Micro Air Vehicle Link (MAVLink) in a Nutshell: A Survey

Early Access article.

Anis Koubâa ; Azza Allouch ; Maram Alajlan ; Yasir Javed ; Abdelfettah Belghith ; Mohamed Khalgui

CISTER-TR-190701

Micro Air Vehicle Link (MAVLink) in a Nutshell: A Survey

Anis Koubâa ; Azza Allouch ; Maram Alajlan ; Yasir Javed ; Abdelfettah Belghith ; Mohamed Khalgui

CISTER Research Centre

Polytechnic Institute of Porto (ISEP P.Porto)

Rua Dr. António Bernardino de Almeida, 431

4200-072 Porto

Portugal

Tel.: +351.22.8340509, Fax: +351.22.8321159

E-mail:

https://www.cister-labs.pt

Abstract

The Micro Air Vehicle Link (MAVLink in short) is a communication protocol for unmanned systems (e.g., drones, robots). It specifies a comprehensive set of messages exchanged between unmanned systems and ground stations. This protocol is used in major autopilot systems, mainly ArduPilot and PX4, and provides powerful features not only for monitoring and controlling unmanned systems missions, but also for their integration into the Internet. However, there is no technical survey and/or tutorial in the literature that presents these features or explains how to make use of them. Most of the references are online tutorials and basic technical reports, and none of them presents a comprehensive and a systematic coverage of the protocol. In this paper, we address this gap, and we propose an overview of the MAVLink protocol, the difference between its versions, and its potential in enabling Internet connectivity to unmanned systems. We also discuss security aspects of MAVLink. To the best of our knowledge, this is the first technical survey and tutorial on the MAVLink protocol, which represents an important reference for unmanned systems users and developers.

IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

Micro Air Vehicle Link (MAVLink) in a Nutshell: A Survey

ANIS KOUBÂA[2,3,4], AZZA ALLOUCH[1,2,3,5,6],MARAM ALAJLAN[7],YASIR JAVED[2],ABDELFETTAH BELGHITH[7],AND MOHAMED KHALGUI[1,6]
[1]School of Electrical and Information Engineering, Jinan University (Zhuhai Campus), Zhuhai 519070, China
[2]Prince Sultan University, Robotics and Internet-of-Things Lab (RIOTU), Riyadh 12435 3276, Saudi Arabia
[3]Gaitech Robotics, Shanghai 201101, China
[4]CISTER, INESC-TEC, ISEP, Polytechnic Institute of Porto, 4200-465 Porto, Portugal
[5]Faculty of Mathematical, Physical and Natural Sciences of Tunis (FST), University of El Manar, Tunis, Tunisia
[6]LISI Laboratory, National Institute of Applied Sciences and Technology (INSAT), University of Carthage, Tunis 1080, Tunisia
[7] College of Computer and Information Sciences, King Saud University, Riyadh 11543, Saudi Arabia

Corresponding author: Anis Koubâa (akoubaa@psu.edu.sa) and Mohamed Khalgui (khalgui.mohamed@gmail.com)

This work was supported by the Robotics and Internet of Things Lab, Prince Sultan University, and Jinan University.

ABSTRACT

The Micro Air Vehicle Link (MAVLink in short) is a communication protocol for unmanned systems (e.g., drones, robots). It specifies a comprehensive set of messages exchanged between unmanned systems and ground stations. This protocol is used in major autopilot systems, mainly ArduPilot and PX4, and provides powerful features not only for monitoring and controlling unmanned systems missions, but also for their integration into the Internet. However, there is no technical survey and/or tutorial in the literature that presents these features or explains how to make use of them. Most of the references are online tutorials and basic technical reports, and none of them presents a comprehensive and a systematic coverage of the protocol. In this paper, we address this gap, and we propose an overview of the MAVLink protocol, the difference between its versions, and its potential in enabling Internet connectivity to unmanned systems. We also discuss security aspects of MAVLink. To the best of our knowledge, this is the first technical survey and tutorial on the MAVLink protocol, which represents an important reference for unmanned systems users and developers.

INDEX TERMS MAVLink, ArduPilot, PX4, Unmanned Aerial Vehicles (UAVs), Ground Control Station (GCSs).

I. INTRODUCTION

Unmanned systems are autonomous platforms that can be easily programmed to perform missions with or without the intervention of a pilot. These systems can be aerial, also known as drones or unmanned aerial vehicles (UAVs), ground (UGV) or underwater. They typically communicate through wireless with a ground control station (GCS) that monitors their status and control their actions. UAVs embed special hardware and software called autopilot that controls the motion of the drone and monitors its status and used to communicate with GCSs using telemetry or WiFi communication. There are many autopilot software platforms for UAVs available including ArduPilot from 3DR [1], Paparazzi UAV [2] developed at Ecole Nationale de l'Aviation Civile (ENAC), Hangar autopilot [3], PX4 Flight Stack [4], MultiWii from Nintendo [5] and several others. The most popular autopilot software is ArduPilot, which is an open-source project effectively maintained by a huge number of developers exceeding 400 contributors. It underpins different types of autonomous systems, including fixed-wing planes, (heli, tri, quad, hexa, and octo) copters, underwater vehicles and boats, and ground vehicles. In the heart of the ArduPilot system, Micro Aerial Vehicle Link (MAVLink) protocol is specified to ensure the communication between the unmanned systems and the ground stations. MAVLink is a well-established lightweight message serialization protocol specified for the unmanned vehicle systems, including drones. Lorenz Meier released MAVLink in 2009 under the LGPL license. MAVLink is designed as a Marshaling library, which means that it serializes messages of the states of the system and the commands that it has to execute into a specific binary format (i.e., a stream of bytes), that is platform-independent.

The binary serialization nature of the MAVLink protocol makes it lightweight as it has minimal overhead as compared to other serialization techniques, (e.g., XML or JSON). The communication using MAVLink is bidirectional between the ground station and the unmanned system. Besides, given the binary serialization feature of MAVLink, its messages are typically of small sizes and can reliably be transmitted over different wireless mediums, including WiFi or even serial telemetry devices with low data rates. It also ensures the reliability and message integrity by a double checksum verification, in its packet header. All these features make the MAVLink protocol as the most popular among its peers for the communication between unmanned systems and GCS.

Despite its popularity and the large community of users and developers, there is a significant lack of surveys about this protocol. New users/developers get usually confused due to the lack of structured references apart from some online documentation of some basic concepts. The only existing tutorial is entitled MAVLink Tutorial for Absolute Dummies (Part I), since 2013 [6], (for which there is no part II!), which is rather an basic introduction to the elementary concepts of the MAVLink protocol. Also, in the last two recent years, there has been much research works around the MAVLink protocol. However, no survey discusses these works and classifies them. Thus, there is a desperate need in the community for a scientific research paper that provides both a tutorial and a survey on the MAVLink protocol to be the first reference for users, researchers and developers on this protocol and unveils the powerful features of this protocol and its main usages, extensions, and applications.

This paper addresses this gap and provides a tutorial and a survey of the MAVLink protocol. The tutorial deals with presenting the main features of the MAVLink protocol and its two versions MAVLink 1.0 and MAVLink 2.0, and the most important messages specified in the protocol. In addition, we present the different tools and Application Program Interfaces (APIs) that developers need to parse and develop their control station programs that communicate with the unmanned systems. The survey part of the paper deals with presenting and discussing the main contributions proposed in the literature around MAVLink, which we classify in different categories including (*i.*) enhancement and extension (*ii.*) security, (*iii.*) applications, (*iv.*) integration with the Internet-of-Things (IoT), (*v.*) Multi-UAV coordination and swarm. To the best our knowledge, this is the first and unique technical survey and tutorial that deals with the MAVLink protocol, which represents an indispensable reference for unmanned systems' users and developers.

The remainder of the paper is organized as follows. Section II presents a general overview of the MAVLink protocol versions 1.0 and 2.0 and their header formats. Section III discusses the security threats and vulnerabilities of MAVLink and presents the solutions proposed in the literature to address these security problems. Section IV provides a comprehensive state of the art that has contributed to the development of MAVLink and its applications in different contexts. Section V presents an overview of the software related to the MAVLink protocol, including ground stations, and simulation models. Section VI concludes the paper and discusses some future challenges.

II. THE MAVLINK PROTOCOL

A. OVERVIEW

This section presents an overview of the MAVLink protocol, namely the transport and communication protocols supported in addition to the messages structures and serialization. In what follows, we use MAVLink messages and MAVLink packets interchangeably.

Note for readers: The following subsections present low-level technical details about the protocol specification that are needed for practitioners and MAVLink developers. These details are summarized in tables whenever relevant.

B. COMMUNICATION AND TRANSPORT PROTOCOLS

The MAVLink protocol defines the mechanism on the structure of messages and how to serialize them at the application layer. These messages are then forwarded to the lower layers (i.e., transport layer, physical layer) to be transmitted to the network. The advantage of the MAVLink protocol is that it supports different types of transport layers and mediums thanks to its lightweight structure. It can be transmitted through WiFi, Ethernet (i.e., TCP/IP Networks) or serial telemetry low bandwidth channels operating at sub-GHz frequencies, namely 433 MHz, 868 MHz or 915 MHz. The sub-GHz frequencies allow us to reach large communication ranges to control the unmanned system remotely. The maximum data rate can reach up 250 kbps, and the maximum range is typically expected to be 500 m, but highly dependent on the environment and level of noise and antenna setup. Table 1 presents the features of some commonly used telemetry devices.

The second alternative is to use a network interface, which is typically WiFi or Ethernet, and stream the MAVLink messages through IP Networks. In this case, the autopilot running the MAVLink protocol typically supports both UDP and also TCP connections at the transport layer between the ground station and the drone, depending on the reliability level required by the application. Of course, it is commonly known that UDP is a datagram protocol that requires no connection between the client and server and it has no mechanism to ensure that messages are reliably delivered, but provides a fast lighter weight alternative for real-time and loss-tolerant message streaming. On the other hand, TCP is a reliable connection-oriented protocol that provides better reliability of transfer thanks to its acknowledgment mechanism but could be subject to congestion and heavy management of the connection. The choice of the transport protocol is left to the user depending on the requirement he needs for the message exchange between the unmanned system and the ground station.

TABLE 1: An overview comparison between Ardupilot telemetry devices [7]

Telemetry device	Frequency	Range	Voltage	Sensitivity	RF transmit power
Bluetooth	Between 2402 and 2480 MHz, or 2400 and 2483.5 MHz	50 m	3.6 to 6 V	-80 dBm	+4 dBm
SiK Radio v2	900 MHz or 433 MHz	500 m	3.3 V	-121 dBm	20 dBm
RFD900	900 MHz or 868 MHz	>40 km	3.3 to 5 V	>121 dBm	+30 dBm
Robsense	433 MHz	3-5 km	5 V	-148 dBm	20 dBm

C. MESSAGE TYPES AND STRUCTURES

The unmanned system communicates with the ground station through the exchange of MAVLink messages, which are binary-serialized messages. *Binary serialization* means that the content of the message is transformed into a sequence of bytes to be transmitted through the network. The receiver of the serialized message performs its deserialization in the opposite direction to reconstitute the original message sent. This property of binary serialization has a significant benefit of reducing the size of the transmitted message to a maximum as compared to other types of serialization, such as XML or even the lighter weight JSON. Each MAVLink message contains a header appended to the message payload. The header carries out information about the message whereas the payload includes the data transported by the message.

In the following section, we present the protocol headers of MAVLink 1.0 and that of the newer MAVLink 2.0 [8] and the difference between each other. In the remaining of the paper, MAVLink refers to MAVLink 1.0 [9], unless otherwise specified.

1) MAVLink 1.0 Protocol Header

As shown in Fig. 1 there are eight important fields.

