Chapter 10

GPS SIGNAL AUTHENTICATION USING A 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 provide accurate time synchronization and location services. Satellites transmit navigation messages that can be used by a receiver to compute its location. However, most navigation messages are not protected and are easily spoofed. Several attacks have been reported that transmit spoofed or replayed GPS signals to divert or hijack autonomous vehicles, ships and drones. Unfortunately, non-cryptographic protection methods that use antenna arrays, pseudorange differences and multi-receivers to detect GPS spoofing tend to be inaccurate due to environmental conditions.

This chapter proposes an efficient GPS signal authentication protocol that engages a dedicated server to continuously compute hash-based message authentication codes of GPS navigation messages received from satellites using the chameleon hash keychain. The keychain is practically unbounded, which enables GPS receivers to easily authenticate the server and verify GPS signals concurrently by checking the hash-based message authentication codes. A proof-of-concept prototype has been developed to demonstrate the feasibility of the authentication scheme. Experimental results demonstrate that the hash key in the keychain can be updated every 30 seconds, enabling every five GPS message subframes to be secured with a different hash key. This makes it difficult for attackers to compromise GPS navigation messages.

Keywords: GPS signals, authentication, integrity, chameleon hashing

1. Introduction

The proliferation of location services have led to the increased use of the Global Positioning System (GPS) in the automotive, maritime and aviation sectors. Safe and secure navigation are vital. Therefore, it is

Published by Springer Nature Switzerland AG 2022
J. Staggs and S. Shenoi (Eds.): Critical Infrastructure Protection XV, IFIP AICT 636, pp. 209–226, 2022.
https://doi.org/10.1007/978-3-030-93511-5_10

important to ensure that automobiles, maritime craft and drones are not diverted or hijacked by malicious entities [2, 16].

Global navigation satellite systems such as GPS are easily spoofed [8]. GPS signals have no authentication or integrity protection, enabling them to be replayed and spoofed. An adversary can leverage an inexpensive software-defined radio such as HackRF One to transmit captured GPS signals to receivers mounted on vehicles, ships [1] and drones [20], causing them to move to the wrong locations. Additionally, malicious entities can spoof their locations for nefarious purposes. In such scenarios, GPS devices must be protected from hardware tampering, but the authenticity of the received GPS signals must also be verified to prevent location spoofing.

This chapter proposes a novel and efficient approach to verify the authenticity of GPS signals by continuously computing hash-based message authentication codes (HMACs) of GPS navigation messages received from satellites using the chameleon hash keychain. The generated HMACs serve as fingerprints for GPS receivers mounted on vehicles, ships, drones and mobile devices, enabling their GPS signals to be authenticated in real time. The approach uses a new key to protect every frame of a navigation message. Verifying each frame using a different key in the keychain renders GPS signal tampering and spoofing very difficult. GPS receivers can easily verify keys in the keychain and use the authenticated keys to compute the HMACs of GPS subframes to perform signal authentication.

A principal advantage of the approach is near real-time, secure GPS authentication using the unbounded one-way chameleon hash keychain. Another advantage is fast and efficient authentication of GPS signals by synchronizing the keychain without a public-key infrastructure and GPS message modification. Additionally, the GPS signal authentication is readily supported by existing network infrastructures (e.g., IP networks) without deploying additional satellites.

2. Background and Related Work

This section discusses GPS signals [22] and chameleon hashing [13]. Also, it describes related work on location spoofing detection.

2.1 GPS Signals

Figure 1 shows the GPS L1 C/A navigation message structure. Each satellite transmits a continuous stream of data at 50 bits per second. The data contains the system time, clock correction values, satellite orbital data (ephemeris), orbital data of all the other satellites (almanac) and

