Cryptographic Authentication for GPS Communications

AA 272 Global Positioning System

Jiayang Wang
Yonglin He
Yiyang Mu

December 5th, 2023

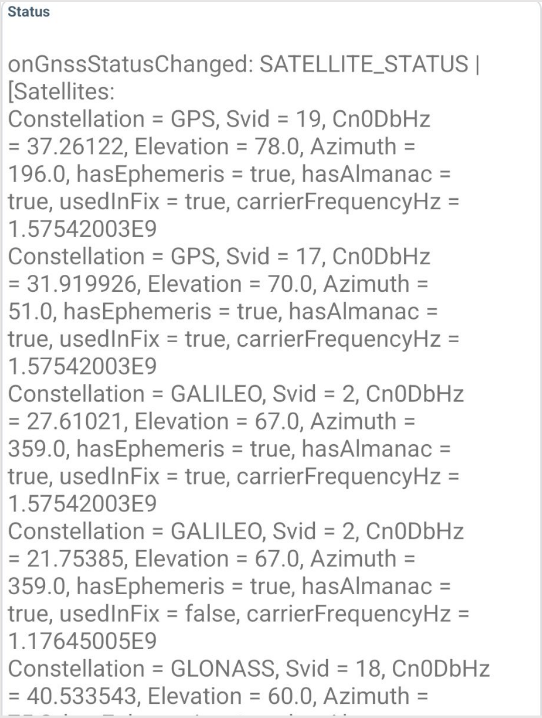
Stanford University

Overview

- Motivation
- Problem Statement
- Approach
- Demonstration
- Results and Findings
- Improvements

Motivations

- Security concerns associated with GPS communications
- Integrity of location-based services
- Potential threats to data accuracy



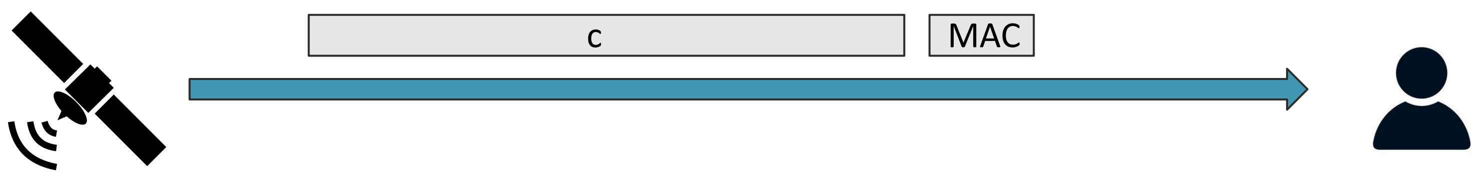
GnssLogger, Google Pixel 6

Motivations

- Security concerns associated with GPS communications
- Integrity of location-based services
- Potential threats to data accuracy
- **Solution: Authentication**

Cryptography - Symmetric Message Authentication

- Used for “Integrity”, not “confidentiality”
  - In GPS applications, this prevents GPS signal spoofing or message alteration.



tag = S(k, c)

Verify(k, c, tag) -> yes/no

MAC: Message Authentication Code

- CBC-MAC
- HMAC
- …

Problems

- Problem 1: **Key Lost**

- Problem 2: **Replay Attack**

- **Definition**: Intercepting secure network communication to manipulate the recipient.
- **Process**: Cybercriminal captures and fraudulently resends or delays the message.
- **Risk**: Advanced decryption skills not required; success achieved through message replay.

Approach: TESLA Algorithm

- Key Chain: $[K_0, K_1, K_2, K_3, \ldots, K_N]$

- Establish a key for each session, which is a type of code that is only valid for one transaction and can't be used again.

