How SIGNATURE Works on MAVLink

Author: Zichao Cong

Latest Modify Date: 3rd, Aug, 2023

MAVLink 2 adds support for message signing, which allows a MAVLink system to verify that messages originate from a trusted source.

1. SIGNATURE Format (13 bytes)

For a signed packet the 0x01 bit of the incompatibility flag field is set true and an additional 13 bytes of "SIGNATURE" data appended to the packet. The signed packet format is shown below.

Fig.4. The format of MAVLink 2 with SIGNATURE.

Note: There are two “signature” represents the whole message signing part (13 bytes) and the real signature part (6 bytes), to differ this two, the formal one would use UPPERCASE to express!

linkID (8 bits) : ID of link on which packet is sent. Normally this is the same as the *channel*.

timestamp (48 bits): Timestamp in 10 microsecond units since 1st January 2015 GMT time. This must monotonically increase for every message on a particular link.

(Note that means the timestamp may get ahead of the actual time if the packet rate averages more than 100,000 packets per second.)

signature (48 bits): A 48 bits signature for the packet, based on the complete packet, timestamp, and secret key.

1.1. Link ID:

The 8 bits link ID is provided to ensure that the signature system is robust for multi-link MAVLink systems. Each implementation should assign a link ID to each of the

MAVLink communication channels it has enabled and should put this ID in the link ID field. The link ID is especially important where there may be a significant latency difference between different links (such as WiFi combined with a telemetry radio).
The monotonically increasing timestamp rule is applied separately for each logical stream, where a stream is defined by the tuple:
(SystemID,ComponentID,LinkID)

1.2. Timestamp:

The timestamp is a 48 bits number with units of 10 microseconds since 1st January 2015 GMT.
A MAVLink-enabled device may not know the current GMT time, for example if it does not have a reliable time source, or if it has just booted and not yet obtained the time from GPS or some other system.

Systems should implement the following rules to obtain a reliable timestamp:

- The current timestamp should be stored regularly in persistent storage (ideally at least once a minute)
- The timestamp used on startup should be the maximum of the timestamp implied by the system clock and the stored timestamp
- If the system does not have an RTC mechanism then it should update its timestamp when GPS lock is achieved. The maximum of the timestamp from the GPS and the stored timestamp should be used.
- The timestamp should be incremented by one on each message sent from a particular link.
- When a correctly signed message is decoded the timestamp should be replaced by the timestamp of the incoming message if that timestamp is greater than the current timestamp.
- The timestamp on incoming signed messages should be checked against the previous timestamp for the incoming (*linkID,srcSystem,SrcComponent*) tuple and the message rejected if it is smaller.

- If there is no previous message with the given (*linkID,srcSystem,SrcComponent*) then the timestamp should be accepted if it not more than 6 million (one minute) behind the current timestamp.

1.3. Signature:

The 48 bits (6 bytes) signature is the first 48 bits of a SHA-256 hash of the complete packet (without the signature, but including the timestamp) appended to the secret key. The secret key is 32 bytes of binary data stored on both ends of a MAVLink channel (i.e. an autopilot and a ground station or MAVLink API).

This is shown below, where + represents concatenation and sha256_48() is a sha256 implementation which returns the first 48 bits of the normal sha256 output:

signature = sha256_48(secret_key + header + payload + CRC + link-ID + timestamp)

2. Secret Key

A secret key is 32 bytes of binary data that are used to create message signatures that can be verified by other holders of the key. The key should be created on one system in the network (often a GCS) and shared to other trusted devices via secure channels. Systems must have a shared key in order to be able to communicate.

3. Logging

In order to avoid leaking the secret key used for signing, systems should omit *SETUP_SIGNING* messages from logs (or replace the secret with 32 0xFF bytes in the logged message).

Similarly, signed packets should have the signature incompatibility bit cleared and the signature block removed before being put into telemetry log files. This makes it harder for potential attacker to collect large amounts of signature data with which to attack the system.