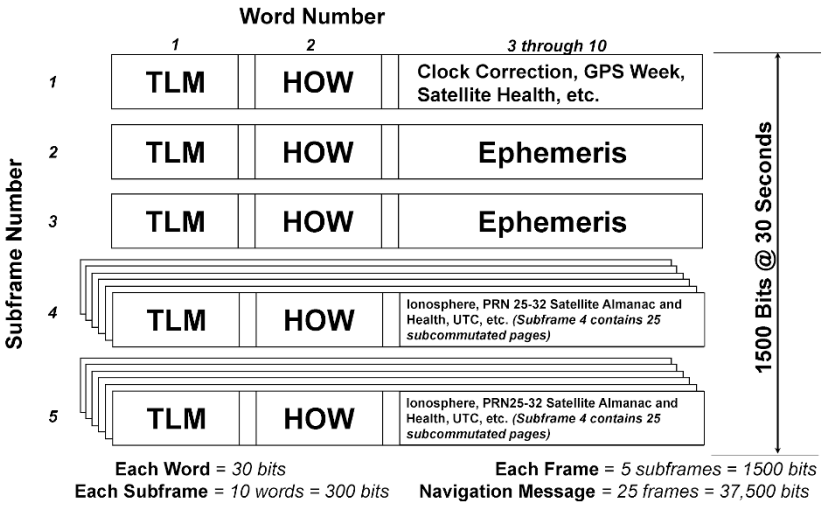

Figure 1. GPS L1 C/A navigation message structure.

satellite system health. The data is grouped into units called frames or pages. A navigation message comprises 25 frames. Each frame contains 1,500 bits and is divided into five subframes, each containing 300 bits. Transmission time from a satellite to a GPS receiver is six seconds. An entire navigation message is transmitted in 12.5 minutes. GPS navigation messages are not protected using cryptography and are, therefore, susceptible to spoofing attacks.

2.2 Chameleon Hashing

Chameleon hashing [13] is a trapdoor collision resistant function that is associated with a public-private key pair. The function is easy to compute in one direction, but very difficult to compute in the reverse direction without the private key (trapdoor). The private key holder can easily detect a collision for every input and can change the input value while computing the same output hash value. Chameleon constructs are based on discrete logarithms and elliptic curve cryptography [12].

Chameleon hashing has been used to secure data integrity in advanced metering infrastructures [21] and industrial control systems [10, 11]. Data tampering is detected by verifying the chameleon hash value.

2.3 Related Work

Liu et al. [14] have proposed a non-cryptographic scheme for computing pseudorange differences to detect meaconing and simple and interme-

diate spoofing attacks. A signal pseudorange model based on the signal transmission path is constructed to establish the double-difference of the pseudoranges of adjacent epochs. By applying the Taylor expansion to the position relationship between a satellite and GPS receiver or spoofer, the authenticity of a signal can be verified by comparing the result of the spoofing detection algorithm against the result of the traditional least squares method.

Other signal-based spoofing detection approaches employ antenna arrays [9, 23], receiver pseudorange or carrier phase differences [3, 19] or correlation methods [4, 18]. However, these signal-based anti-spoofing methods are often inaccurate due to environmental conditions.

Cryptography schemes have also been proposed to protect global navigational satellite system signals. Wu et al. [25] proposed an anti-spoofing scheme for BeiDou-II navigation messages using the Chinese SM cryptographic standards for message encryption. The integrity of navigation messages is protected by inserting spread spectrum information between subframes 1 and 2 in D2 navigation messages. However, this requires modifications to the navigation message format, which increases the difficulty of deployment. Other schemes use RSA or the Elliptic Curve Digital Signature Algorithm (ECDSA) to secure navigation messages and broadcast the authenticated signals as QZSS L1-SAIF navigation messages [15] or GPS civil navigation (CNAV) messages [24].

The effectiveness of broadcast authentication has led to the use of TESLA [17] to secure navigation messages. In the scheme of Fernandez-Hernandez et al. [6], senders encrypt their navigation messages using TESLA and receivers only need to wait for the senders to reveal the keys in order to authenticate the signals. Ghorbani et al. [7] applied TESLA to GPS L1 C/A navigation messages, but this requires the protocol to be implemented for civil navigation (CNAV-2) signals of GPS L1C. Yuan et al. [26] used ECDSA in conjunction with TESLA to secure BeiDou civil navigation signals; ECDSA ensures signal reliability while TESLA is efficient for broadcast authentication. Although a one-way keychain is very efficient, TESLA is a bounded keychain and, therefore, requires a new keychain to be set up after its keys are exhausted. Additionally, TESLA relies on loosely-synchronized time between a server and receivers to ensure data authenticity. The approach proposed in this chapter addresses these weaknesses using an unbounded keychain.