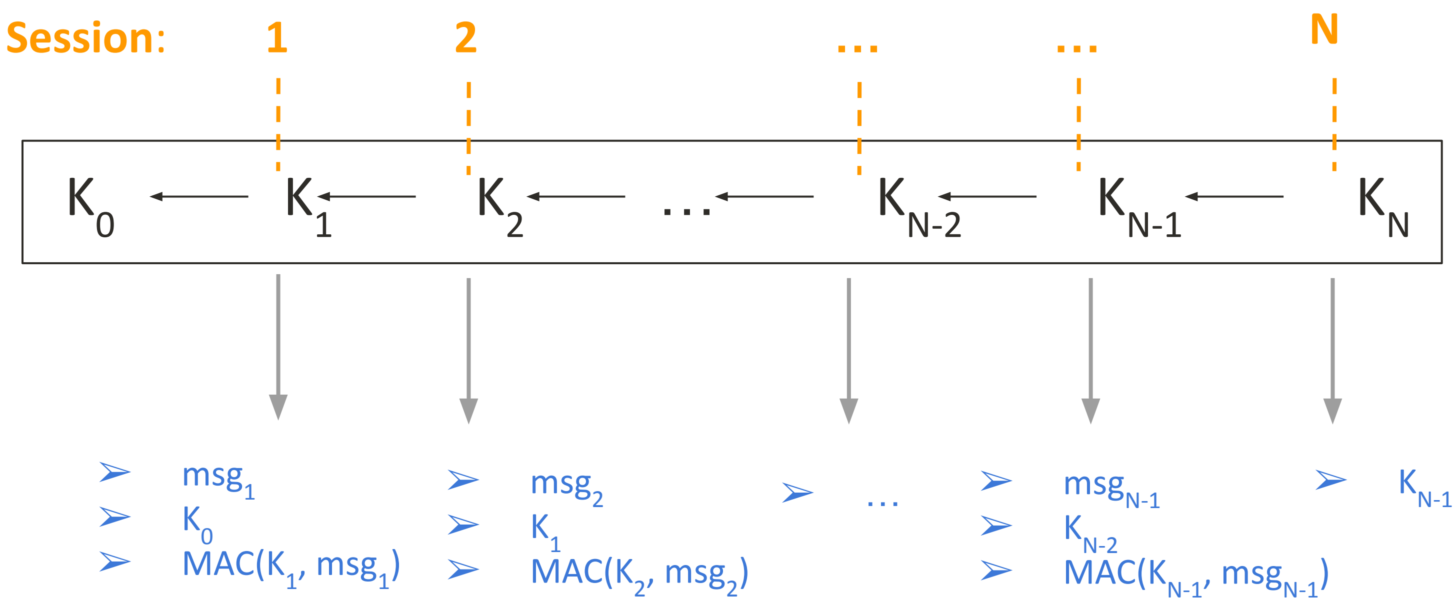
$$K_0 \longleftarrow K_1 \longleftarrow K_2 \longleftarrow \ldots \longleftarrow K_{N-2} \longleftarrow K_{N-1} \longleftarrow K_N$$

- $K_{N-1} = F(K_N)$,
- $K_{N-2} = F(K_{N-1})$
- …
- $K_0 = F(K_1)$

- F : Secure Hash Algorithm 256-bit (**SHA-256**)

Approach: TESLA Algorithm

- **Key Distribution is delayed by one session.**



Do we take K_o for granted?

Approach: TESLA Algorithm

- **Key Distribution is delayed by one session.**

Session: 1 2 ... ... N

$$K_0 \leftarrow K_1 \leftarrow K_2 \leftarrow \ldots \leftarrow K_{N-2} \leftarrow K_{N-1} \leftarrow K_N$$

Session 1:
- msg_1
- K_0
- $MAC(K_1, msg_1)$
- $sign(sk, K_0)$

Session 2:
- msg_2
- K_1
- $MAC(K_2, msg_2)$

...:
- ...

Session N-1:
- msg_{N-1}
- K_{N-2}
- $MAC(K_{N-1}, msg_{N-1})$

Session N:
- K_{N-1}

Approach: TESLA Algorithm

Algorithm 1 TESLA Sender

Generate key chain: $[K_0, K_1, K_2, ..., K_N]$
for $i \leftarrow 1$ **to** N **do**
  $mac_i \leftarrow MAC(K, msg_i)$
  Send (msg_i, K_{i-1}, mac_i)
  if $i = 1$ **then**
    $sig \leftarrow Sign(sk, K_0)$
    Send (sig)
  end if
end for

Algorithm 2 TESLA Receiver

msg_{prev}, mac_{prev}
for $i \leftarrow 1$ **to** N **do**
  $msg \leftarrow msg_i$
  $K_{prev} \leftarrow K_{i-1}$
  $mac \leftarrow mac_i$
  if $i = 1$ **then**
    verify (pk, sig, K_0)
  else
    verify $(K_{prev}, msg_{prev}, mac_{prev})$
  end if
  $msg_{prev} \leftarrow msg$
  $mac_{prev} \leftarrow mac$
end for

How do we validate public key (pk)?