The first field is STX and refers to the symbol that represents the start of a MAVLink frame. In MAVLink 1.0, STX is equal to special symbol `0XFE`. The second byte (LEN) represents the message length in bytes and is encoded into 1 Byte. The third byte (SEQ) denotes the sequence number of the message. It is encoded into 1 Byte and takes values from 0 to 255. Once it reaches 255, the sequence number is reset again to 0 and incremented in each generated message. The sequence number of message enabled to detect message losses in the receiver. The fourth byte SYS represents the System ID. Every unmanned system should have its System ID, in particular, if they are managed by one ground station. The System ID 255 is typically allocated for ground stations. One limitation of MAVLink 1.0 is that it restricts the number of drones managed by one ground station to 254 because the System ID is encoded in 1 Byte. The fifth byte is the Component ID, and it identifies the component of the system that is sending the message. There are 27 hardware types (i.e., components) in MAVLink 1.0. If there is no subsystem or component, then it is not used. The sixth byte represents the Message ID (MSGID), which refers to the type of the message embedded in the payload. For example, the message ID equal to 0 refers to a message of type HEARTBEAT, which indicates that the system is alive and is sent every one second. One more example with a Message ID equal to 33, which refers to a message that carries out the GPS coordinate of the unmanned system. The message ID is the essential information that allows to parse the payload and extract the information from it, based on the type of message. Each message contains a specified number of fields and serialized in binary format in a particular order, according to the standard specification. The payload is located just after the message ID and can take a maximum of 255 bytes. Finally, the last two bytes are for the checksum. The CKA and CKB represent the Cyclic Redundancy Check (CRC) calculated with seed values A and B, respectively. The CRC ensures that the message has not been changed during its transmission and that both the sender and the receiver have the same message. It is calculated using the ITU X.25/SAE AS-4 hash of the bytes of the message, excluding the STX field (the hash is applied to 6+n+1 bytes, and the extra is the seed value). The seed is added at the end of the message when computing the CRC.

The minimum message length of MAVLink 1.0 is 8 bytes for acknowledgment packets without the payload. On the other hand, the maximum length of a MAVLink 1.0 message is 263 bytes for full payload.

Summary and explanation of each MAVLink 1.0 header fields are presented in Table 2.

2) MAVLink 2.0 Protocol Header

The MAVLink 2.0 protocol header was released in early 2017 and is the current recommended version. It is backward compatible with the MAVLink 1.0 version and includes several improvements over the MAVLink 1.0 version. We first start with presenting the MAVLink 2.0 protocol header; then we highlight the main differences between the two versions. Fig. 2 shows the header structure of MAVLink 2.0.

The MAVLink 2.0 header shares all the fields with MAVLink 1.0 header, and adds its new fields, in addition to the changing the size of some existing fields. The first byte is the start-of-text marker, and its specific value is `0xFD` for MAVLink 2.0 (as opposed to `0xFE` for MAVLink 1.0). Thus, the parser has to first recognize this character before being able to parse the remaining fields of the MAVLink 2.0 message. The payload length is the next field and is the same as in the legacy protocol. MAVLink introduces two new flags before the sequence number (SEQ) of the message. The first flags are *Incompatibility flags*, which are flags that affect the message structure. The flags indicate whether the packet contains some features that must be considered when parsing the packet. For example, an Incompatibility flag equal to $0x01$ means that the packet is signed and that a signature is appended at the end of the packet. The second flag is *compatibility flags*, which does not affect the structure of

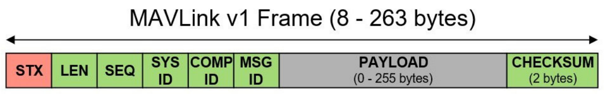


FIGURE 1: The MAVLink 1.0 Header Structure

TABLE 2: Explanation of MAVlink frame acronyms along with its contents.

Acronym	Content	Description
STX	0XFE	It describes start of frame and will always be 0xFE as in official documentation of MAVLink 1.0.
LEN	0 to 255	The value of LEN is described by length of payload.
SEQ	0 to 255	The sequence of packet is shown in this part of message. Such as 0 represent the first message. It is used for detecting lost MAVlink packets.
SYS	1 to 255	This field represents the ID of the unmanned system.
COMP	0 to 255	This field represents which component in the system is sending the message.
MSG	0 to 255	This field represents the message type.
Payload	0 to 255 bytes	This carries out the real data of the message, which depends on the message type.
CKA and CKB (CRC) or checksum	Two bytes contents	The CKA and CKB is referred as checksum. The signing of packet happens from Least Significant Bit (LSB) to Most Significant Bit(MSB).

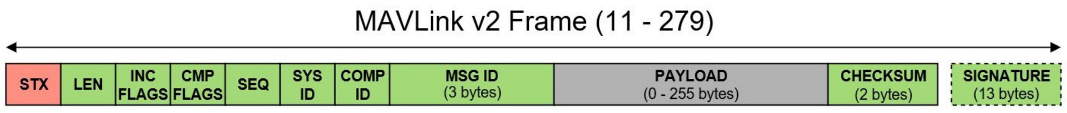


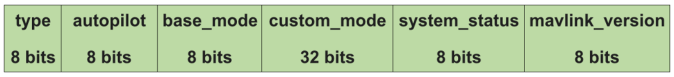
FIGURE 2: MAVLink 2.0 Header

the message. It indicates flags that can be ignored if not understood and it does not prevent the parser from processing the message even if the flag cannot be interpreted. For example, this may refer to flags that indicate the priority of the packet (e.g., High Priority) as it does not affect the packet structure. The sequence number (SEQ), the System ID (SYSID) and Component ID (COMPID) are the same as in the MAVLink 1.0 protocol header. However, the Message ID (MSGID) is encoded into 24 bits instead of 8 bits in the previous version, which allows a much higher number of message types in MAVLink 2.0, reaching up to 16777215 possible types. It is not clear what is the reason to design such a huge space of possible message types, as the number of possibilities is overly large. The Payload field can take up to 255 bytes of data, which depends on the message type. The checksum is similar to its peer in MAVLink 1.0. Finally, MAVLink 2.0 uses an *optional* Signature field of 13 bytes to ensure that the link is tamper-proof. This features significantly improve security aspects of the MAVLink 1.0 as it allows the authentication of the message and verifies that it originates from a trusted source. The signature of the message is appended if the incompatibility flags are set to $0x01$.

The 13 bytes of the message signature contain the following fields:

- *LinkID*: it is one byte that represents the ID of the link (channel) used to send the packet. A link or a channel can be WiFi or telemetry and can be combined. Every channel used to send data should have its own LinkID. It provides a means for multi-channel unmanned system control using MAVLink 2.0.
- *timestamp*: it is encoded with 6 bytes in 10-microsecond units since 1st January 2015 GMT. It increases for every message sent over the channel. It is applied to every stream where a stream is defined by the tuple `(SystemID, ComponentID, LinkID)`. The timestamp is used to avoid replay attacks.
- *signature*: it is encoded in 6 bytes for the message and is calculated based on the complete message, timestamp, and the secret key. The signature includes the first 6 bytes (48 bites) of the SHA-256 hash applied to the MAVLink 2.0 message (excluding the signature, and including the timestamp). The secret key is a shared symmetric key of 32 bytes stored on both ends, namely the autopilot, and the ground station or the MAVLink API.

The signature of the MAVLink 2.0 message has consequences on how to process incoming MAVLink messages. If the message is signed, then it is discarded if (*i.*) the timestamp of the received message is older than the previous packet received from the same stream identified by `(SystemID, ComponentID, LinkID)` tuple, (*ii.*) the computed signature at the reception is different from the signature appended to the message. This may infer a data alteration in the message, (*iii.*) the timestamp exceeded one minute as compared to the local system's timestamp. If the message is not signed, then the acceptance/rejection of the packet is implementation specific.



type	autopilot	base_mode	custom_mode	system_status	mavlink_version
8 bits	8 bits	8 bits	32 bits	8 bits	8 bits

FIGURE 3: MAVLink Heartbeat Message

D. MAVLINK MESSAGES TYPES

MAVLink defines several types of messages, which are identified by their Message ID. Messages with Message IDs lower than 255 are common for both MAVLink 1.0 and MAVLink 2.0, and those with Message IDs higher than 255 are specific to MAVLink 2.0. As mentioned in the previous section, the Message ID in MAVLink 1.0 is encoded in only 8 bits and was extended to 24 bits in MAVLink 2.0.

We categorize MAVLink messages into two classes:

- *State messages*: these messages are sent from the unmanned system to the ground station and contain information about the state of the system, such as its ID, location, velocity, and altitude.
- *Command messages*: they are by the ground station (or user program) to the unmanned system to execute some actions or missions by the autopilot. For example, the ground station can send a command to a drone to take off or to land or to go to a waypoint or even a to execute a mission with several waypoints.

Considering the large number of MAVLink messages, the comprehensive description of all these messages is out of the scope of this paper and can be found in details in [10]. In what follows, we present the most relevant state and command messages used in common implementations of autopilots. More messages are presented in Table 4.

1) State Messages

There are several types of state messages defined in MAVLink.

HEARTBEAT message: The HEARTBEAT message is the most important message in MAVLink, and its structure is depicted in Fig. 3. It indicates that the vehicle system is present and active. The unmanned system periodically sends the heartbeat message (in general every second) to the ground station to inform the GCS that it is active. The heartbeat is a required message. In addition to the header, the message payload contains essential information about the unmanned system. The first field is the `type`, which indicates the type of the Micro Aerial Vehicle. According to the latest specification in [10], there are 33 pre-defined types defined in the `MAV_TYPE`, including quadrotor (`MAV_TYPE_QUADROTOR = 2`), helicopter (`MAV_TYPE_HELICOPTER = 4`), fixed wing (`MAV_TYPE_FIXED_WING = 1`), and several others. The autopilot field indicates the type of autopilot. There are several types defined in the `MAV_AUTOPILOT` enumeration structure. For example, `MAV_AUTOPILOT_GENERIC = 0` indicates a generic autopilot, `MAV_AUTOPILOT_ARDUPILOTMEGA = 3` indicates ArduPilot autopilot, `MAV_AUTOPILOT_PX4 = 12` for PX4 autopilots.

The `base_mode` field indicates different operation modes. Understanding the `base_mode` is crucially important to correctly parse the heartbeat message and extract useful information out of it. It is encoded in 8 bits. There are 8 pre-defined flags, from 2^0 to 2^7. Here are the eight different modes:

- `Flag = 1` is reserved for future use
- `Flag = 2` means that the test mode is enabled. This mode is used for temporary tests and not used for regular flights.
- `Flag = 4` means that the autonomous mode (AUTO) is enabled. This means that the unmanned system is operating autonomously by navigating to the goal waypoints sent to it from the ground station. In AUTO mode, a mission is loaded to the autopilot. A mission consists of a set of *several* waypoints that the system has to navigate.
- `Flag = 8` means that the GUIDED mode is enabled. In GUIDED mode, a mission consists of a *single* waypoint sent to the system. The system then navigates to the specified location autonomously.
- `Flag = 16` means that the system stabilizes its attitude (orientation and altitude), and possibly its position, by automatic control. This requires external sensors like GPS in an indoor environment, altitude sensors (barometer, LIDAR) or motion capture for indoor positioning to be able to hover in a stable state. The system needs external control inputs to make it move around.
- `Flag = 32` means that the hardware in the loop simulator is activated, i.e., all motors and actuators of the motors are blocked while the internal autopilot is fully operational.
- `Flag = 64` means that manual mode is enabled, which requires that the pilot manually control the system using a remote control input. In manual control, there is no automatic control made by the autopilot.
- `Flag = 128` the system is in *armed* state, which means that motors are enabled/running and can start the fly.

The custom mode is also essential. It indicates autopilot specific flags that are interpreted in addition to the base mode. It is used in heartbeat message parsing to determine the flight modes of the autopilot system. There are pre-defined values for the custom mode including 0 for manual flight mode, 4 for guided mode, 10 for auto mode, 11 for RTL mode, 9 for LAND mode, 2 for ALT_HOLD, and 5 for LOITER. In the next section, we provide a comprehensive overview of the different flight modes in ArduPilot systems supported by MAVLink.