Table 1. Chameleon hash keychain notation.

$\mathbb{CH}$	Chameleon hash function
$\mathbb{CH}'$	Trapdoor chameleon hash function
$\mathbb{K}$	Trapdoor key
HK	Chameleon hash value or hash key
m	Input message to $\mathbb{CH}$
m'	Message $m' \neq m$
r	Input random prime number to $\mathbb{CH}$
r'	Collision resulting from $\mathbb{CH}'$

3. GPS Signal Authentication

This section presents the threat model and assumptions, along with the chameleon hash keychain. Also, it presents the proposed GPS authentication protocol using an unbounded chameleon hash keychain.

3.1 Threat Model and Assumptions

An adversary intends to spoof the locations of automobiles, ships, drones and mobile devices by transmitting spoofed or replayed GPS signals. The following assumptions are adopted in this work:

- An adversary has access to a software-defined radio or low-cost radio device to capture GPS signals from satellites and broadcast spoofed GPS signals.
- An adversary must use a software-defined radio or low-cost radio device with a transmitter that is proximal to a victim's device in order to launch a location spoofing attack.
- An adversary does not need access to the hardware or software of a victim's GPS receiver to tamper with GPS signals.
- An adversary does not need to control a victim's device in order to launch a location spoofing attack.

3.2 Chameleon Hash Keychain

This section describes the chameleon hash keychain, an unbounded one-way keychain with chameleon hashing that provides fast, efficient authentication between two parties [5]. Table 1 shows the notation used to discuss details about the chameleon hash keychain.

A one-way unbounded chameleon hash keychain $HK_0 \rightarrow HK_1 \rightarrow HK_2 \rightarrow ... \rightarrow HK_\infty$ is generated using a random message m_0 and

random prime number r_0 by computing HK_n with $n = 0$:

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

The principal property of chameleon hashing is that a hash collision is computed easily when the trapdoor key $\mathbb{K}$ is known. Specifically, it is feasible to derive a pair of messages (m', r') that when hashed using the chameleon hash function yields the same chameleon hash value (hash key):

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

where $m \neq m'$ and $r \neq r'$.

Each subsequent hash key HK_n ($n = 1, 2, ...$) is generated using Equation 1 with parameters m_n and r_n. The keychain is formed by linking a new hash key HK_{n+1} with its previous hash key HK_n by finding a collision. This is accomplished by computing the corresponding r' using the trapdoor key $\mathbb{K}$:

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

where $m'_n = HK_{n+1}$.

The resulting relationship between two consecutive hash keys in the keychain HK_n and HK_{n+1} generated using (m_{n+1}, r_{n+1}) is given by:

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

It is easy to determine the authenticity of HK_{n+1} and r'_n because they yield a hash value equal to the previous hash key HK_n.

Note that $\mathbb{CH}$ is a public function that does not require knowledge of $\mathbb{K}$. Thus, anyone can verify the authenticity of a hash key in the chain, but it is very difficult to derive future hash keys. Additionally, the two communicating parties do not need to synchronize their clocks because new hash keys can be revealed on demand and verified immediately.

3.3 Architecture Overview

Figure 2 shows the proposed GPS signal authentication architecture to combat GPS location spoofing. Because GPS L1 C/A signals are not protected, multiple land-based GPS authentication servers using a chameleon hash keychain are deployed to secure GPS navigation messages transmitted by satellites. All the navigation message frames received are protected by HMACs computed using the latest hash keys in the keychain every 30 seconds. This is done by the main authentication