Solution 1:

Every receiver stores pk(s) for all GPS satellites.

- Cons:
  - requires pk integrity at each receiver
  - requires timely updates when new satellites are launched

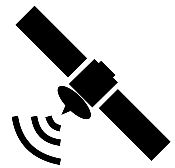
Better Solution:

- Use certificates issued by trusted third parties to deliver pk(s)

Secret Key & Public Key

- $(pk_B, sk_B) \leftarrow Gen()$
- $sig_B = sign(sk_B, pk)$
- $cert = (pk, sig_B)$
-

Secret Key & Public Key

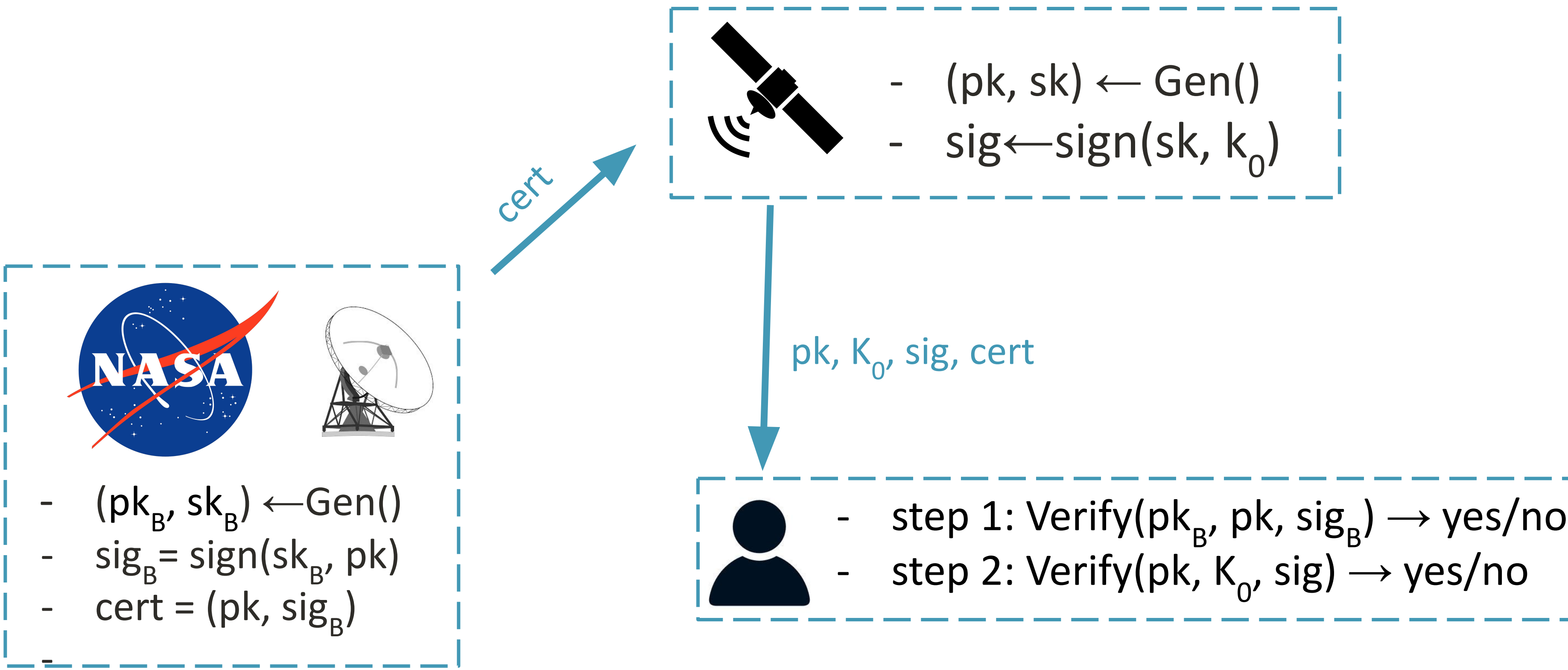


- $(pk, sk) \leftarrow Gen()$
- $sig \leftarrow sign(sk, k_0)$

cert

- $(pk_B, sk_B) \leftarrow Gen()$
- $sig_B = sign(sk_B, pk)$
- $cert = (pk, sig_B)$
-