The `system_status` field represents a flag that indicates the system state. There are night states defined as of the latest specification [10]:

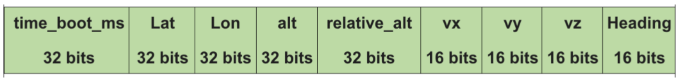
- `system_status = 0` refers to a system that is not initialized system or an unknown state.

- `system_status = 1` indicates that the system is booting.
- `system_status = 2` means that the system is performing a a calibration. In fact, the sensor calibration is a very important phase to make sure that flight sensors such as Inertial Measurement Units (IMU), and Compasses are consistent and run as expected.
- `system_status = 3` it means that the system is in *standby* mode and can be started at any time.
- `system_status = 4` indicates that the motors are engaged and that the system is *active* and might be airborne.
- `system_status = 5` indicate potential errors and that the system is in *critical* state, although it can still navigate. This can happen for example during temporary interference or battery level starting to be low, etc.
- `system_status = 6` this means an *emergency* situation where the unmanned system lost control over some parts and is in distress situation. The system may have already been crashed.
- `system_status = 7` indicates that the system has started its power-off process and is now *shutting down*.
- `system_status = 8` indicates that the system is *terminating* itself and ending its flight.

Finally, `mavlink_version` field indicates the MAVLink version. It is not editable by the user and is set by the protocol.

System Status message: The system status message has a Message ID equal to 1 and is composed of data about the onboard control sensors embedded into the unmanned system and specifies which of these sensors are enabled/disabled and which sensors are operating or having errors. It also carries out data about the battery status and the remaining voltage, which is useful to track the battery level of the unmanned system. Besides, it provides information about the communication errors and the ratio of dropped packets in the communication link. The information about battery and communication link are crucial to take appropriate failsafe action when the battery level goes down, or the communication quality deteriorates. In this case, the unmanned system can be pre-programmed to execute a failsafe operation in case of low battery level or bad communication quality such as landing and going back home for an unmanned aerial system.

Global Position message: The global positioning message has an id equal to 33 and represents the filtered GPS coordinate provided by the Global Positioning sensor. It is illustrated in Fig. 4. This message carries out important information of the unmanned system related to its global positions, namely, the GPS latitude (`lat`), longitude (`lon`) and also *absolute* altitude (`alt`). These three values are encoded into four bytes (32 bits). The values of (`lat`) and (`lon`) must be divided by 10^7, to get the real floating GPS value, it is needed to divide them by 10^7. The altitude is expressed in millimeters. The message also contains a *relative* altitude field (`relative_alt`), which represents the altitude relative to the takeoff ground point of the unmanned aerial system. It is different from the absolute altitude. For example, the absolute altitude of Riyadh city is 612 meters, which corresponds to a relative altitude of 0 meters, as the drone is on the ground. If the drone takes off at a relative altitude from the ground of 10 meters, then its absolute altitude becomes 622 meters. Besides, this message also carries out information about the linear speed of the unmanned system along the 3 axis ((x,y,z) in addition to orientation referred to as heading. This information is collected from the GPS sensor, and it can also be read from other sensors such as the Inertial Measurement Unit (IMU) or the compass, which are available in other MAVLink messages.



time_boot_ms	Lat	Lon	alt	relative_alt	vx	vy	vz	Heading
32 bits	32 bits	32 bits	32 bits	32 bits	16 bits	16 bits	16 bits	16 bits

FIGURE 4: System status message

2) Command Messages

There are several command messages in MAVLink that give the ability to request the unmanned system to perform certain actions. In what follows, we provide an overview of the most important commands. Table 3 presents a summary of a selected set of MAVLink commands.

COMMAND_LONG: The COMMAND_LONG is a multi-purpose command that allows sending different types of commands depending on the command type of the message and its parameters. The COMMAND_LONG message has a Message ID equal to 76 and is defined with 11 fields, as illustrated in Fig. 5. The `target_system` field and the `target_component` field specify the system that will execute the command, and its underlying component.

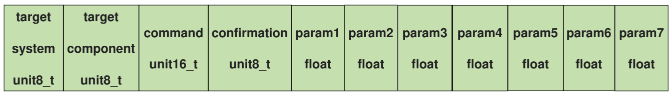
The command field refers to the type of command to be executed. It is defined in the `MAV_CMD` command enumerations. Also, for each command, a set of parameters relevant to the command can be set. In what follows, we provide a summary of some commands and their respective parameters.

For example, the command ID with 21 refers to the LAND command, and it has no parameter. Some other commands have parameters like the takeoff command (ID=22), which has to specify the takeoff altitude in `param 7`, and the command arm/disarm which specifies a Boolean value in `param1` to indicate whether to arm or disarm the motors. There are around 60 commands types defined in the MAV_CMD command enumerations. The confirmation field indicates if the message was sent for the first time with value 0, and other values represent a confirmation of the message. The 7 parameters depend on the type of command. For example, for the LAND command, all seven parameters are useless. In the takeoff command, the seventh parameter represents the altitude requested for the takeoff.

The Mission Item Command: The mission item command has a Message ID equal to 39 and allows to send a waypoint to an unmanned system so that it navigates

TABLE 3: Selected List of Important MAVLink Commands

Command	Command ID	Parameters	Description
TAKEOFF	22	param7: double	this command makes the aerial unmanned system takeoff at an altitude specified in param1
LAND	21	No parameters	this command makes the aerial unmanned system land to ground
GET_HOME	410	No parameters	this command allows to get the Home position, which is the first waypoint in the mission list
SET_HOME	179	param5: double param6: double param7: double	this command allows to get the Home position, which is the first waypoint in the mission list
ARM DISARM	400	param1: boolean	this command allows to arm the motors if param1 is set to true, and disarm them if param1 is set to false



target system	target component	command	confirmation	param1	param2	param3	param4	param5	param6	param7
unit8_t	unit8_t	unit16_t	unit8_t	float	float	float	float	float	float	float

FIGURE 5: Command long

autonomously to that specific waypoint in GUIDED mode. Every mission item command message has a sequence number that specifies its number in the mission, starting from 0, which specifies the home location. It also has three fields `(x,y,z)`, which specify the coordinates of the waypoint. However, the coordinate must be specified with respect to a reference frame. Thus, the message has a field called `frame`, which specifies the reference coordinate frame of the waypoint. This parameter is important because it is essential to interpret the meaning of the coordinates `(x,y,z)`. For example, if we set the waypoint coordinate as (24.68773, 46.72185, 10) with reference to the global frame MAV_FRAME_GLOBAL, this would mean to go to the waypoint at GPS location (24.68773, 46.72185) and an *absolute* altitude of 10 meters with respect to sea level. For example, in Riyadh city (Saudi Arabia), the absolute altitude is 620 m, so this would be that the drone may go down and crash to the ground because the target altitude is lower than that of the ground. However, usually, we want to specify the location with respect to the ground (e.g., 10 meters above ground) and for this, we need to specify the reference frame to be a global frame but with *relative altitude* namely MAV_FRAME_GLOBAL_RELATIVE_ALT. The command also specifies the target system and target component as in other command messages.

TABLE 4: Selected List of Important MAVLink Messages

Message Type	**Message Representation**	**Description**
0	HEARTBEAT	It is the most important message in MAVLink that tells if the unmanned system is alive or not.
1	SYS_STATUS	It defines the unmanned system state including onboard sensors, communication quality and battery status.
2	SYSTEM_TIME	It defines the system time of the master clock that is usually the onboard clock.
5 and 6	CHANGE OPERATOR (CONTROL, ACK)	They represent the request to take control over the unmanned system and its corresponding acknowledgment
20, 21, 22, 23	PARAM REQUEST (READ, LIST),PARAM (VALUE ,SET)	These four important messages are related to on-board parameters whose value can be obtained by GCS or can be set by GCS. For example, it is possible to request to read the SystemID parameter or to change it.
24, 25, 33, 48, 49, 123, 124, 127	GPS RAW INT, GPS STATUS, GLOBAL POSITION INT, SET_GPS GLOBAL ORIGIN , GPS GLOBAL ORIGIN, GPS INJECT DATA, GPS2 RAW, GPS RTK,	These messages are related to the GPS sensor information, such the raw GPS value, the Global Position value, etc.
26,27	SCALED IMU and RAW IMU	These messages contain the scaled and raw IMU sensor data according to 9 degrees-of-freedom including acceleration, gyro (angular speed) and magnetic field all in three axes
37 up to 47 and 51	MISSION related messages	There are 10 messages defined for missions request a mission, or set a waypoint in a mission, clearing a mission (i.e., delete all its waypoints), or getting acknowledgement for a mission, etc. A mission is defined as a set of waypoints sent to the unmanned system to navigate to them in the autonomous mode.
34, 35, 50, 65, 70, 92	RC_CHANNELS SCALED, RAW, PARAM MAP RC, OVERRIDE, HIL RC INPUTS RAW	All these messages are for RC control to get the channels, write data or controls.
75, 76	COMMAND (INT, LONG, ACK)	It sends the command to the unmanned system for performing actions such as (a) navigation commands (b) do commands (start, jump etc.) or (3) condition commands, all listed in MAV CMD.

E. FLIGHT MODES

Understanding flight modes is crucially important to be able to pilot a unmanned system running Ardupilot and the MAVLink protocol. There are several flight modes that were defined by Ardupilot. In this section, we present the most important and more common flight modes.

- **The STABILIZE mode** In this mode, the unmanned system is fully controlled by the user, position, altitude, and heading. The unmanned system will respond to every input from the RC controller controlled by the user, and it is up to the user to compensate any drift

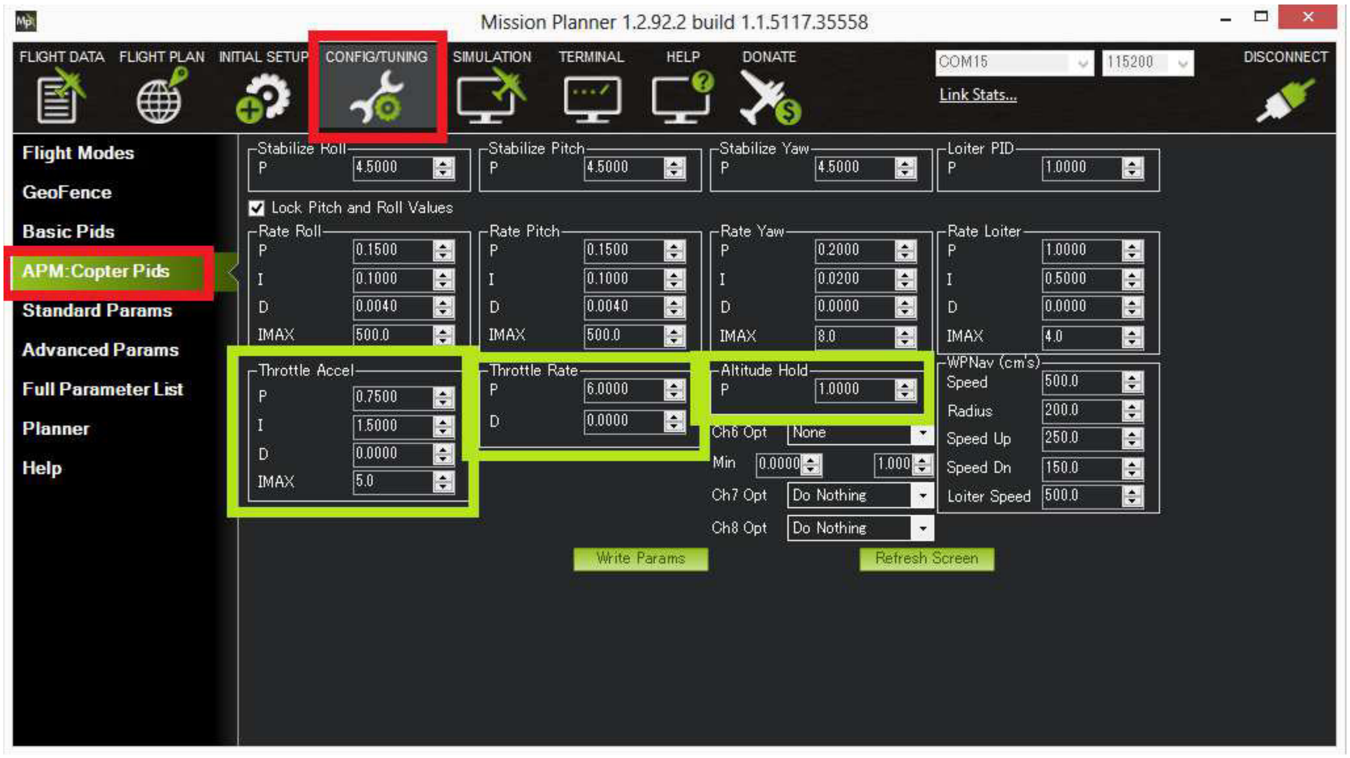
made by the unmanned system.
It is recommended that users immediately switch to the STABILIZE mode to manually control the unmanned system, when the autopilot fails to control the vehicle in any other autonomous mode.
It has to be noted that it is possible to download dataflash log files from your unmanned system, to analyze the flight performance by opening it in mission planner in the graph tab.