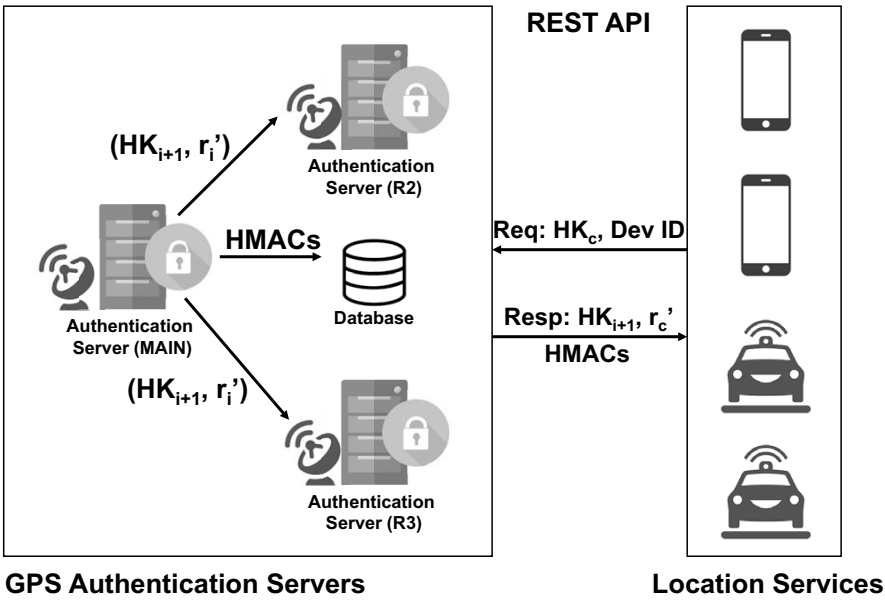

Figure 2. GPS signal authentication architecture.

server, which is housed in a secure facility. Other authentication servers located at different locations can verify the HMACs based on the latest hash keys obtained from the main authentication server. If HMAC verification fails, then the GPS signals of one or more authentication servers have been spoofed.

The proposed GPS authentication protocol has three phases:

- **Hash Key Generation and Distribution:** The chameleon hash keychain is set up on the GPS authentication server, which distributes hash keys HK to clients periodically.
- **GPS Navigation Message Protection:** The GPS authentication server computes the HMACs of message subframes using the latest hash keys in real time.
- **GPS Signal Verification:** The clients verify the hash keys and the GPS signals received from satellites by verifying the HMACs with the GPS authentication server.

Any location-based service can execute the proposed GPS signal authentication protocol to verify the GPS authentication server. The location-based service verifies the GPS signals it receives by checking

Table 2. GPS signal authentication protocol notation.

$\mathbb{SF}$	Subframe of a GPS navigation message
$\mathbb{E}$	Elliptic curve used by the chameleon hash function
G	Base point of the NIST prime curve `secp256r1`
Y	Elliptic curve cryptography public key
y	Elliptic curve cryptography private key (trapdoor key $\mathbb{K}$)

the chameleon hashes of the signals using a representational state transfer (REST) interface. Table 2 presents the notation used to specify the GPS signal authentication protocol.

Hash Key Generation and Distribution. The GPS authentication server generates and maintains the chameleon hash keychain that secures GPS navigation messages. The NIST prime curve P-256 E of the form $y^2 \pmod p = x^3 + ax + b \pmod p$ on the finite field F_p is used to construct the chameleon hash function. First, the elliptic curve cryptography domain parameters are generated, where p is a large prime number, a and b are elliptic curve coefficients and G is a generator selected from the elliptic curve. A random value y is chosen as the private key (trapdoor key) and the corresponding public key is computed as $Y = yG$.

The first hash key HK_0 is generated using a random message m_0 and random prime number 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 input key Y).

Whenever 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 as follows:

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

Note that only the GPS authentication server can generate a hash collision to obtain r'_n because it is the only entity that possesses the private key y (trapdoor key). HK_{n+1} and r'_n are then distributed to all the clients.

A location-based service that requires GPS authentication can bootstrap its application by registering the device and obtaining the initial

hash key or the latest hash key from the GPS authentication server via a TLS session.

GPS Navigation Message Protection. The proposed authentication protocol secures GPS navigation messages received from satellites. Each GPS message frame comprising five subframes is protected using a different hash key in the chameleon hash keychain.

The GPS authentication server is set up with a dedicated GPS signal receiver to collect navigation messages from all the GPS satellites that are detected. This server should be housed in a security facility to prevent attackers from spoofing GPS signals. Upon receiving a complete frame comprising five subframes, the server computes the next hash key in the chameleon hash keychain using Equation 5, where m is the concatenation of subframes 1 to 5 of each message frame and r is a random prime number that is generated securely by the authentication server.

