathbb{CH}(m'_n, r'_n) \tag{2}$$

For subsequent hash keys, HK_n where n = 1,2,3,..., by using Eq. 1, each new HK_n is generated using two parameters m_n and r_n. The keychain is formed by linking the new hash key, HK_{n+1} with its previous hash key, HK_n by finding a collision, i.e., compute the corresponding r' using the Trapdoor key $\mathbb{K}$, where $m'_n = HK_{n+1}$ using Eq. 3.

$$r'_n = \mathbb{CH}'(\mathbb{K}, m_n, r_n, HK_{n+1}) \tag{3}$$

Eq. 4 shows the resulting relationship between two consecutive hash keys on the keychain such that HK_n is equals to $\mathbb{CH}(HK_{n+1}, r'_n)$ where HK_{n+1} is generated using (m_{n+1}, r_{n+1}). It is easy to determine the authenticity of HK_{n+1} and r'_n in that when they are hashed, it results in a hash value that is equal to the previous hash key, HK_n.

$$HK_n = \mathbb{CH}(HK_{n+1}, r'_n) \tag{4}$$

$\mathbb{CH}$ is a public function without requiring the knowledge of $\mathbb{K}$, anyone is able to verify the authenticity of the hash key on the chain, but it is difficult to derive the future hash keys. With this, there is no need for the two parties to synchronise their clocks, as the new hash keys can be revealed on-demand and then verified immediately.

3.3 Architectural Overview

Figure 2 shows the architectural overview of the proposed GPS authentication scheme to defend against location spoofing. As GPS L1 C/A signals are not protected, we propose to establish multiple land-based *GPS Authentication Servers* to secure the GPS navigational messages from satellites using *Chameleon Hash Keychain.* All navigational message's frames received are protected by a HMAC computed using the latest hash key on the keychain every 30 seconds. This is done by the main authentication server in a physically secured location. Other au-

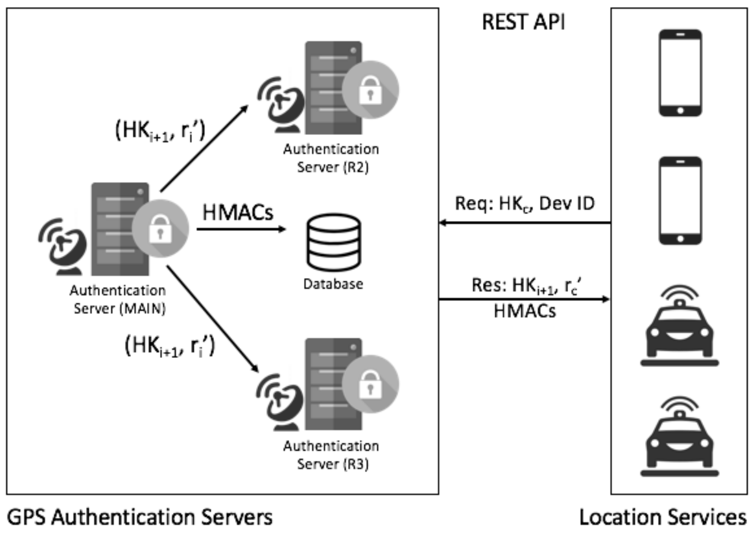
REST API
(HK_{i+1}, r_i')
Authentication Server (R2)
HMACs
Database
Authentication Server (MAIN)
(HK_{i+1}, r_i')
Authentication Server (R3)
Req: HK_c, Dev ID
Res: HK_{i+1}, r_c' HMACs
GPS Authentication Servers
Location Services

Figure 2. Architectural Overview of GPS Signal Authentication.

thentication servers located at a different location can verify the HMACs based on the latest hash key obtained from the main authentication server. If HMAC verification fails, this implies that one of the servers' GPS signals have been spoofed.

The proposed GPS authentication protocol has three phases:

- *Hash Key Generation and Distribution* – Set-up the *Chameleon Hash Keychain* by the *GPS Authentication Server* and distribute the hash key, HK to clients periodically.
- *Securing GPS Navigational Messages* – *GPS Authentication Server* computes HMACs of each subframe using the latest hash key in real-time.
- *Verifying GPS Signals* – Clients verify the hash key and GPS signals they received directly from the satellites by verifying the HMACs with the *GPS Authentication Server*.