4. Process for message signing

4.1. Secret Key Generation:

The method of generating the secret key is implementation dependent. For example, it could be generated by:

- A user-entered string that is then run through SHA-256. (Normally used and would generate a Message-Digest to be transported, really similar to MD5)
- A random key generator.

Note: The secret key should be stored in persistent storage, and must not be exposed via any publicly accessible communication protocol. In particular, the key must not be exposed in MAVLink parameters, MAVLink log files or dataflash log files that may be used for public log analysis.

4.2. Share the Secret Key: The secret key may be shared to other devices using the SETUP_SIGNING message. The message should only ever be sent over a secure link (e.g. USB or wired Ethernet) as a direct message to each connected *system_id/component_id*. The receiving system must be set up to process the message and store the received secret key to the appropriate permanent storage.

SETUP_SIGNING (#256)

[Message] **(MAVLink 2)** Setup a MAVLink2 signing key. If called with secret_key of all zero and zero initial_timestamp will disable signing

Field Name	Type	Description
target_system	uint8_t	system id of the target
target_component	uint8_t	component ID of the target
secret_key	uint8_t[32]	signing key
initial_timestamp	uint64_t	initial timestamp

Fig.5. Message for setup signing.

Note: The same secure method can be used to both set and reset a system's key (reseting a key does not have to be "more secure" than setting it in the first place).

4.3. SIGNATURE Generation: three parts of the message SIGNATURE (link id, timestamp and signature) can be generated.

4.4. Messages Signing: SIGNATURE should be added at the end of each message if needed.

5. Conditions to accept or decline the packet(s):

5.1. Accepting Signed Packets:

When a signed packet arrives, it should be discarded if the:

- Timestamp is older than the previous packet from the same logical stream - where a logical stream is defined as the sequence of MAVLink packets with the same (SystemID, ComponentID, LinkID) tuple.
- Computed 48 bits signature does not match the signature included in the packet.
- The timestamp is more than 1 minute (6,000,000) behind the local system's timestamp.

5.2. Accepting Unsigned Packets:

MAVLink libraries should provide a mechanism that allows a system to conditionally accept unsigned packets.

The rules for accepting these packets will be implementation specific, but could be based on a combination of a parameter setting, transport type, message type, (in) compatibility flags etc.

Some suggestions (offered by official) for when to accept unsigned packets:

- Accept all unsigned packets based on a system-specific parameter.
- Accept all unsigned packets if the connection is over a "secure channel" (e.g. local USB cable or local wired Ethernet cable).
- RADIO_STATUS packets are always accepted without signing (to make life easier for telemetry radios).
- Accept all unsigned packets when in an "unsigned mode" (perhaps triggered by a hardware button pressed on boot).
- Accept all unsigned packets until a signed packet is received (unconditionally), then move to the more restricted signing rules above.

Note: All packets that do not meet the system-specific unsigned packet acceptance rules must be rejected (otherwise there is no benefit gained from signing/authentication).

5.3. Accepting Incorrectly Signed Packets:

MAVLink libraries should provide a mechanism that allows a system to conditionally accept incorrectly signed packets.

This feature might be useful for finding a lost vehicle with a corrupted secret key (the GCS could choose to still display position information, albeit ideally with a different "untrusted" icon).

Note: A system that is accepting incorrectly signed packets should provide a highly conspicuous indication that the connection is unsafe/insecure. Malformed signed packets indicate a bad configuration, transport failure, protocol failure, or hostile manipulation.

Note: MAVLink provides message signing, which allows systems to authenticate that messages are from a trusted source. MAVLink does not provide message encryption!

Useful Resources:

https://docs.google.com/document/d/1ETle6qQRcaNWAmpG2wz0oOpFKSF_bcTmYMQvtTGI8ns/edit

https://docs.google.com/document/d/1upZ_KnEgK3Hk1j0DfSHl9AdKFMoSqkAQVeK8LsngvEU/edit