The GPS authentication server executes the following steps:

- HK_0 is generated using a random message m_0 and random prime number r_0, i.e., $HK_0 = \mathbb{CH}(m_0, r_0)$. The GPS authentication server maintains the chameleon hash keychain, generating a hash key every 30 seconds. The current hash key is HK_i.

- The five subframes of a received GPS navigation message are concatenated to create next message m_{i+1}. For example, when $i = 0$, $m_1 = \mathbb{SF}_{1,1} \cdot \mathbb{SF}_{1,2} \cdot \mathbb{SF}_{1,3} \cdot \mathbb{SF}_{1,4} \cdot \mathbb{SF}_{1,5}$.

- The next hash key HK_{i+1} is generated via Equation 5 by selecting a random prime number r_{i+1}. For example, $HK_1 = \mathbb{CH}(m_1, r_1)$.

- The private key y is used to compute r'_i via Equation 6 in order to link HK_{i+1} and HK_i. For example, $HK_0 = \mathbb{CH}(HK_1, r'_0)$.

- The HMAC of each subframe is computed using SHA-256 and the new hash key HK_{i+1}. The HMACs are stored in a database to facilitate GPS verification requests by clients.

Algorithm 1 summarizes the chameleon hash keychain generation and GPS message protection computations performed by the GPS authentication server. The hash key is used to compute the HMAC fingerprint of every message subframe that is later 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 an attacker to spoof GPS signals without being detected.

Algorithm 1 : Hash keychain generation and GPS message protection.

Initialize public key: $Y = yG$
Choose a random message m_0 and random prime number r_0
Generate initial $HK_0 = HMAC(m_0, Y) \cdot Y + r_0 \cdot G$
$i = 0$
for each complete message frame m_{i+1} received **do**
  $//m_{i+1} = \mathbb{SF}_{i+1,1} \cdot \mathbb{SF}_{i+1,2} \cdot \mathbb{SF}_{i+1,3} \cdot \mathbb{SF}_{i+1,4} \cdot \mathbb{SF}_{i+1,5}$
  Generate a random prime number r_{i+1}
  $HK_{i+1} = HMAC(m_{i+1}, Y) \cdot Y + r_{i+1} \cdot G$
  $r'_i = y \cdot [HMAC(m_i, Y) - HMAC(HK_{i+1}, Y)] + r_i$

  //Generate HMAC for each subframe $\mathbb{SF}_{i+1,j}$ using HK_{i+1}
  for j=1; j<6; j++ **do**
    $HMAC_{i+1,j} = HMAC(\mathbb{SF}_{i+1,j}, HK_{i+1})$
    Store $HMAC_{i+1,j}$ in the database
  end for

  i++
end for

GPS Signal Verification. Location-based service clients verify GPS navigation messages received from satellites by first synchronizing the hash key with the GPS authentication server as follows:

$$\begin{array}{lcll} \text{Client} & \rightarrow & \text{Server:} & HK_c, Device_{ID} \\ \text{Server} & \rightarrow & \text{Client:} & HK_{i+1}, r'_c \end{array}$$

A client possessing a hash key $HK_c \neq HK_i$ sends its $Device_{ID}$ and HK_c to the GPS authentication server to obtain the next hash key HK_{i+1}. The server computes r'_c for the requesting client using Equation 6 such that $HK_c = \mathbb{CH}(HK_{i+1}, r'_c)$. The server then sends HK_{i+1} and r'_c to the client, enabling the client to authenticate the server. When this is successful, the client can maintain a synchronized chameleon hash chain with the server to continuously authenticate GPS signals received during a session.

To verify a GPS navigation message, a client sends a request to the GPS authentication server to obtain the HMACs of each message frame by indicating the *Satellite ID* and *Frame ID*. The client also receives the message subframes directly from a satellite and uses the next hash key HK_{i+1} to verify the HMAC of each subframe. The GPS data is authentic if the computed HMACs match the HMACs obtained from the GPS authentication server.