Any location-based services can execute the proposed security protocol to authenticate the *GPS Authentication Server*, and verify the GPS signals it receives by checking the chameleon hash of the signals through an Representational State Transfer (REST) interface.

We have defined additional notation in Table 2 for the proposed GPS signals authentication protocol.

Table 2. Notations used for GPS signal authentication protocol.

Notation	Description
$\mathbb{sf}$	Subframe of GPS navigational message
$\mathbb{E}$	Elliptic curve used for Chameleon Hash Function
G	Base point of NIST defined curve $secp256r1$
Y	ECC public key
y	ECC private key, a.k.a Trapdoor key, $\mathbb{K}$

3.3.1 Hash Key Generation and Distribution.

The *GPS Authentication Server* is responsible for generating and maintaining a chameleon hash keychain to secure the GPS navigational messages. The *NIST Prime Curve P-256*, E of the form $y^2\ (mod\, p) = x^3 + ax + b\ (mod\ p)$ over the finite field, F_p is used as the construct for *Chameleon Hash Function*. The ECC domain parameters are first generated, where p is a large prime number and a, b are the coefficients of the elliptic curve. G is a generator selected from the elliptic curve. A random value y is chosen as the private key (a.k.a. trapdoor key), and then the corresponding public key is computed as $Y = yG$.

The first hash key, HK_0 is generated using a random message, m_0 and a random prime, r_0 as follows:

$$HK_n = \mathbb{CH}(m_n, r_n) = HMAC(m_n, Y) \cdot Y + r_n \cdot G \tag{5}$$

where $HMAC(m_n, Y)$ is a keyed message authentication code of m_n, i.e., SHA-256 with an input key Y.

Every time a new hash key, HK_{n+1} is generated by the *GPS Authentication Server* using $\mathbb{CH}(m_{n+1}, r_{n+1})$, an r'_n is derived to link HK_{n+1} with the previous hash key HK_n using Eq. 6. Note that only the *GPS Authentication Server* can generate a hash collision to find r'_n as it is the only entity that possesses the private key y (a.k.a. trapdoor key). Both HK_{n+1} and r'_n are then distributed to all the clients.

$$r'_n = y \cdot [HMAC(m_n, Y) - HMAC(HK_{n+1}, Y)] + r_n \tag{6}$$

Location-based services which require GPS authentication can bootstrap its application by registering its device, and obtain the initial hash key, HK_0 and/or the latest hash keys from the *GPS Authentication Server* through a TLS session.

3.3.2 Securing GPS Navigational Message. The proposed authentication protocol secures the entire GPS navigational messages received from all satellites. Each GPS message frame (i.e., five subframes) is protected using a different hash key from the *Chameleon Hash Keychain*.

We propose that a physically secured *GPS Authentication Server* is set up, with a dedicated GPS signals receiver to collect all the navigational messages from each GPS satellite that can be detected. This server can be placed in a highly secured premise (e.g., in a government office) to prevent attacker from spoofing the GPS signals. Whenever a complete frame consisting of five subframes, has been obtained, the server will compute the next hash key on the *chameleon hash keychain* using Eq. 5, where m is the concatenation of Subframe 1 to 5 of each message frame, r is a random prime securely generated by the authentication server. The following provides detailed steps to be executed by the *GPS Authentication Server*:

1 In the set-up phase, HK_0 is generated using a random message m_0 and a prime number r_0, $HK_0 = \mathbb{CH}(m_0, r_0)$. The *GPS Authentication Server* maintains a *chameleon hash keychain*, generating a hash key every 30 seconds. The current hash key is denoted as HK_i.

2 When receiving GPS navigational message, concatenate every five subframes together as the next message, m_{i+1}, e.g., when $i = 0$, $m_1 = \mathbb{sf}_\mathbb{1} + \mathbb{sf}_\mathbb{2} + \mathbb{sf}_\mathbb{3} + \mathbb{sf}_\mathbb{4} + \mathbb{sf}_\mathbb{5}$.