- **ALTITUDE HOLD**: A more comfortable mode to control the unmanned system is the ALTITUDE HOLD or ALT HOLD mode where the user does not have to worry about maintaining a fixed altitude for the unmanned system, as the autopilot will take care of controlling automatically using a PID controller the altitude. The user will have to take care of controlling the direction and position of the unmanned system manually.
The ALT_HOLD mode automatically controls the altitude of the unmanned system by the autopilot. However, it does not control the heading, and position, which are left to the user. This mode is more recommended for newbies than the STABILIZE mode and does not require a GPS because it estimates the altitude with the barometer.
It is possible to tune the setting of the PID controller of the ALT_HOLD mode in APM mission Planner as illustrated in Fig. 6. The altitude is maintained with proportional controller that estimate the error between the desired altitude and the actual altitude and tune the vertical acceleration proportionally to that error. The proportional gain can be set through a ground station as illustrated in Fig. 6. The Proportional gain must be carefully set because a very high gain will make the control more aggressive and less stable, whereas a very low gain will make the control very slow and non-responsive.
- **LOITER**: An even more accessible mode to control the unmanned system is the LOITER mode, which maintains the current location, orientation, and altitude of the unmanned system once the user does not provide input to the RC controller. This mode is similar to the STABILIZE mode, but the unmanned system will control its position, heading, and altitude once the user gets his fingers out of the RC sticks. The LOITER mode requires a GPS 3D fix to work with, or optical flow.
It has to be noted that it is not possible to arm the vehicle in LOITER mode only if (1) GPS has 3D Lock, (2) HDOP is smaller than 2.0. As such, this mode requires a GPS 3D fix to work with.
To achieve excellent LOITER performance, it is essential to have (1) GPS Lock, (2) low magnetic interference of the compass, and (3) low vibration.
The PID controller gains can be tuned from the Mission Planner ground station or similar ground stations.
The LOITER SPEED represents the max horizontal speed in cm/s and is typically equal to 500 cm/s (which is equivalent to 5m/s). The default configuration is that the maximum acceleration is equal to the half of the LOITER SPEED (i.e., $2.5m/s^2$).
The parameters of the LOITER mode can be configured through a ground station as illustrated in Fig.6 by setting PID control gains of the altitude, position and orientation (Yaw, Pitch and Roll).
- **LAND**: The LAND mode will force the unmanned system to land to the ground.
- **RTL**: The RTL mode, also called Return to Launch, will force the unmanned system to return to start position where it performed the TAKEOFF.
LAND and RTL mode are used in case of violation of navigation safety and geofence, for example, it is possible to program on the autopilot that if the battery goes under a certain level, then the unmanned system needs to LAND immediately or return to start position automatically. This is called GEOFENCE.
- **GUIDE**: The GUIDED mode is essential and operates only with GPS mode. When the GPS of the unmanned system performs 3D fix and is activated, then the unmanned system can be sent to navigate autonomously to a particular GPS coordinate through the ground station. It is called GUIDED mode because the unmanned system is guided by the user to navigate autonomously to a specific waypoint chosen by the user.
The GUIDED mode is used in conjunction with GPS and allows the user to send the unmanned system to specified waypoint defined by their GPS coordinates. It is not possible to arm the unmanned system in GUIDED mode only when the GPS has the 3D fix.
In GUIDED mode usually, a ground station is used to send a navigation waypoint to the unmanned system to navigate to it. It is therefore important to have a telemetry device connected to the unmanned system and the ground station to allow long-range communication between them.
Typically, a user needs to click on a point on the map, in GUIDED mode and the unmanned system will plan its path and move towards the goal.
- **AUTO**: The AUTO mode refers to the autonomous mode, where the unmanned system will follow a predefined mission. A Mission is a set of waypoints stored in the unmanned system autopilot. When the AUTO mode is selected, the unmanned system will autonomously go to each waypoint in the same order as they are stored.

III. SECURITY ISSUES OF MAVLINK

A. MAVLINK SECURITY REQUIREMENTS

Many research works dealt with the security of unmanned systems but much less addressed MAVLink security issues. Among them, several works tackled the security requirements that must be taken into account in the MAVLink protocol. In what follows, we summarize the most important requirements in terms of confidentiality, integrity, availability, authentication, non-repudiation, authorization, and privacy in



Stabilize Roll: P 4.5000
Stabilize Pitch: P 4.5000
Stabilize Yaw: P 4.5000
Loiter PID: P 1.0000

✓ Lock Pitch and Roll Values

Rate Roll: P 0.1500, I 0.1000, D 0.0040, IMAX 500.0
Rate Pitch: P 0.1500, I 0.1000, D 0.0040, IMAX 500.0
Rate Yaw: P 0.2000, I 0.0200, D 0.0000, IMAX 8.0
Rate Loiter: P 1.0000, I 0.5000, D 0.0000, IMAX 4.0

Throttle Accel: P 0.7500, I 1.5000, D 0.0000, IMAX 5.0
Throttle Rate: P 6.0000, D 0.0000
Altitude Hold: P 1.0000

Ch6 Opt: None
Min 0.0000 1.000
Ch7 Opt: Do Nothing
Ch8 Opt: Do Nothing

WPNav (cm's): Speed 500.0, Radius 200.0, Speed Up 250.0, Speed Dn 150.0, Loiter Speed 500.0

FIGURE 6: Settings of the PID controller

order to secure MAVLink communications.

- **Confidentiality:** It is an essential requirement to guarantee private data exchange between entities. It protects against attacks pertaining to non authorized disclosure of secret information exchanged in the network between the unmanned system and the ground station, since an adversary might be able to intercept commands sent from GCS to the unmanned system, or steal other system state data transmitted from the GCS to the unmanned system.
- **Integrity:** It is mandatory to secure communications between GCSs and UAVs by ensuring the integrity of exchanged data. Integrity is required to ensure that telemetry information sent from the unmanned systems and control signals sent from the GCS have not been intentionally or unintentionally interrupted, altered or modified. It protects against threats that pertain to unauthorized information modification.
- **Availability:** The communication between UAVs and GCSs should be available, as well as the information itself is always available when needed or requested, even if a fault has occurred on the UAV system or an attacker tries to jam the UAV/GCS channel.
  It is fundamental that all the elements of the UAV system are operating and performing their requested functions and expected services when needed. Considering system maintenance, it is mandatory to ensure the continuous operation of the service without interruption, and to guarantee that the system performance is maintained so that the system keeps its availability uninterrupted during operation. Besides, for the accessibility to the UAV, the service must be available when the user needs it.
- **Authentication:** In a network of unmanned systems, multiple entities are participating and exchanging information in the network. Authenticating these entities and information origins is mandatory. Authentication allows each node to verify the origin of the data transmitted, i.e., make sure that the message is effectively received from an authentic source. The authentication of the unmanned system by the GCS is highly critical to make sure the GCS is controlling an authorized drone, not a fake one. Also, it is essential that GCS can also be authenticated so that an unmanned system does not send its state or accept commands from a hacked/fake ground station. Therefore, it is mandatory to authenticate both ends to ensure that data sources are trustworthy.
- **Non-repudiation:** Any entity in UAV network does not deny that it has sent or received data or control commands.
- **Authorization:** It refers to the ability of a system to permit access to information, and which actions these entities are allowed to perform. In the case of MAVLink only authorized GCSs and UAVs, can enter the network, are permitted to exchange telemetry data and send control command.
- **Privacy:** Exchanged information between GCSs and UAVs includes sensitive information about location,

battery status, speed, weather, wind speed, mission status, etc. This private information must not be leaked to unauthorized third parties. Thus, it is mandatory to preserve the privacy of communications, hide UAVs and GCSs' identities and protect sensitive information issued by UAVs and GCSs from intruders.

B. MAVLINK SECURITY THREATS

Communication between UAVs and GCS is established by a communication protocol via a wireless channel, which makes them vulnerable to various attacks since the communication protocol MAVLink does not support security procedures. Both confidentiality and authentication mechanisms are not natively supported. The GCS exchange data with UAVs through an unauthenticated channel and without encryption. These connections can be easily hacked, someone with an appropriate transmitter can communicate with the drone, inject commands into an existing session and easily launch attacks on UAVs.

The open nature of communications makes MAVLink vulnerable against various malicious attacks. These attacks can be classified as Interception (Attacks that compromise data confidentiality), Modification (Attacks that compromise data integrity), Interruption (Attacks that compromise data availability) and Fabrication (Attacks on authenticity). In the following, we detail further these threats:

1) Confidentiality and privacy attacks

In this category of attacks, an intruder gets an unauthorized access to confidential and sensitive information by intercepting data, commands or messages exchanged between UAV and GCS. The confidentiality and privacy of the information are affected in this category of attack. Such attacks concern eavesdropping, identity spoofing traffic analysis and unauthorized access and are a result of deficiencies in the security of MAVLink.

- **Eavesdropping (Communication capture):** Due to the lack of encrypted connections, an attacker listens to the communication happening between UAVs and GCS, eavesdrops on the information exchanged between UAVs and GCS directly from the open environment. This kind of attacks is exploited by the adversary to obtain information about the UAV, and consequently perform more elaborated attacks (active attacks). The attacker captures control data and commands sent from the GCS to the UAV, to be used in a replay or a fabrication attack. Telemetry data broadcasted from UAVs to the GCS is intercepted by an adversary to gain knowledge about the UAVs location and flying speed. Eavesdropping is a passive attack that breaches the confidentiality and privacy of the control signal and telemetry data. The lack of data encryption and authentication in communication stimulates such attacks.
- **Identity spoofing:** The MAVLink Communication protocol is unencrypted and uses the System IDs to identify the drone which sends or is expected to receive the messages. System ID is sent in clear within the unencrypted MAVLink header. Thus, an attacker can compromise the communication link to get the identity of the sending system.
- **Traffic analysis:** Traffic analysis is a passive attack. An intruder may collect exchanged data to infer specific data to reveal specific patterns about the communication between UAV and GCS. It can be any useful information such as frequency of MAVLink communication, size of MAVLink packets, etc. Traffic analysis is a method to gather useful and sensitive information that potentially can be used in other attacks.
- **Unauthorized access:** It occurs when the attacker gets access to the UAV and/or GCS, their services and resources using duplicated SYSID or COMPID. This attack usually results in unauthorized disclosure of GCS and telemetry information from UAV.