Algorithm 2 : GPS message authenticity verification.

Client maintains the chameleon hash keychain: $HK_0 \rightarrow HK_1 \rightarrow ... \rightarrow HK_i$
Authenticate with the GPS authentication server
Synchronize the keychain
Receive HK_{i+1} and r'_i from the GPS authentication server
Verify HK_{i+1} such that $HK_i = \mathbb{CH}(HK_{i+1}, r'_i)$

for each complete message frame m_{i+1} received **do**
  $//m_{i+1} = \mathbb{SF}_{i+1,1} \cdot \mathbb{SF}_{i+1,2} \cdot \mathbb{SF}_{i+1,3} \cdot \mathbb{SF}_{i+1,4} \cdot \mathbb{SF}_{i+1,5}$
  Request HMACs from the GPS authentication server for $\mathbb{SF}_{i+1,j}$, $j = 1, 2, ..., 5$
  Compute $HMAC'_{i+1,j} = HMAC(\mathbb{SF}_{i+1,j}, HK_{i+1})$, $j = 1, 2, ..., 5$

  for j=1; j<6; j++ **do**
    if $HMAC_{i+1,j} == HMAC'_{i+1,j}$ **then**
      Subframe $\mathbb{SF}_{i+1,j}$ is authenticated
    else
      Subframe $\mathbb{SF}_{i+1,j}$ is spoofed
    end if
  end for

end for

Algorithm 2 summarizes the steps involved in GPS message authentication by a client. The steps in the verification process are:

- The client synchronizes the keychain with the GPS authentication server.
- The client obtains the current hash key HK_i and verifies that the key is authentic, thereby authenticating the GPS authentication server.
- After the keychain is synchronized, the client authenticates the server every 30 secs by verifying that $HK_i = \mathbb{CH}(HK_{i+1}, r'_i)$.
- The client requests the HMACs of each satellite frame m_{i+1} comprising five subframes $\mathbb{SF}_{i+1,1}$, $\mathbb{SF}_{i+1,2}$, ..., $\mathbb{SF}_{i+1,5}$ from the server.
- For all the subframes in frame m_{i+1}, the client computes the corresponding $HMAC(\mathbb{SF}_{i+1,j}, HK_{i+1,j})$ where $j = 1, 2, ..., 5$ and verifies that they all match the respective HMACs provided by the GPS authentication server.

It is crucial that clients ensure the authenticity of hash keys so that the HMAC verification can be trusted. The authenticity of hash keys is provided by the unique chameleon hash keychain property that the $\mathbb{CH}$

of the current hash key equals the previous hash key (Equation 6). Since the HMAC computations are fast, clients can verify a GPS navigation message in every 30 second interval.

4. Prototype Implementation

A proof-of-concept prototype was constructed to validate the proposed GPS signal authentication protocol. The GPS authentication server running a Linux Ubuntu operating system was deployed in the Amazon Elastic Compute Cloud (EC2). The GPS authentication service was operational 24/7 and was accessible on the Internet.

An Android smartphone was used as a GPS signal receiver for the GPS authentication server, which enabled raw GPS navigation messages to be obtained directly from satellites. In a real deployment, a proper GPS receiver should be used. `Node.js` and `Express.js` were used to develop a representational state transfer API that enabled location-based applications and clients to request authenticated GPS signals from the authentication server. GPS navigation messages were received continuously with one subframe received every six seconds. Since each message frame contained five subframes, it took about 30 seconds to collect the five subframes, after which the GPS authentication server computed the subframe HMACs that were stored in a Mongo DB Atlas cloud database.

A C library was developed for chameleon hashing based on elliptic curve cryptography. The OpenSSL library and Bouncy Castle cryptography library were employed. The two main constructs included:

- `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)`

The implementation employed the NIST prime curve P-256. The chameleon hash keychain implementation generated chameleon hash values HK_n and computed collisions r'_n.

An Android application was developed to obtain raw GPS navigation messages using the Android location library. Since GPS navigation messages are continuous streams of data transmitted by satellites, the implementation used `GnssNavigationMessage.Callback()` to listen for event changes. The `onGnssNavigationMessageReceived()` and `onStatusChanged()` functions were triggered to retrieve the raw GPS data received by the smartphone.