Secret Key & Public Key



Implementation

- **Sender**

```python
key_auth = key_chain[session]
key_send = key_chain[session-1].hex()
print("Entered session: ", session)
file_name = input("Enter filename to send: ")
with open(file_name, 'rb') as file:
    file_data_b = file.read()
    file_data = file_data_b.decode('utf-8')
hm = hmac.HMAC(key_auth, hashes.SHA256())
hm.update(file_data.encode('utf-8'))
mac = hm.finalize().hex()
if session == 1:
    signature = private_key.sign(
        key_send.encode('utf-8'),
        padding.PSS(
            mgf=padding.MGF1(hashes.SHA256()),
            salt_length=padding.PSS.MAX_LENGTH
        ),
        hashes.SHA256()
    )
    sndr_socket.send((key_send+";"+file_data+";"+mac+";"+pem_public_key.hex()+";"+signature.hex()).encode('utf-8'))
else:
    sndr_socket.send((key_send+";"+file_data+";"+mac).encode('utf-8'))
print("Key sent in session", session, "is", key_send)
print("Just info: Message", file_data, "can only be authenticated with key", key_auth.hex())
print("Session", session, "finished.")
```

Implementation

- **Receiver**

```python
# Verify the prev_msg with received key
hm = hmac.HMAC(key, hashes.SHA256())
hm.update(prev_msg.encode())
try:
    hm.verify(prev_mac)
    print("Message received in session", session-1, prev_msg, "is authenticated.")
    prev_msg = msg
    prev_key = key
    prev_mac = mac
    print("Session", session, "finished.")
except InvalidSignature:
    print("MAC is not valid. Messaged is spoofed.")
    print("Received message", msg, "is discarded.")
    session -= 1
    print("Downgraded session to id", session, "to sync with gps.")
```

Implementation

- **Certification**

```
with open(sat_pk_path, 'rb') as sat_pk_file:
    pem_public_key_sat = sat_pk_file.read()

public_key_sat = serialization.load_pem_public_key(
    pem_public_key_sat,
    backend=default_backend()
)

issuer = x509.Name([
    x509.NameAttribute(NameOID.COMMON_NAME, u"nasa"),
])
certificate = x509.CertificateBuilder().subject_name(
    issuer
).issuer_name(
    issuer
).public_key(
    public_key_sat
).serial_number(
    x509.random_serial_number()
).not_valid_before(
    datetime.datetime.utcnow()
).not_valid_after(
    # Valid for 1 year
    datetime.datetime.utcnow() + datetime.timedelta(days=365)
).sign(private_key_nasa, hashes.SHA256(), default_backend())

# Serialize certificate
pem_certificate = certificate.public_bytes(encoding=serialization.Encoding.PEM)

cert_path = "cert.pem"
with open(cert_path, 'wb') as cert_file:
    cert_file.write(pem_certificate)
    print("Certificate generated for satellite public key")
```