3 Compute the next hash key, HK_{i+1} using Eq. 5 by selecting a random prime r_{i+1}, hence $HK_1 = \mathbb{CH}(m_1, r_1)$.

4 Compute r'_i using the private key y to link HK_{i+1} and HK_i together using Eq. 6, such that $HK_0 = \mathbb{CH}(HK_1, r'_0)$.

5 Finally, compute the HMAC of each subframe, using SHA-256 and the new Hash Key, HK_{i+1}. Store the HMACs in the database to facilitate GPS verification requests by the clients.

Algorithm 1 summarises the generation of *chameleon hash keychain* using GPS navigational messages as inputs. The hash key is then used to compute HMAC (fingerprint) of every message subframe, that can later be verified by clients over the Internet. This means that the hash key is renewed per frame, i.e., every 30 seconds, making it extremely difficult for the attacker to spoof the GPS signals.

Algorithm 1 Generation of Chameleon Hash Keychain and Securing GPS Navigational Messages (By GPS Authentication Server)

```
1: Initialise public key: $Y = yG$
2: Choose a random message $m_0$ and random prime $r_0$
3: Generate initial $HK_0 = HMAC(m_0, Y) \cdot Y + r_0 \cdot G$
4: $i = 0$
5: for Each complete message frame received do
6:    $m_{i+1} = \mathbb{sf}_1 + \mathbb{sf}_2 + \mathbb{sf}_3 + \mathbb{sf}_4 + \mathbb{sf}_5$
7:    Generate a random prime, $r_{i+1}$
8:    $HK_{i+1} = HMAC(m_{i+1}, Y) \cdot Y + r_{i+1} \cdot G$
9:    $r'_i = y \cdot [HMAC(m_i, Y) - HMAC(HK_{i+1}, Y)] + r_i$
10:
11:   Generate HMAC for each subframe using $HK_{i+1}$
12:   for j=1; j<6; j++ do
13:      $HMAC_j = HMAC(\mathbb{sf}_\mathbb{j}, HK_{i+1})$
14:      Store $HMAC_j$ in the database
15:   end for
16:   i++
17: end for
```

3.3.3 Verification of GPS Signals. Location-based services clients can verify the GPS navigational messages they received from the satellites directly by first synchronising the hash key with the *GPS Authentication Server* as follows:

Client → *Server*: HK_c, $Device_{ID}$
Server → *Client*: HK_{i+1}, r'_c

Assuming that the client currently possesses a hash key, where $HK_c \neq HK_i$, it sends its $Device_{ID}$, HK_c to the *GPS Authentication Server* to obtain the next hash key, HK_{i+1}. The server can easily compute an r'_c specific for the requesting client using Eq. 6 such that $HK_c = \mathbb{CH}(HK_{i+1}, r'_c)$. The server then sends HK_{i+1} and r'_c back to the client, enabling the client to authenticate the server. When successful, the client can maintain a synchronised *chameleon hash chain* with the server to continuously authenticating the GPS signals for the session.

In order to verify the GPS navigational messages, the client sends a request to the *GPS Authentication Server* to obtain the HMACs of the frame, by indicating the *satellite id*, and *frame id*. The client device also receives the subframes messages directly from the satellites, it can then use the next hash key HK_{i+1} to verify the HMAC of each subframe. The

GPS data is authentic if the computed HMACs match with the HMACs obtained from the *GPS Authentication Server.* Algorithm 2 shows the verification process as described below:

1 Client synchronises the keychain with the *GPS Authentication Server.*

2 Client obtains the current hash key, HK_i and verifies that the key is authentic, hence authenticating the *GPS Authentication Server.*

3 Once the keychain is synchronised, the client can authenticate the server every 30 secs, by verifying that $HK_i = \mathbb{CH}(HK_{i+1}, r'_i)$.

4 Client requests HMACs of each satellite's frame k consisting of five subframes, i.e., $\mathbb{SF}_{\mathbb{1}_\mathbb{k}}$, $\mathbb{SF}_{\mathbb{2}_\mathbb{k}}$ ... $\mathbb{SF}_{\mathbb{5}_\mathbb{k}}$ from the server.