Table 3. Navigation signals supported by Samsung Note 8 and Note S8+ models.

Samsung Model	Android Version	Navigation Messages	Accumulated Delta Range	Supported Signals
Note 8	9.0	Yes	Yes	GPS, GLO, GAL, BDS
Note S8+	9.0	Yes	Yes	GPS, GLO, GAL, BDS, QZS

Samsung Note 8 and S8+ Android smartphone models were used to obtain the raw GPS data. One smartphone functioned as the GPS receiver for the GPS authentication server while the other functioned as a client that attempted to verify the received GPS navigation messages. Table 3 shows the signals that can be retrieved by the Samsung Note 8 and Samsung Note S8+ model smartphones.

5. Evaluation Results and Discussion

This section discusses the evaluation results related to execution time, communications overhead and security aspects.

5.1 Execution Time

Since there are 31 satellites in the GPS constellation, it was important that the GPS authentication server generates hash keys concurrently (although not all the GPS satellites could be detected at a given location).

Table 4. Chameleon hash keychain function and HMAC computation times.

Computation	Time
Chameleon hash key generation: $HK = \mathbb{CH}(m, r)$	137.2 μs
Collision generation: $r'_i = \mathbb{CH}'(y, m_i, r_i, HK_{i+1}))$	20.1 μs
HMAC computation: $HMAC(m, Y)$	10.7 μs

Table 4 shows the average times over 1,000 sequential executions of the chameleon hash keychain generation, collision generation and HMAC computation functions. Hash key generation took 137.2 μs and computing r'_i for collision generation took 20.1 μs. Chameleon hash key generation required more time than collision generation due to its two elliptic curve point multiplication operations compared with one point multiplication for collision generation. The $HMAC(m, Y)$ function was fast and efficient, requiring only 10.7 μs to generate five HMACs.

5.2 Communications Overhead

The proposed GPS signal authentication protocol incurs less overhead for a large number of clients compared with the conventional ECDSA digital signature approach that generates a signature for each GPS message subframe. Assuming that an ECDSA signature is 64 bytes, a complete GPS navigation message contains 25 frames (i.e., 125 subframes) and incurs an overhead of 7.81 KB per navigation message per client. In contrast, the HMAC in the GPS signal authentication protocol is 32 bytes and the hash key HK_{i+1} and r'_i are 64 bytes each. The hash key is renewed every frame, which incurs an overhead of 3.12 KB and the HMACs for 125 subframes incur an overhead of 3.91 KB. The total overhead per navigation message per client is 7.03 KB, which is 10% less than the overhead in the ECDSA scheme. Each subframe has to be signed or its HMAC computed separately because some receivers may not have received all the subframes. Therefore, each subframe needs to be verified individually.

5.3 Security Aspects

Based on the trapdoor collision property of chameleon hash functions, an efficient probabilistic polynomial time algorithm exists such that, upon receiving a private key y, message pair (m, r) and m', the GPS authentication server outputs a value r' such that a hash collision occurs (i.e., $\mathbb{CH}(m', r') = \mathbb{CH}(m, r)$). The m' and r' values are sent to clients that wish to verify GPS signals, advance their chameleon hash keychains and renew their hash keys. If a client cannot verify the pair $(m',\ r')$ it receives, then the new hash key did not originate from the GPS authentication server.

The proposed authentication protocol also has the collision resistant property. No probabilistic polynomial time algorithm exists that, upon input of a hash key HK and without the knowledge of the private key y, would enable the GPS authentication server to find pairs (m, r) and (m', r') where $m \neq m'$ such that $\mathbb{CH}(m', r') = \mathbb{CH}(m, r)$ has a non-negligible probability. This is equivalent to solving the elliptic curve discrete logarithm problem (ECDLP), which is known to be computationally hard. The significance of the collision resistant property is that no entity apart from 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 would be very 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, the authenticity guarantees of the GPS navigation messages increase, but this comes with the overhead of distributing the hash keys more frequently to all the clients.