2) Integrity attacks

The integrity of MAVLink can be compromised by modifying the data being sent. Violation of the MAVLink integrity allows the following attacks:

- **Man-in-the-middle:** In the MAVLink communication protocol, messages are sent in plain text, which represents vulnerability and a threat to network security. The M-I-T-M attack can be successfully established in the channel. The attacker is located between UAVs and GCS and listen to the exchanged communications. The attacker can infer the content of the intercepted MAVLink payload and reconstruct commands. He can replay previously recorded packets, modify control and telemetry data and send these wrong data back to the GCS or the UAV. Thus, the integrity of the control data and telemetry data is hampered.
- **Hijacking (Unauthorized Command Injection):** There is a possibility, whenever a M-I-T-M attack against a UAV is successful, the attacker sends unauthorized commands to the UAV to takes its control from its GCS while allowing the GCS to believe that it is still controlling the UAV. Once the drone is under his control, the attacker can catch and withhold the UAV. There are two ways to hijack a drone using MAVLink vulnerabilities: (*i.*) `(Skyjacking)` : Exploiting MAVLink's lack of authentication, a drone called Skyjack hacks other drones by using airplay-ng software to force disconnecting the authentic user from the drone by injecting de-authentication messages. Since the drone does not authenticate users, an hacker may easily connect to it and take control over the device as soon as the WIFI connection is established. (*ii.*) `(Radio Jacking)` : It is another way to hack a UAV using the MAVLink's vulnerability. To control the drone using telemetry via MAVLink, it is mandatory to set up the NetID to connect to the drone. If an intruder recognizes the NetID field, he can easily hijack the UAV through the use of an antenna

with the sniffed NetID to transmit malicious MAVLink packets and false information.

- **Replay attack:** Due to the ease of message capture and open nature of communications, the MAVLink protocol is vulnerable to replay attacks. A malicious user records the control data sent to the drone and replays them later to misuse the drone and produces an unauthorized effect. This attack may cause loss of control over the drone and possible crash.
- **Message modification:** Modification of messages means altering the contents of the data packet. The attacker captures the control data sent by the GCS, modifies them and sends wrong data back to the UAV. As a result, the GCS's control data are misinterpreted by the drone resulting in the drone being uncontrollable. The integrity of the control data and telemetry data is compromised.
- **False location update:** An attacker can send spoofed messages to the GCS using the data link that seems to be from a UAV containing false UAV location data using Scapy (a packet manipulation tool) to spoof heartbeat messages. This attack makes GCS believing that the UAV is in another location, or is following a wrong trajectory.

3) Availability attacks

Attacks that compromise the availability of MAVLink can be achieved through interruption of the link used to exchange data between the drone and the ground station. There are several means on how to perform this attack, in particular through jamming, deletion attack, falsifying signals and Denial of Service (DoS/DDoS) attacks.

- **Jamming:** This attack affects system availability. An attacker who is trying to take full control of the drone interrupts the UAV reception of the GCS control signals by breaking the communication link. The jamming attack results in the loss of communication between UAVs and the GCSs through the loss of control signals, which leads the drone to enter into a *lost-link state* preventing the controller from operating correctly, and thereby causing unavailability of services.
- **Denial of service (DoS):** An attacker may use the MAVLink vulnerability to flood the UAVs GCS communication channels with data; the network becomes interrupted, which leads to resource (UAV and GCS) unavailability. This form of attack is called a DoS attack. In such an attack, the control messages as well as the mission data is not properly received by the drone. As a result, the drone cannot remain in a stable state, and the mission is not executed appropriately. If DoS attack is successful, it can result in a (M-I-T-M) attack. By conducting a MITM attack, the attacker sends unauthorized commands in an infinite loop to the UAV. This could effectively deny the communication between the GCS and the UAV, preventing the legitimate commands sent by the GCS from being treated by the UAV, as the drone would always be occupied by commands issued from the attacker. A successful DoS attack against the drone makes it no longer responsive to the GCS, or vice versa, because of the violation of system availability.
  Furthermore, in DDoS attack, an adversary sends a huge number of packets to the UAV or the GCS which causes a network congestion and prevents the UAV and the GCS from communicating with each other (failing to respond to commands).
- **Flooding:** This attack works on the principle of flooding the network with a huge number of various packets to make it down. Generally, packets of types SYN, UDP, ICMP and Ping are used in this kind of attacks. In [11], a simulated attack, ICMP flooding attack, was performed to exploit the vulnerability of the MAVLink waypoint protocol. The intruder sends many ICMP request packets to both the GCS and the UAV during their mission. The GCS and the UAV are too overloaded, and thus, cannot respond to commands. As such, the UAV sensor values, the GCS mission commands were not appropriately transmitted. Furthermore, the heartbeat message necessary for maintaining the connection between the UAV and the GCS is received after the target time because of the ICMP flooding attack. In this case, the UAV crashed without operating the fail-safe mode due to an error in the fail-safe mode.

4) Authenticity attacks

Authenticity attacks try to make the GCS/UAV believe that falsified data is authentic. Authenticity of a MAVLink message can be hampered by fabricating malicious data to replace the legitimate data. Fabrication attacks include data fabrication and GCS spoofing.

- **Data fabrication:** To perform a fabrication attack, the adversary needs to have extended knowledge about how the GCS and the considered drone communicate, i.e., knows the protocol used by the drone and the GCS, which can be achieved by having performed eavesdropping and traffic analysis attacks. This attack violates the integrity of the control data and/or telemetry data, and can lead to a hostile takeover of the drone.
- **GCS spoofing:** The UAV connects with the GCS via wireless links for data and control signal exchanges. However, since the wireless environment is open, an attacker could successfully spoof MAVLink commands. More specifically, a malicious attacker can send a false wireless control command to take over the UAV illegitimately.

Fig. 7 summarizes the potential security threats against the MAVLink protocol, the identified attacks, and the corresponding violated security properties.

IV. MAVLINK SECURITY SOLUTIONS

Despite the widespread use of the MAVLink protocol, it has security gaps and is prone to several attacks that result in critical threats and safety concerns. The protocol does

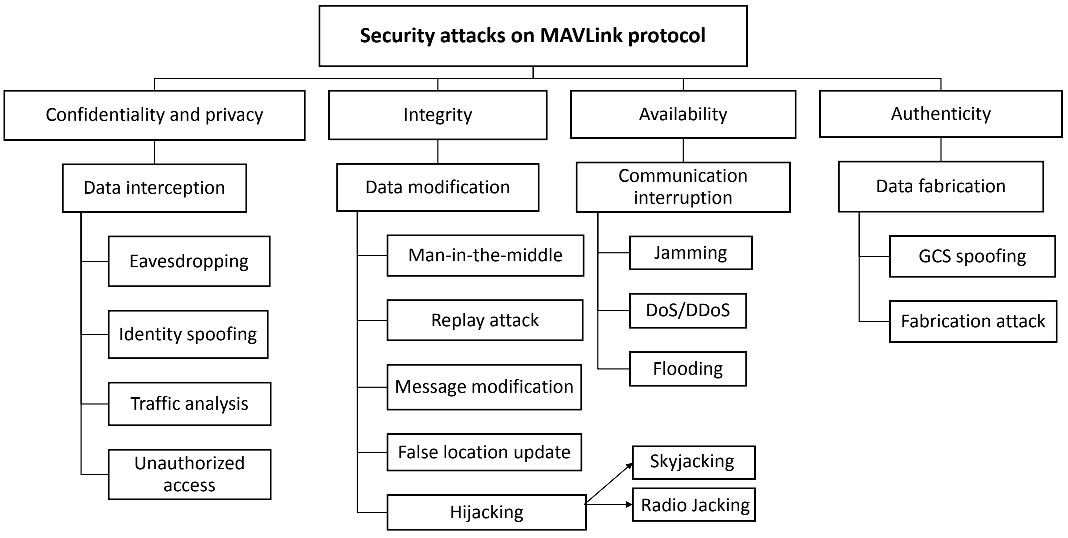


FIGURE 7: Security threats and attacks against MAVLink Protocol

not implement encryption and authentication mechanisms. Therefore, MAVLink is very prone to security threats and attacks.

Several techniques and solutions have been already proposed to address the security issues described previously. In this section, we provide an overview and a classification of the proposed solutions for each security service as illustrated in Table. 5. Existing security solutions proposed for securing the MAVLink communication protocol can be classified into *hardware* and *software* approaches.

A. HARDWARE-BASED SOLUTION

Several embedded and hardware security solutions have been introduced to secure the MAVLink protocol. In [12], a lightweight hardware-based solution is proposed to secure the communication between the GCS and the drone. An FPGA module connected to the drone embeds the symmetric key cryptography function: AES-CBC-MAC was used to encrypt and authenticate both commands and payload data communicated between the drone and the GCS. However, the hardware solution negatively affects system performance and power consumption due to the extra hardware weight. In [13], the authors proposed the idea of an additional encrypted communication channel to improve UAV data security through Raspberry *Pi*. This channel was designed to resume the control of the UAV if any attack was detected on the drone. However, this hardware solution induces delays between the GCS and the Raspberry *Pi* and increases the CPU usage on the Raspberry *Pi*.

B. SOFTWARE BASED SOLUTION

We may here further distinguish among several approaches: the classical security approaches, the Intrusion Detection System (IDS) based approaches, and the new emerging Blockchain technology based approaches.

1) Classical security approaches

This category of solutions groups cryptographic-based approaches used in the context of MAVLink to address the main security services. We review the proposed cryptographic solutions and discuss their main advantages and shortages relatively to each security service.

a: Confidentiality solutions

To ensure confidentiality of both control signals and telemetry data, MAVLink data has to be encrypted before being sent. However, it is important to use both a strong key and a powerful cryptographic algorithm to mitigate the vulnerabilities of the MAVLink protocol in terms of confidentiality. Cryptographic algorithms are classified into two main classes: symmetric and asymmetric cryptographic solutions.

(*i.*) **Symmetric key solutions:** Each entity in the system shares the same cryptographic keys with all other entities. The main benefits of symmetric key schemes are their easy implementation, fast design, and low computation requirements. In fact, they use the same key to encrypt and decrypt data, which makes it appropriate for limited-resource drones.