5 For all subframes in frame k received by the client, it computes the corresponding $HMAC(\mathbb{SF}_{\mathbb{j}_\mathbb{k}}, HK_{i+1})$ where $j = 1, 2, 3, ...5$ and verifies that they all match the respective HMAC provided by the *GPS Authentication Server.*

It is crucial that the clients ensure the authenticity of Hash Key, so that HMAC verification can be trusted. The authenticity of HK is provided through the unique property of *Chameleon Hash Keychain* in that the $\mathbb{CH}$ of the current hash key is equals to the previous hash key, i.e. Eq. 6. As the computation of HMAC is fast, it is efficient for the clients to verify the GPS navigational message in every 30 seconds interval.

4. Implementation and Evaluation

We have implemented a proof-of-concept prototype to validate the proposed GPS signal authentication protocol. We made use of an Android smart phone to act as the GPS signals receiver for the *GPS Authentication Server*, so that the raw GPS navigational messages can be obtained directly. In the real deployment, a proper GPS receiver should be used. The GPS navigational messages are received continuously, in which a subframe is received every 6 seconds. As each message frame contains five subframes, it takes about 30 seconds to collect all five subframes, after which the *GPS Authentication Server* can compute the respective HMACs to be stored in the cloud database.

The *GPS Authentication Server* was deployed on Amazon Elastic Compute Cloud (EC2) running Ubuntu OS. The GPS authentication service is operational 24/7 and accessible through the Internet. *NodeJS* and *Express JS* were used to develop an REST API to allow for any location-based applications and clients to request for authenticated GPS

Algorithm 2 Verifying the Authenticity of GPS Navigational Message (By the Client)

```
1: Client maintains the Chameleon Hash Keychain, $HK_0 \rightarrow HK_1 \rightarrow ... \rightarrow HK_i$
2: Perform authentication with GPS authentication server
3: Synchronise the keychain
4: Receive $HK_{i+1}$ and $r'_i$ from Server
5: Verify $HK_{i+1}$ such that $HK_i = \mathbb{CH}(HK_{i+1}, r'_i)$
6:
7: for Each message frame $\mathbb{k}$ received do
8:   Request HMACs from Server for $\mathbb{SF}_{\mathbb{jk}}$ where $j = 1, 2, ..., 5$
9:   Compute $HMAC'_j = HMAC(\mathbb{SF}_{\mathbb{jk}}, HK_{i+1})$, $j = 1, 2, ..., 5$
10:  if $HMAC_j == HMAC'_j$ then
11:    Subframe is authenticated
12:  else
13:    Subframe is spoofed
14:  end if
15: end for
```

signals from the authentication server. As for the database, *Mongo DB Atlas*, a cloud-based database system that uses NoSQL was used.

4.1 Chameleon Hash Keychain

We also developed a *C library* for Chameleon Hashing based on Elliptic Curve Cryptography (ECC), using the OpenSSL library and Bouncycastle Crypto Library. The two main constructs are as follows:

- EC_POINT ***generateChameleonHash***(EC_GROUP *E, EC_POINT *Y, unsigned char *m, BIGNUM *r)*
- BIGNUM ***computeChameleonRPrime***(EC_GROUP *E, EC_POINT *Y, BIGNUM *y, BIGNUM *r, unsigned char *m, unsigned char *m_prime)*

Our implementation used the *NIST Prime-Curve P-256.* The *Chameleon Hash Keychain* implementation is based purely on these two constructs, i.e., to generate a chameleon hash value HK_n, and to compute the collision r'_n.

4.2 GPS Receiver using Android Phones

We also developed an Android application to obtain the raw GPS navigational messages using the Android location library. As the GPS

Table 3. Supported GNSS systems by Samsung Note 8 & S8+.