6. Conclusions

This chapter has presented a novel approach using a chameleon hash keychain to efficiently protect GPS navigation messages, enabling clients to verify their GPS signals via a web service interface. The approach adopts an unbounded one-way keychain generated using chameleon hash constructs, providing the ability to use a new hash key to protect every frame of a GPS navigation message. The resulting GPS signal authentication protocol is effective because clients can authenticate the GPS authentication server easily by verifying the one-way property of new hash keys they receive. The approach also eliminates the need to loosely synchronize time between the GPS authentication server and clients.

This research has conducted a preliminary evaluation of the prototype implementation. The next step is to work with government agencies to roll out a larger deployment to investigate network latency and system scalability. In a real deployment, the GPS authentication server will have to be hardened and integrated with adequate web security protection measures. Future research will also investigate extensions of the GPS authentication protocol to protect other global navigational satellite systems such as GLO, GAL and BeiDou.

References

[1] J. Bhatti and T. Humphreys, Hostile control of ships via false GPS signals: Demonstration and detection, *Journal of the Institute of Navigation*, vol. 64(1), pp. 51–66, 2017.

[2] M. 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*, pp. 153–155, 2020.

[3] 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.

[4] 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(3), pp. 475–487, 2015.

[5] 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.

[6] I. Fernandez-Hernandez, V. Rijmen, G. Seco-Granados, J. Simon, I. Rodriguez and J. Calle, A navigation message authentication proposal for the Galileo Open Service, *Journal of the Institute of Navigation*, vol. 63(1), pp. 85–102, 2016.

[7] K. Ghorbani, N. Orouji and M. Mosavi, Navigation message authentication based on a one-way hash chain to mitigate spoofing attacks on GPS L1, *Wireless Personal Communications*, vol. 113(4), pp. 1743–1754, 2020.

[8] T. Humphreys, B. Ledvina, M. Psiaki, B. O'Hanlon and P. 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.

[9] K. Jansen, N. Tippenhauer and C. Popper, Multi-receiver GPS spoofing detection: Error models and realization, *Proceedings of the Thirty-Second Annual Computer Security Applications Conference*, pp. 237–250, 2016.

[10] S. Keoh, K. Au and Z. Tang, Securing industrial control systems: An end-to-end integrity verification approach, presented at the *Industrial Control System Security Workshop of the Thirty-First Annual Computer Security Applications Conference* (`www.acsac.org/2015/workshops/icss`), 2015.

[11] S. Keoh, H. Tan and Z. Tang, Authentication and integrity protection for real-time cyber-physical systems, in *Handbook of Real-Time Computing*, Y. Tian and D. Levy (Eds.), Springer, Singapore, chapter 3, 2020.

[12] N. Koblitz, Elliptic curve cryptosystems, *Mathematics of Computation*, vol. 48(177), pp. 203–209, 1987.

[13] H. Krawczyk and T. Rabin, Chameleon Hashing and Signatures, *Cryptography ePrint Archive*, Report 1998/010 (`eprint.iacr.org/1998/010`), 1998.

[14] K. Liu, W. Wu, Z. Wu, L. He and K. Tang, Spoofing detection algorithm based on pseudorange differences, *Sensors*, vol. 18(10), article no. 3197, 2018.

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

[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 no. 12, 2019.

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

[18] M. Psiaki, B. O'Hanlon, J. Bhatti, D. Shepard and T. 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. Radin, P. Swaszek, K. Seals and R. Hartnett, GNSS spoof detection based on pseudoranges from multiple receivers, *Proceedings of the Twenty-Eighth 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. Tan, K. Lim, S. Keoh, Z. Tang, D. Leong and C. Sum, Chameleon: A blind double trapdoor hash function for securing AMI data aggregation, *Proceedings of the Fourth IEEE World Forum on the Internet of Things*, pp. 225–230, 2018.

[22] J. Van Sickle and J. Dutton, GEOG862: GPS and GNSS for Geospatial Professionals, Department of Geography, Pennsylvania State University, University Park, Pennsylvania (`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 Integrated Communications, Navigation and Surveillance Conference*, 2015.

[24] K. Wesson, M. Rothlisberger and T. Humphreys, A proposed navigation message authentication implementation for civil GPS antispoofing, *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: A scheme for 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.