Demonstration

Github Repo: https://github.com/jw4149/GPS_Project/tree/main

```
(base) jiayangwang@DNa80d9da code % python satellite.py
Entered session:  1
Enter filename to send: data1.txt
Key sent in session 1 is af069be81c78a8a55f096c4016d2d472af1131a884ce9f29f9a7068e7889500f
Just info: Message gps_millis:1273611504445.0,gnss_id:gps,sv_id:2,corr_pr_m:20774577.36402971
,x_sv_m:-11867755.90150921,y_sv_m:-21956297.905915044,z_sv_m:9801130.518319722
 can only be authenticated with key ccc45cd05b2e6ce787d6bd926865a5f7ba7fbcb8b5d55cc606927ce1f
1d82621
Session 1 finished.
Entered session:  2
Enter filename to send: data2.txt
Key sent in session 2 is ccc45cd05b2e6ce787d6bd926865a5f7ba7fbcb8b5d55cc606927ce1f1d82621
Just info: Message gps_millis:1273611506892.0,gnss_id:gps,sv_id:2,corr_pr_m:20774577.36402971
,x_sv_m:-11867767.90150921,y_sv_m:-21956299.905915044,z_sv_m:9801143.518319722
 can only be authenticated with key afdcdf9b061eb4f4ab09f4b51795a60ebc5902e8c60a89ebc79e15022
cf03ce9
Session 2 finished.
Entered session:  3
Enter filename to send: data3.txt
Key sent in session 3 is afdcdf9b061eb4f4ab09f4b51795a60ebc5902e8c60a89ebc79e15022cf03ce9
Just info: Message gps_millis:1273611509763.0,gnss_id:gps,sv_id:2,corr_pr_m:20774577.36402971
,x_sv_m:-11867771.90150921,y_sv_m:-21956302.905915044,z_sv_m:9801150.518319722
 can only be authenticated with key 529a5baec6c17f381965068857ce98827d32c4e04c541ccf247a5529f
b30c015
Session 3 finished.
```

```
(base) jiayangwang@DNa80d9da code % python receiver.py
Receiver ready.
Entered session:  1
Received message gps_millis:1273611504445.0,gnss_id:gps,sv_id:2,corr_pr_m:20774577.36402971,x
_sv_m:-11867755.90150921,y_sv_m:-21956297.905915044,z_sv_m:9801130.518319722
 will be verified in next session.
Session 1 finished.
Entered session:  2
Received message gps_millis:1273611506892.0,gnss_id:gps,sv_id:2,corr_pr_m:20774577.36402971,x
_sv_m:-11867767.90150921,y_sv_m:-21956299.905915044,z_sv_m:9801143.518319722
 will be verified in next session.
Message received in session 1 gps_millis:1273611504445.0,gnss_id:gps,sv_id:2,corr_pr_m:207745
77.36402971,x_sv_m:-11867755.90150921,y_sv_m:-21956297.905915044,z_sv_m:9801130.518319722
 is authenticated.
Session 2 finished.
Entered session:  3
Received message gps_millis:1273611506892.0,gnss_id:gps,sv_id:2,corr_pr_m:20774577.36402971,x
_sv_m:-11867767.90150921,y_sv_m:-21956299.905915044,z_sv_m:9801143.518319722 will be verified
 in next session.
MAC is not valid. Messaged is spoofed.
Received message gps_millis:1273611506892.0,gnss_id:gps,sv_id:2,corr_pr_m:20774577.36402971,x
_sv_m:-11867767.90150921,y_sv_m:-21956299.905915044,z_sv_m:9801143.518319722 is discarded.
Downgraded session to id 2 to sync with gps.
Entered session:  3
Received message gps_millis:1273611509763.0,gnss_id:gps,sv_id:2,corr_pr_m:20774577.36402971,x
_sv_m:-11867771.90150921,y_sv_m:-21956302.905915044,z_sv_m:9801150.518319722
 will be verified in next session.
Message received in session 2 gps_millis:1273611506892.0,gnss_id:gps,sv_id:2,corr_pr_m:207745
77.36402971,x_sv_m:-11867767.90150921,y_sv_m:-21956299.905915044,z_sv_m:9801143.518319722
 is authenticated.
Session 3 finished.
```

Improvements

- Message loss handling
- Signal-Level Authentication
- …

References

Petovello, M. (2017), GNSS Solution. Q: What is navigation message authentication?

Fernández-Hernández, I., Rijmen, V., Seco-Granados, G., Simon, J., Rodríguez, I., & Calle, J. D. (2016). A navigation message authentication proposal for the Galileo open service. *NAVIGATION: Journal of the Institute of Navigation*, *63*(1), 85-102.

Lo, S. C., & Enge, P. K. (2010, May). Authenticating aviation augmentation system broadcasts. In *IEEE/ION position, location and navigation symposium* (pp. 708-717). IEEE.

Questions?