Model	**Android Version**	**NAV Msg**	**ADR**	**Supported Global Systems**
Note 8	9.0	Yes	Yes	GPS, GLO, GAL, BDS
S8+	9.0	Yes	Yes	GPS, GLO, GAL, BDS, QZS

navigational message is a continuous stream of data transmitted by the satellites, our implementation used *GnssNavigationMessage.Callback()* to listen for event changes. Specifically, the *onGnssNavigationMessageReceived()* and *onStatusChanged()* functions are triggered to retrieve the raw GPS data received by the phone. However, only selected models of Android-based mobile phones are supported. In this implementation, we used *Samsung Note 8* and *Samsung S8+* to obtain the raw GPS data, one acting as the GPS receiver for the *GPS Authentication Server*, while the other acted as the client attempting to verify the received GPS navigational messages. Table 3 shows the supported GNSS signals that can be retrieved by *Samsung Note 8* and *S8+*.

4.3 Execution Time

As there are 31 GPS satellites in the GPS constellation, it is important that the *GPS Authentication Server* is able to run the hash key generation concurrently, though not all GPS satellites may be detected at one location.

Table 4 shows the performance of chameleon hash function using native C implementation. The hash key generation took 137.2 μs, while computing r'_i, i.e., collision generation took approximately 20.1 μs. The result shows the average time of 1000 sequential execution of each function. The chameleon hash key generation took longer time as compared to collision generation due to the two elliptic curve point multiplication operation, as compared to one point multiplication for collision generation. The $HMAC(m, Y)$ function was fast and efficient as it took 10.7 μs to generate five HMACs consecutively.

4.4 Communication Overhead

When comparing our protocol with the conventional ECDSA digital signature approach that generates a signature for each GPS message subframe, our approach incurs less overhead for a large number of clients. Assuming that the size of ECDSA signature is 64 bytes, a complete GPS navigational message contains 25 frames (i.e., 125 subframes) and hence this incurs an overhead of 7.81 KB per navigation message per client.

Table 4. Execution time of Chameleon Hash Keychain Functions and HMAC.

Function	**Time (μs)**
Chameleon Hash Key Generation $HK = \mathbb{CH}(m, r)$	137.2
Collision Generation $r'_i = \mathbb{CH}'(y, m_i, r_i, HK_{i+1})$	20.1
Computing five HMACs $HMAC(m, Y)$	10.7

In our protocol, the HMAC is 32 bytes, while Hash Key HK_{i+1} and r'_i are 64 bytes each. The hash key is renewed every frame, this incurs an overhead of 3.12 KB, while HMACs for 125 subframes incur 3.91 KB. The total overhead per navigation message per client is 7 KB. This has resulted in 10% less overhead as compared to ECDSA scheme for each message per client. The reason that each subframe needs to be signed or HMAC-ed separately is because some receivers may not have received all the subframes, and therefore each subframe needs to be verified individually.

4.5 Security Analysis

Based on the *Trapdoor collision property* of Chameleon Hash Function, there exists an efficient probabilistic polynomial time (PPT) algorithm A that on input of private key y, a message pair (m, r) and m', the *GPS Authentication Server* is able to output a value $r' \in Z_n$ such that hash collision occurs, i.e., $\mathbb{CH}(m', r') = \mathbb{CH}(m, r)$. The m' and r' value are sent to the clients wishing to verify its GPS signals, to advance their Chameleon Hash Keychain and renew the hash key. If the client cannot verify the (m', r') received, then the new hash key is not originated from the *GPS Authentication Server*. This can be detected without fail based on this property.

Another property of the proposed authentication protocol is *Collision resistant property.* There is no PPT algorithm that on input of a Hash Key, HK and without the knowledge of the private key y, the *GPS Authentication Server* can find pairs (m, r) and (m', r') where $m \neq m'$ such that $\mathbb{CH}(m', r') = \mathbb{CH}(m, r)$ with a non-negligible probability. This is equivalent to solving the ECDLP problem which is known to be computationally hard. By this property, no other entity other than the *GPS Authentication Server* can extend the keychain.

Every frame is protected with a HMAC using a different hash key. Unless the private key y of the *GPS Authentication Server* is compromised, it will be difficult for an attacker to fix the next hash key in advance, and use it to compute the HMACs of the next set of GPS frames. By changing the hash key frequently, it increases the authenticity guarantee of the GPS navigational messages, but this comes with an overhead to distribute the hash keys more frequently to all the clients.