In [14], the authors introduced SMACCMPilot, a secure UAV project based on the MAVLink protocol. SMACCMPilot refers to GIDL as the application level protocol. GIDL uses AES to encrypt the MAVLink payload, header, and CRC. The community of the MAVLink protocol developers are currently discussing a secure version of MAVLink (sMAVLink) [15]. sMAVLink has the same encryption algorithm as GIDL but encrypts only the payload, which makes the MAVLink packet structure untouched. To the best of our

TABLE 5: Survey on the state-of-the-art on existing security solutions

Category	Key method	Proposed solution	Research focus	REF
Hardware based solution	AES-CBC-MAC embedded in FPGA module	Hardware	Confidentiality and authentication	[12]
	Additional encrypted channel through Raspberry *Pi*		Resume the control of UAV if any attack was detected	[13]
Classical security approaches	AES	Symmetric key	Confidentiality	[14], [15]
	Galois Embedded Crypto Library			[16]
	AES-ECB, AES-CBC			[17]
	Rabbit stream cipher, XXTEA stream cipher, and Salsa20 stream cipher			[18]
	RC5			[19]
	Caesar cipher			[20], [21]
	RSA and ECC	Asymmetric key		[17]
	probabilistic selective data encryption	Identity-based encryption (IBE)		[22]
	(IBE-Lite)			[23]
	Private key	Digital signature	Integrity	[24]
	Authenticated encryption algorithm	Symmetric key		[25]
	Message Authentication Code(Poly1305) and authenticated encryption algorithm Galois/Counter Mode (GCM)			[18]
	Message Authentication Code (MAC)			[26]
	Strong authentication based solution	Symmetric key	Availability	[12], [27]
	AES-GCM	Symmetric key	Authenticity	[18], [28]
	Caesar Cipher			[20], [21]
	Message Authentication Code (MAC)			[12]
	Elliptic Curve Cryptography (ECC)	Asymmetric key		[17]
	The signature represents the first 48 bits of an SHA-256 hash of the secret key, header, payload, CRC, link ID, and timestamp	Digital signature		[8]
Intrusion Detection System (IDS)	Behavior rule-based solution	Rule-based specification detection	Detect and guard a UAV system against cyber-attacks	[29]
	Behavior rule-based solution		Evaluate the behavior of attacks that target UAV	[30]
	UAV behavior based fight commands	Signature-based detection	Authentication	[31]
	Statistical method (recursive least squares technique)	Anomaly-based detection	UAV real-time monitoring system	[32]
	Belief-based threat estimation		Protect UAVs from attacks targeting data integrity	[33]
	Neural network and fuzzy learning algorithm		Protect UAVs against a distributed denial-of-service (DDoS) attacks	[34]
	Support Vector Machine (SVM) algorithm		Detect cyber-attacks that target autonomous avionic systems	[35]
	Bayesian game model		Protect UAV-aided network against lethal attackers	[36]
	Rule-based detection and SVM-based anomaly detection	Hybrid-based detection	Identify cyber-attacks	[37]
	Signature-based anomaly detectors and residual-based anomaly detectors. Bayesian network to estimate possible attacks		Detects GPS spoofing attacks	[38]
New emerging security solutions	Blockchain	Blockchain	Data integrity, trusted source , accountability, and resilient backend	[39]
			Secure the communication among UAVs	[40]
			Securely relay drone information	[41]
			Security and privacy	[42]

knowledge, sMAVLink has not yet been implemented.

In [16], the authors proposed an encrypted radio control link based on Galois Embedded Crypto library with the openLRSng open-source radio project for securing communication links among open source UAV systems. The proposed solution uses the symmetric key produced by a trusted third-party entity and manually hard-coded in the autopilot. This approach is not efficient and may lead to security vulnerabilities and restricts its feasibility. In [17], the authors chose AES-ECB and AES-CBC to protect command messages.

In [26], the authors proposed a transfer protocol, which provides confidentiality service and data transfer service between the drone and GCS. User data are encrypted with a key, derived during the initialization phase to guarantee data confidentiality,

The work in [18] presented four suitable cryptographic implementations that may be able to mitigate the confidentiality vulnerabilities presented in the MAVLink protocol using strong symmetric-key encryption algorithms. The proposed algorithms are Rabbit stream cipher, XXTEA stream cipher, and Salsa20 stream cipher. They both encrypt MAVLink messages rapidly while preserving the confidentiality of communication among the GCS and UAV.

Subsequently, an encryption mechanism *RC5* is used in [19] to secure the MAVLink communication protocol. In [20], [21], the Caesar cipher cryptography algorithm is used for data encryption of MAVLink messages between the ground station and the Micro Aerial Vehicles (MAV). However, in this method, the secret key is sent as a plain text to the drone, during the establishment phase. The key could easily be detected which breaks the whole security system. Moreover, the Cesar encryption algorithm used in this method is proved to be insecure and is vulnerable to cryptanalysis.

(*ii.*) Asymmetric key solutions: Another form of encryption is public-private key cryptography, also known as asymmetric encryption. It uses a couple of public/private keys. The public key is used for data or information encryption, and the private key, only known to the receiver, is used for the decryption process. The advantages of these approaches are their flexibility, scalability and skey management efficiency. However, these solutions can cause severe computational, memory, and energy overhead which are not suitable for constrained devices. RSA and ECC algorithms were used in [17] to encrypt of Aerial Robotics Communication.

(*iii.*) Identity-based encryption (IBE): The problem of public key cryptography resides in its dependency on the third-party authority that issues the certificates. To overcome the scalability and complexity issues, IBE is proposed by suggesting the idea to use known information that uniquely identifies users (e.g., phone number, email, etc.) as their public keys for data encryption and thus eliminates the necessity for certificates and Public Key Infrastructures (PKI). Despite the clear achievement in scalability and efficiency, IBE needed further refinements to become lightweight and consequently viable for the use in resources constrained devices such as UAVs.

The contribution in [22] is twofold. First, a hierarchical architecture has been designed for the UAV networks using identity-based encryption and bilinear pairing over elliptic curve cryptography (ECC) without compromising system security. Second, a lightweight cryptographic primitive is proposed using a probabilistic selective data encryption technique. The proposed method improves system performance and increases the efficiency of the transmitted message without affecting security. Stenography or data watermarking technique is used to reduce overheads and increase message confidentiality. In [23], a lightweight IBE scheme (IBE-Lite) is specially designed for resource-constrained IoD architecture. The proposed scheme facilitates the secure sharing of drones' data.

b: Integrity solutions

Integrity can be ensured using signature, hash functions, message authentication code (MAC) and authenticated encryption cryptographic primitives [43], [44].

In [24], the authors proposed to add a digital signature to the data packet using the UAV private key.

According to [25], the authors proposed to use authenticated encryption cryptographic mechanisms to enforce the integrity of the data.

In [18], the authors addressed two cryptographic implementations that can mitigate integrity vulnerabilities presented in the MAVLink protocol: the Poly1305 Message Authentication Code (MAC) and the Galois/Counter Mode (GCM). The GCM implementation adds an authentication code along with the ciphertext and the initialization vector. The authentication code is used to create an authentication tag that is used as a method for validating the message integrity. Poly1305 is a message authentication code (MAC) used to verify data integrity.

However, using Poly1305 or GCM increases the packet size due to the added padding, which increases the latency in UAV communications and reduces the energy efficiency of the autopilot processor. Moreover, Galois/Counter Mode is the most computation-extensive mechanism because it authenticates and encrypts every message using a slower method.

In [26], the authors proposed a data transfer and confidentiality services between the ground station and the drone. This protocol ensures the message integrity using a message authentication code (MAC) function using an alternative key computed based on the master key.

c: Availability solutions

Protecting the UAV against malicious availability attacks is important to succeed in UAV missions.

The contribution in [33] consists of proposing an estimation model based on estimated beliefs to detect the existence of a system threat. This work includes specific detection policies to maintain the availability of UAV network.

Authors in [12], [27] surveyed the potential DoS attacks that could cause serious availability issues in UAV systems. They also proposed strong authentication based solutions to mitigate these attacks.

d: Authenticity solutions

Authentication is essential to make sure that the GCS is controlling an authorized drone not a fake one, and that the UAV is sending its state or accepting commands from a legitimate GCS not from a hacked/fake ground station. In other words, authentication enables the UAV and the GCS to guarantee that they communicate with each other. Authentication techniques used in UAV networks are Symmetric key solutions, Asymmetric key solutions, and Digital signature.

(*i.*) **Symmetric key solutions:** A symmetric encryption algorithm AES-GCM is used in [18], [28] to ensure the authenticity of the transmitted signals. In [20], [21], the authors tackled the problem of authentication using the Caesar Cipher method in the MAVLink protocol. To guarantee a secured communication between the UAV and the GCS, the drone is authenticated at the beginning of the communication using Cesar Ciphering. Thus, GCS data will not be received by the UAV unless it gets authenticated, through sending an encrypted text similar to the one produced by the GCS.

The Message Authentication Code (MAC) algorithm is used to provide message authentication by using a symmetric key encryption technique. Authors in [12] claimed that attaching the MAC to each MAVLink payload could be suitable to verify the authenticity of the message.

(*ii.*) **Asymmetric key solutions:** The authors in [17] employed the use of Elliptic Curve Cryptography (ECC) for ensuring authentication between the GCS and the UAV. All feedback messages are encrypted with the drone's private key, which means that only the drone's public key can decrypt these messages. This procedure ensures that the receiver authenticates the origin of the received messages. On the other hand, the GCS's public/private keys guarantee that the UAV recognizes the message sent by the GCS so that the drone is authenticated.

(*iii.*) **Digital signature:** As we stated before, a MAVLink version 2.0 has been designed to support the packet signing mechanism and to bring security to MAVLink communications [8]. The signature represents the first forty eight bits of an SHA-256 hash of the secret key, payload, link ID, header, CRC, and timestamp. The 13 bytes (including signature, link ID and timestamp) must be appended at the tail of a MAVLink 2.0 message to ensure that messages are sent by trusted sources.

Despite packet signing has backward-compatibility and best portability as compared to other security alternatives, developers still need to upgrade their autopilots to MAVLink v2.0 to support packet signing [45].

However, adding packet signing security solution is not free. The cost consists in several factors including the computational overhead of 26 microseconds per packet, the increase of firmware size (packet signing code is 812 bytes) and the increase in power consumption due to the higher CPU and network traffic usages. This leads also to increase the communication time [45].

2) Intrusion Detection System (IDS)

It is essential to protect the UAV system against attackers by allowing to detect possible cyber attacks against the drone communication system. Intrusion Detection Systems (IDSs) are typically deployed to monitor the incoming communication, supervise and identify indications of abnormal activity or behavior. Some works in [29]–[38] have discussed the area of intrusion detection systems (IDSs) for UAVs. In what follows, we survey these existing studies based on the categories of the techniques proposed for intrusion detection including, the rule-based specification detection, anomaly-based detection, hybrid-based detection, and signature-based detection,.

a: Rule-based specification detection

This method is based on comparing the behavior of UAVs against a set of specified rules based on the expected behaviors of UAVs.

In [29], Mitchell and Chen proposed the BRUIDS intrusion detection mechanism, which aims to detect and protect a UAV system against security threats. BRUIDS is a rule-based specification detection technique for intrusion detection of compromised UAVs. The authors proposed a set of behavioral rules related to cyber-attacks constructed based on defined attack models to build a model of a normal UAV behavior.

Kim et al. [30] evaluated the behavior of attacks that target UAV. They proposed a behavior rule-based intrusion detection system for UAVs, in which the rules are specified according to these malicious anomalies to model a normal UAV behavior.

b: Signature-based detection

This method refers to the detection of known attacks based on predefined known signatures, features, and patterns. Such signatures are used to compare patterns with captured events to recognize and ensure the detection of possible intrusions. Although signature-based detection can easily detect known attacks, it is difficult to detect unknown or new attacks for which no pattern is available.

Authors in [31] proposed a technique (behaviometrics) for continuous authentication of data command transmitted by a ground station to the drone based on UAV behavior. The drone's behavior is specified using a set of flight commands, which are considered later as a unique signature to identify authorized UAVs and detect malicious commands stemmed from attackers.

c: Anomaly-based detection

Anomaly-based mechanism defines normal and baseline features to build a model of normal behavior profile and to follow any possible variation from the normal behavior. Anomaly detection usually uses statistic analysis, machine learning techniques, and game theory to enhance the detection of anomaly behavior of a monitored node and unknown attacks. The key benefit of this mechanism lies in its ability to detect new or unknown attacks when there are no predefined signatures of the unknown attacks.

Authors in [32] presented another IDS, where they proposed a prototype of UAV real-time monitoring system to control avionics and flight controller systems. Their approach adapts the recursive least squares technique to estimate UAV navigation sensor, controller parameters, and other related parameters. Through the application of this statistical method, the IDS can identify the system's parameters values, and the anomaly is detected whenever the monitored parameters deviate from their expected values during the flight.

In [34], the authors suggested an intrusion detection system based on Neural network and fuzzy learning algorithms to protect UAVs against a distributed denial-of-service (DDoS) attacks.

Authors in [35] developed an anomaly detection scheme based on Support Vector Machine (SVM) algorithm to detect cyber-attacks that target autonomous avionic systems.

The authors in [33] proposed and implemented an intrusion detection system to protect UAVs from attacks targeting data integrity. The detection of these attacks relies on a belief-based threat estimation model to reduce false positive and false negative rates.