5. Conclusions

This paper has presented a novel approach using *Chameleon Hash Keychain* to efficiently protect GPS navigational messages, allowing for clients to verify their GPS signals through a web service interface. Specifically, our approach adopts an unbounded one-way keychain generated using chameleon hash constructs, it provides the ability to use a new hash key to protect every frame of GPS navigational messages. The use of a one-way keychain also makes our protocol effective, in that clients can easily authenticate the *GPS Authentication Server*, by verifying the one-way property of the new hash key received. In the proposed scheme, there is no longer a need to loosely synchronise the time between the *GPS Authentication Server* and the clients. It is anticipated that in the future with 5G-enabled network, the network latency to verify the GPS signals through the Internet can be significantly reduced.

As we have only presented a preliminary evaluation of the prototype, the next step is to work together with government agencies to conduct a large-scale deployment of the proposed location authentication scheme, studying the network latency and system scalability. In the real deployment, the *GPS Authentication Server* must be hardened and integrated with sufficient web security protection.

Finally, the proposed GPS authentication protocol can be extended further to protect other GNSS systems such as GLO, GAL, BeiDou, etc, to ensure wider coverage worldwide.

References

[1] J. Bhatti and T.E. Humphreys, Hostile Control of Ships via False GPS Signals: Demonstration and Detection, *NAVIGATION*, vol. 64(1), pp. 51–66, 2017.

[2] D. Borio, and C. Gioia, A Sum-of-squares Approach To GNSS Spoofing Detection, *IEEE Transactions on Aerospace and Electronic Systems*, vol. 52(4), pp. 1756–1768, 2016.

[3] A. Broumandan, A. Jafarnia-Jahromi and G. Lachapelle, Spoofing Detection, Classification and Cancellation (SDCC) Receiver Architecture for a Moving GNSS Receiver, *GPS Solutions*, vol. 19, pp. 475–487, 2015.

[4] R. Di Pietro, A. Durante, L. Mancini and V. Patil, Practically Unbounded One-Way Chains for Authentication with Backward Secrecy. *Proceedings of the First IEEE International Conference on Security and Privacy for Emerging Areas in Communications Networks*, pp. 400–402, 2005.

[5] I. Fernndez-Hernndez, V. Rijmen, G. Seco-Granados, J. Simon, I. Rodrguez and J.D. Calle, A Navigation Message Authentication Proposal for the Galileo Open Service. *NAVIGATION* vol. 63(1), pp. 85–102, 2016.

[6] K. Ghorbani, N. Orouji and M. Mosavi, Navigation Message Authentication Based on One-Way Hash Chain to Mitigate Spoofing Attacks for GPS L1, *Wireless Personal Communications*, vol. 113, pp. 1743–1754, 2020.

[7] T.E. Humphreys, B.M. Ledvina, M.L. Psiaki, B.W. O'Hanlon and P.M. Kintner Jr., Assessing the Spoofing Threat: Development of a Portable GPS Civilian Spoofer, *Proceedings of the Twenty-First International Technical Meeting of the Satellite Division of The Institute of Navigation*, pp. 2314–2325, 2008.

[8] K. Jansen, N.O. Tippenhauer and C. Popper, Multi-Receiver GPS Spoofing Detection: Error Models and Realization. *Proceedings of the Thirty-Second Annual Conference on Computer Security Applications*, pp. 237–250, 2016.

[9] S.L. Keoh, K.W.K. Au and Z. Tang, Securing Industrial Control System: An End-to-End Integrity Verification Approach, presented at *2015 Industrial Control System Security (ICSS) Workshop*, (`https://www.acsac.org/2015/workshops/icss/`), 2015.

[10] S.L. Keoh, H.C. Tan and Z. Tang, Authentication and Integrity Protection for Real-time Cyber-physical Systems, in *Handbook of Real-Time Computing*, Y.C. Tian, and D.C. Levi (Eds.), Springer, Singapore, pp. 1–22, 2020.

[11] N. Koblitz, Elliptic Curve Cryptosystems, *Mathematics of Computation*, vol. 48, pp. 203–209, 1997.