Besides, the authors in [36] proposed a collaborative intrusion detection framework, named Security Game Framework (SGF), to protect UAV-aided network against lethal attackers. SGF is formulated based on the Bayesian game model to detect attacks accurately.

d: Hybrid-based detection

It takes the advantages of both rule-based, anomaly detection, and signature-based and combines them to catch known and/or unknown attack signatures and abnormal events.

In [37], authors investigated the use of an intrusion detection. The IDS uses an SVM learning algorithm to classify threats while monitoring the behavior of UAVs. A sequence of detection policies related to each cyber-attack is proposed based on a hybrid approach (rule-based detection and SVM-based anomaly detection) to model a normal UAV behavior.

The authors in [38] proposed a framework based on IDS that detects GPS spoofing attacks onboard the UAV. The IDS uses the attack-signature-based anomaly detectors as well as residual-based anomaly detectors. The Bayesian network takes anomaly detectors outputs as evidence to estimate a possible attack through Bayesian inference.

3) New emerging security solutions

Blockchain is an emergent technology that can be efficiently used solve the aforementioned security issues of MAVLink. Blockchain is originally used for recording financial transactions between entities in a distributed and decentralized manner. The transaction is verified collaboratively using trusted entities in the network, thus eliminating the need for a controlling authority. Moreover, transactions are stored on the Blockchain, which makes tampering with data extremely challenging because Blockchain relies on fully distributed cryptographic techniques. In this way, any modification on these transactions can be easily detected.

All these advantages led several researchers to consider this technology to deal with security issues in UAV network since Blockchain provides privacy, integrity, accountability, authorization, authentication, confidentiality, identity hiding and non-repudiation.

In [39], the authors included Blockchain and cloud storage in their framework to guarantee the UAV data integrity. This idea addresses the following objectives: trusted source, timing, and data integrity, accountability and resilient backend.

In [40], Blockchain is used to secure the communication among UAVs as they collaborate to make cooperative decisions and exchange data.

Sharma et al. [41] exploited the Blockchain features to securely relay drone information, especially in ultra-dense environments.

Furthermore, authors in [42] presented a system model based on the public Blockchain technology which provides security and privacy to the IoD network.

The approach proposed by [46] relies on Blockchain principles to identify compromised UAVs using trust rules and detect wrong information when a UAV is hijacked.

V. LITERATURE REVIEW

The MAVLink protocol has attracted the research community, and several contributions were proposed in the literature. Some of these works proposed some extensions and enhancements to the protocol (e.g., [47], [48], and [49]), while some other works presented the integration of the MAVLink protocol with the cloud and the Internet of Things (e.g., [50], [51]). Furthermore, recent works also addressed how to use MAVLink for autonomous agents and swarms (e.g. [52], [53], [52]). In this section, we present an overview of the recent research contributions that dealt with the MAVLink protocol.

A. EXTENSIONS AND ENHANCEMENTS

Recently, several research studies proposed extensions to the MAVLink protocol. In particular, the authors in [47], [48], and [49] extended the MAVLink protocol to support the multi-drones' cooperation.

In [47], the authors defined a new set of messages and data structures to manage a swarm and to enable drone-to-drone communication. The proposed messages are divided into two groups: swarm formation set and swarm maintenance set. In total, the authors developed six enumerations, six new commands, and thirty-three new messages. However, the authors did not validate their approach neither with a simulation nor experimental implementation. The concept of groups was introduced into the MAVLink protocol by adding two identifiers: 1) Group ID (group-wide), and 2) Group Member ID (group-internal). For the swarm maintenance, two factors were considered: drone replacement and recharging. The drone replacement is the process of finding a suitable alternative drone, then move all the mission-related data to that new drone, and finally the physical and logical replacement of the old drone with the new drone to finish the mission. The drone replacement is needed in several cases such as when the flight duration is undefined or longer than the capacity of the battery, or when the drone encounters some hardware problems. The recharge is a specific operation to extend the mission duration because of the limited battery capacity.

In [49], Erdelj et al. also proposed additional new MAVLink messages and commands support collaboration between drones in multi-UAV scenarios. The paper proposed an approach to ensure the continuity of the drone service, which means that when a drone has to leave the mission, it is *immediately* replaced with another. A proof-of-concept simulation was presented to show the effectiveness of the proposed algorithm. Also, the authors analyzed the performance of their system in terms of the total number of bytes induced by the collaboration.

The work in [54], presented a system that translates MAVLink messages to STANAG 4586 standard [55]. STANAG 4586 is a complete standard aiming at having NATO interoperability between UAVs from different countries. A bridge between MAVLink and STANAG 4586 created in a Raspberry Pi computer to make it easy to integrate with any UAV (Fig. 8). This work aimed to allow any GCS compatible with STANAG 4586 to be able to operate with any MAVLink-based UAV to accomplish the interoperability between the UAVs of the NATO's member. The proposed system was tested using the SITL simulator. Only important messages were translated such as waypoints messages because STANAG 4586 has a large set of messages. The messages were received successfully by the UAV, and the change on the parameters (long, lat, and alt) was not significant.

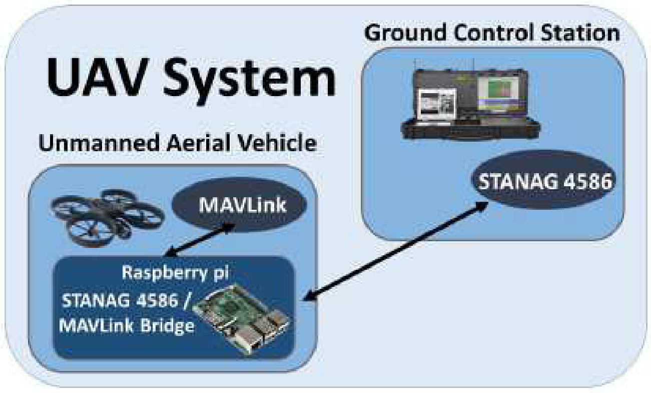


FIGURE 8: Bridge between MAVLink and STANAG 4586 [54]

B. CLOUD AND IOT INTEGRATION

The use of Internet-of-Things to communicate, manage and control multiple drones has gained a lot of interest.

In [51], the authors proposed Dronemap Planner, which is a cloud-based system to manage drones over the Internet and to offload heavy computation from the UAV to the cloud (e.g., Image Processing). The authors used the MAVLink protocol to send information from the drone to the cloud. The cloud forwards UAV data to corresponding users, which also send their commands to the drones through the cloud. The paper provided a complete implementation of the cloud-based management system and demonstrated how to monitor and control drones over the Internet effectively.

The work in [50] proposed a multi-UAV system for tracking and scanning missions in disaster response applications. A UAV is turned into an IoT device by embedding an 4G dongle with the drone autopilot. The MAVLink protocol is used for the communication between the UAV and the GCS. The authors proposed to use APM software on Windows 2016 OS running on a cloud server (Elastic Compute Cloud (EC2)). Experiments on a scanning mission for an area of size 0.16 km with real drones were conducted on a university campus. The missions were executed on EC2 and MAVLink through UDP and TCP protocols to exchange data and receive commands from the ground station. In addition, this paper analyzed the behavior of the MAVLink protocol using both UDP and TCP connections through 4G network.

In [56], the authors proposed an IoT architecture of different types of systems including UAVs, sensor devices and mobile phones. The communication between UAVs and the server was performed using the MAVLink protocol.

The work in [57] aimed to create an framework where the drones might be used to support the communication in areas having no or limited infrastructure. The focus is on the point-to-point connection between drone and car in an ad-hoc communication. The position information is received from the UAV's GPS using MAVLink protocol, which allows the communication between the drone's hardware and the Raspberry Pi.

C. SIMULATION AND MODELING

Several research works proposed simulation frameworks and evaluated the performance of MAVLink-based unmanned aerial systems.

The work in [58] analyzed the data loss and network latency in the communication channel between the ground station and the autopilot. The work in [59] describes a simulator for MAVLink-based UAVs. The proposed visual simulator combines SITL simulator which uses MAVLink for the communication with a scene model generated in OpenGL environment which allows visualizing the actions executed by the simulated UAV.

The work in [60] designed a Hardware-In-the-Loop testbed to test and simulate a control system for tail-sitter UAV. MAVLink is used to exchange messages between the vehicle and the GCS. The work in [61], introduced a new simulator called ArduSim which allows to control UAV flights in real-time and manage the communication between multiple UAVs. The MAVLink is used to send the control messages via TCP. In [62], the authors proposed a modular hardware-in-the-loop (HITL) simulation framework for multi-vehicle autonomous systems. The proposed simulation framework is compatible with any ground control station that operates MAVLink protocol over UDP.

The work in [63] addressed the development of a methodology for automated modeling of collaborative underwater vehicles. The MAVLink protocol is used to communicate with the Pixhawk autopilot and the Odroid-XU4 single board computer of the robot.

The work in [64] proposed a framework that combines different verification methods (simulation, symbolic, and statistical and model checking) to analyze the different stages of drone development. The proposed framework is composed of 3 main components: (1) Simulator: to specify the physical model of the drone, (2) formal executable language: used to specify the drone behavior and environment, (3) statistical model checking algorithm: used to analyze the system behavior. The communication uses the MAVProxy ground station that acts as a proxy that forwards the MAVLink commands to the drone.

D. APPLICATIONS

The MAVLink protocol is used as a communication protocol in a wide range of UAV applications, ranging from agriculture, construction, and environment monitoring. In what follows, we present an overview of some applications.

In **Agriculture applications**, the work in [65] proposed a system for crop monitoring through a multispectral camera mounted on a UAV. The MAVLink protocol is used for the wireless communication between the UAV to the ground station to send the multispectral crop images through telemetry. In the context of **environmental monitoring applications**, the work in [66] proposed an autonomous underwater vehicle system for cooperative environmental sensing. The MAVlink protocol is used to handle the communication between the modules, in addition the base station workstation. In [67], the authors developed an autonomous drone for the monitoring of oil and gas pipelines. All the data are transmitted using the MAVLink protocol, which ensures reliable communication within a 5-km radius. For **construction inspections**, the work in [68], proposed to integrate UAVs for inspection purposes and used MAVLink for communication between Ground Control Stations and UAVs. For **Maritime application**, the work in [69] proposed an architecture for maritime surveillance using battery-powered drones. The MAVLink protocol was used to exchange data between the autopilot and the flight duration enhancement system.

In addition, several recent research works addressed disaster management applications, such as [70] and [71]. In [70], the authors developed a fleet of UAVs for research and rescue applications. Rescue operations are sent from mobile phone to the drones through the MAVProxy ground station. The work in [71] proposed an agent-hardware integration architecture for search and rescue operations. The proposed architecture embeds JaCaMo agents on UAVs. The UAV's flight controller exchanges data via MAVLink protocol.

E. AUTONOMOUS AGENTS

There are several efforts have been made to make the UAVs autonomous. In [52], authors proposed an architecture to provide the UAV with the capability of locating itself using computer vision, modeling its environment, and planning and executing a 3D trajectories. The work was successfully tested with Solo from 3D Robotics which compatible with MAVLink. Gstreamer is used to receive the video feed, and Dronekit, which is compatible with MAVLink, is used to gain access to the vehicle. In [72], authors deal with precision landing problem for UAVs. As the GPS quality reduced when the drone gets close to the ground, this paper proposed using low cost adaptive fuzzy multi-sensor data fusion architecture. PX4FLOW sensor is used to get an accurate velocity measurement and recognize the moving features. PX4FLOW data was acquired using the MAVlink protocol. The work in [73] presents a system designed for multicopters to enable them from autonomous landing on a moving object. The presented system is based on visual tracking and landing of a marker on the object. The MAVLink protocol is used for the communication between the autopilot simulator and the rest of the system. The work in [74] proposed an algorithm to quantify the risk in a population and to find the optimal path with minimum risk. The proposed algorithm consists of two phases: first, identify the risk in a specific area by generating a risk map based on factors like population density, no-fly zones, sheltering, and obstacles. The second phase is the path planning algorithm to search for the best path that minimizes the risk. The communication between the autopilot and ROS is done using MAVLink protocol.