[12] H. Krawczyk and T. Rabin, Chameleon Hashing and Signatures, *IACR Eprint Archive*, (`http://eprint.iacr.org/1998/010`), 1998.

[13] K. Liu, W. Wu, Z. Wu, L. He and K. Tang, Spoofing Detection Algorithm based on Pseudorange Differences, *MDPI Sensors*, vol. 18(10), article 3197, 2018.

[14] D. Manandhar and R. Shibasaki, Authenticating GALILEO Open Signal using QZSS Signal. *Proceedings of the Thirty-First International Technical Meeting of the Satellite Division of The Institute of Navigation*, pp. 3995–4003, 2018.

[15] M.S. bin Mohammad Fadilah, V. Balachandran, P. Loh and M. Chua, DRAT: A Drone Attack Tool for Vulnerability Assessment, *Proceedings of the Tenth ACM Conference on Data and Application Security and Privacy (CODASPY)*, pp. 153–155, 2020.

[16] J. Noh, Y. Kwon, Y. Son, H. Shin, D. Kim, J. Choi and Y. Kim, Tractor Beam: Safe-Hijacking of Consumer Drones with Adaptive GPS Spoofing. *ACM Transactions on Privacy and Security*, vol. 22(2), Article 12, 2019.

[17] A. Perrig and J.D. Tygar, TESLA Broadcast Authentication, in *Secure Broadcast Communication*, A. Perrig and J.D. Tygar (Eds.), Springer, Boston, pp. 29–53, 2003.

[18] M.L. Psiaki, B.W. OHanlon, J.A. Bhatti, D.P. Shepard and T.E. Humphreys, GPS Spoofing Detection via Dual-Receiver Correlation of Military Signals, *IEEE Transactions on Aerospace and Electronic Systems*, vol. 49(4), pp. 2250–2267, 2013.

[19] D.S. Radin, P.F. Swaszek, K.C. Seals and R.J. Hartnett, GNSS Spoof Detection Based on Pseudoranges from Multiple Receivers, *Proceedings of the 2015 International Technical Meeting of The Institute of Navigation*, pp. 657–671, 2015.

[20] J. Su, J. He, P. Cheng and J. Chen, A Stealthy GPS Spoofing Strategy for Manipulating the Trajectory of an Unmanned Aerial Vehicle, *IFAC-PapersOnLine*, vol. 49(22), pp. 291–296, 2016.

[21] H.C. Tan, K. Lim, S.L. Keoh, Z. Tang, D. Leong and C.S. Sum, Chameleon: A Blind Double Trapdoor Hash Function for Securing AMI Data Aggregation, *Proceedings of the IEEE Fourth World Forum on Internet of Things*, pp. 225–230, 2018.

[22] J. Van Sickle and J. Dutton, GEOG862: GPS and GNSS for Geospatial Professionals, Penn States Online Master of GIS (`https://www.e-education.psu.edu/geog862/home.html`), 2014.

[23] W. Wang, G. Chen, R. Wu, D. Lu and L. Wang, A low-complexity Spoofing Detection and Suppression Approach for ADS-B, *Proceedings of the 2015 Integrated Communication, Navigation and Surveillance Conference*, pp. 1–24, 2015.

[24] K. Wesson, M. Rothlisberger and T.E. Humphreys, Proposed Navigation Message Authentication Implementation for Civil GPS Anti-spoofing, *Proceedings of the Twenty-Fourth International Technical Meeting of the Satellite Division of The Institute of Navigation*, pp. 3129–3140, 2011.

[25] Z. Wu, Y. Zhang and R. Liu, BD-II NMA&SSI: An Scheme of Anti-Spoofing and Open BeiDou II D2 Navigation Message Authentication, *IEEE Access*, vol. 8, pp. 23759–23775, 2020.

[26] M. Yuan, Z. Lv, H. Chen, J. Li and G. Ou, An Implementation of Navigation Message Authentication with Reserved Bits for Civil BDS Anti-Spoofing, in. *China Satellite Navigation Conference (CSNC) 2017 Proceedings: Volume II*, J. Sun, J. Liu, Y. Yang, S. Fan and W. Yu (Eds.) Springer, Singapore, pp. 69–80, 2017.