The work in [75] proposed an algorithm for offline path planning in a static environment. The proposed algorithm was tested with real UAV. The algorithm runs on a single-board computer onboard (Odroid XU4) To allow a completely autonomous flight. The proposed system was tested using Iris+, a ready-to-fly quadcopter from 3DRobotics to demonstrate the efficiency of the proposed system in practice. The Pixhawk flight control board is used to control the Iris+. The communication with the Pixhawk is done using MAVLink.

F. SWARM CONTROL

In [53], authors presented an algorithm to avoid collisions for a swarm of UAVs. The authors used small quadrotors with diameter 250mm equipped with GPS and distance sensors. An embedded Linux computer (Raspberry *Pi*) and Pixhawk autopilot board are used to control the UAVs. The control algorithm runs on the Raspberry *Pi* and is implemented as C++ package in the ROS platform. The MAVLink protocol is used for the communication between the Raspberry Pi and the Pixhawk.

The work in [76] proposed a decentralized control algorithm to be executed by each drone on the local onboard computer (Raspberry *Pi*). The proposed algorithm implemented as Python scripts, and in each iteration after computing the desired velocity, they sent to the flight control through MAVLink using DroneKit Python interface.

VI. MAVLINK RELATED SOFTWARE

There several types of software that support the MAVLink protocol, including (*i.*) ground stations, (*ii.*) Simulation frameworks. In the following subsections, we provide the main candidates for each category.

A. SUPPORTED GROUND STATIONS

A Ground Station is a software that communicates with the micro-vehicle through a serial or network interface by exchanging MAVLink messages. The communication can either take place over a serial port generally through a telemetry device or through a network interface inside a wireless local area network using UDP or the protocol. The advantage of telemetry devices is that they allow more extended communication range than traditional WLAN technologies, and can reach up to 5 km in range. This section presents the most common available ground stations. Table 6 shows a comparison of each GCS software discussed in this section.

- **QGroundControl:** The most commonly used ground station is QGroundControl [77]. This ground station was natively developed in C++ and also has a wrapper package for Android. It fully supports the MAVLink protocol in addition to Ardupilot and PX4 powered vehicles. QGroundControl has several functionalities, including defining and planning autonomous missions, full control of the vehicle, graphical visualization of the map and location tracking of the vehicle through its GPS coordinates. It also provides support for video streaming and changing the internal parameters of the autopilot, in addition to the calibration of the sensors of the autopilot. QGroundControl runs on different platforms, namely, Windows, Mac OS, IOS, and Android devices.
- **Mission Planner:** Is the second most popular ground stations for MAVLink-vehicles [78]. It is created by Michael Oborne and runs on Windows platforms only. Similarly to QGroundControl, Mission Planner also allows for planning an autonomous mission and making full control of the MAVLink vehicle. It has an additional feature of downloading and analyzing the log files of a mission. This means after completing any mission or operation of the unmanned systems, all internal parameters and state variables are stored in log files inside the autopilot, which can be download and analyzed by the Mission Planner. This helps you to understand how the autopilot behaves and provides a means to analyze any exotic behavior and analyze the performance of the autopilot.
- **APM Planner 2.0:** Is also a ground station software that is very similar to mission planner but is also available for both MAC OS and Linux environments [79]. It is considered as the best ground station to use for MAC and Ubuntu operating systems. APM Planner 2.0 provides almost all the functionalities of Mission Planner including analyzing log files.
- **MAVProxy:** Is a Linux-based ground station, that is primarily a command line interface and console-based interface with some graphical modules for map visualization and mission editing [80]. MAVProxy is written in Python. It uses a set of simple command to interact with Ardupilot autopilot. The advantage of MAVProxy is that it is portable and lightweight as compared to other ground stations, and also quite easy to use.
- **DroidPlanner:** For Android devices, DroidPlanner is also known as Tower software, is the best alternative for Android devices [81]. DroidPlanner relies on a Java ground station at a lower level that interfaces with the users through an Android GUI. It presents an excellent interface to interact with an autopilot through either serial telemetry interfaces and also network interfaces using both UDP and TCP like other ground stations. It also allows the user to configure the parameters of the autopilot and create missions on the fly. Nonetheless, it does not analyze log files like Mission Planner and APM Planner 2.
- **Universal Ground Control Software (UGCS):** It is a simple desktop software solution able to communicate with and control multiple unmanned systems simultaneously [82]. It also supports various autopilots from different manufacturers such as APM, Pixhawk, DJI, Mikrokopter, YUNEEC, Micropilot, Micro unmanned systems, Parrot (Ar.unmanned system) and other MAVLink compatible. UGCS supports several map layers and map providers. It provides a much more robust interface with many features such as NFZs and immersive 3D simulation. It runs on different platforms, namely, Windows, Mac OS, Ubuntu, iOS and Android devices [83].

B. UNMANNED SYSTEMS SIMULATORS

Unmanned systems simulator help simulating any environment and any unmanned systems activities in a digital environment to make easier the test and the validation of algorithms and protocols developed for the UAVs. The choice of the appropriate simulator depends on the objectives, the areas of application and the functionalities given by the simulator. Table 7 provides a comparison between unmanned systems simulators discussed in this section.

- **FlightGear:** Is a free, open-source flight simulator framework used for research and academic environments [84]. It works on different environment such as Windows, Mac, and Linux operating systems platforms [85]. The entire source code is available for modification and published under the General Public License (GPL). Aircraft models must be created by an external 3D modeling application. Typically, the UAV structure and features are described by an XML file. FlightGear can run Software-In-The-Loop (SITL) and Hardware-In-The-Loop (HITL).
- **UE4Sim:** In 2017, the Unreal Engine (UE4) is developed at King Abdullah University of Science and Technology, based on the open-source computer game engine Unreal Engine (UE4) [86]. The simulator has been designed to facilitate the integration of computer vision and machine learning techniques into a realistic looking 3D environment. UE4Sim gives accurate unmanned system physics, an evaluation tool based on the

TABLE 6: An overview comparison between GCS software

GCS software	Free/ commercial	Interface	Supported Autopilots	Platforms	MAVLink compatible	Implementation language	License
QGroundControl	Free	Graphical	PX4 Pro, ArduPilot (APM) or any vehicle that communicates using the MAVLink protocol.	Windows/Mac/Linux/iOS and Android devices	Yes	C++	Open Source (GPLv3)
Mission Planner	Free	Graphical	APM/PX4	Windows/Mac OS (Using Mono)	Yes	C#	Open source (GPLv3)
APM Planner 2.0	Free	Graphical	MAVlink based autopilots including APM and PX4/Pixhawk	Windows, Mac OS, and Linux	Yes	C++	Open source (GPLv3)
MAVProxy	Free	Command line and console based interface	Ardupilot MAVLink compatible	Linux	Yes	Python	Open source (GPLv3)
DroidPlanner	Free	Graphical	APM	Android Phones and Tablets	Yes	Java	Open source (GPLv3)
UGCS	Free version with limited capabilities	Graphical	APM, Pixhawk, DJI, Mikrokopter, YUNEEC, Micropilot, Microunmanned systems, Lokheed Martin, Parrot (Ar.unmanned system) and other MAVLink compatible multirotors, fixed wings and VTOLs	Windows, Mac OS, Ubuntu, Android, iOS	Yes	Human control interface with C#, Universal control server with JAVA, Vehicle specific layer with Java or C++	Not open source with a free licence available

latest advanced tracking algorithms, and a deep learning interface based on TensorFlow for autonomous driving without requiring manually collected training data.

- **X-Plane:** Is a commercial flight simulator produced by Laminar Research [87]. X-Plane simulator works on different environment, namely, Windows, Linux, and Android. It is certified by the FAA (Federal Aviation Administration) as a training simulator because it is more flexible and offers high fidelity simulation than the flight model when it is used with specific hardware configurations [88]. The flight model was created using the Plane-Maker, an application provided with X-Plane. This tool allows users to design any aircraft based on the vehicle's physical specifications. X-Plane uses UDP or TCP-based protocols to connect different instances through a network. X-Plane can exchange information through the UDP communication protocol, which guarantees high-speed data traffic.
- **Aerial Informatics and Robotics Platform (AirSim):** This simulator was produced in 2017 by Microsoft to develop and test deep learning, computer vision, and reinforcement learning algorithms for autonomous vehicle applications [89]. It is open-source, cross-platform, built on Unreal Engine 4 (UE4), and supports SITL and HITL with popular flight controllers such as Ardupilot and PX4 with the possibility of interfacing with MAVLink protocol to render the simulation more realistic [90].
  AirSim can retrieve data, images, control and interact with the vehicle based on APIs, via C++, Python, C# and Java languages. However, AirSim simulation is limited to quadunmanned systems. AirSim does not support ROS and cloud connectivity [91]. This simulator is also computation-extensive and needs advanced computing requirements as compared to other simulators.
- **Gazebo:** Is an open source simulation tool for robots and vehicles used for several applications [92]. This simulator was developed at the University of Southern California and currently managed by the Open Source Robotics Foundation (OSRF). It supports different robots and can simulate complex 3D virtual worlds with supporting various physical simulation engines and different sensors, to test robot designs and AI algorithms using real scenarios. Gazebo supports Ardupilot and PX4 with the ability to run Software In The Loop and Hardware-In-Tl-Loop. An API is provided allowing the creation of new sensors for Gazebo. Moreover, Gazebo is one of the most popular simulators since it enables multi-robot simulation, supports ROS and enables cloud connectivity [93]. However, it is too computationally demanding to simulate multi-vehicle operations in real-time [94].
- **Java Micro Air Vehicle Simulator (jMAVSim):** Is a Java UAV simulator developed by the PIXHAWK engineering team [95]. The main advantage of jMAVSim is that it is simple to use and lighweight. jMAVSim allows flying unmanned system type vehicles running PX4 around a simulated world. It supports the MAVLink protocol with the possibility to run SITL via UDP and HITL via a serial connection. It supports ROS and uses a Java3D library for graphical visualization. There is no possibility to integrate other sensors in the simulation.

VII. CONCLUSIONS

In this paper, we presented a comprehensive survey on the MAVLink protocol, which is a lightweight protocol for communication with unmanned systems. This survey addresses the need for having a technical reference for MAVLink-based systems developers. We thorough presented the characteristics of the MAVLink protocol version 1 and version 2 and their messages formats. Furthermore, we discussed the MAVLink security requirements, threats, possible solutions, and we presented the recent research works that dealt with security aspects of MAVLink. We also presented a comprehensive literature review of related research works about MAVLink.

We believe that this survey provides a handy reference for the large community of practitioners and developers to learn about the MAVLink protocol, in particular with the absence

TABLE 7: Comparison between unmanned systems simulators

Simulator	Main domain	Commercial/ free	Implementation language	Open source	Operating systems	License	Supported Vehicles	MAVLink compatible	ROS interface	SITL/HITL
FlightGear	unmanned systems	Free	C, C++	Yes	Windows, Linux, Mac OS-X, IRIX FreeBSD, Solaris	GNU/GPL	Aircraft, unmanned systems	Yes	No	Yes
UE4Sim	Vehicles	Free	Python, C++,	Yes			unmanned systems, cars	No	No	No
X-Plane	unmanned systems	Commercial	C++	No	Android, iOS, Linux, MacOS, WebOS, Windows	Proprietary with Free Trial	Plane	Yes	No	HITL
AirSim	unmanned systems, cars	Free	C++, Python, C#, Java	Yes	Windows, Linux	MIT	Iris (MultiRotor model and a configuration for PX4 QuadRotor in the X configuration)	Yes	No	Yes
Gazebo	Robots	Free	C++, JavaScript	Yes	Linux, Mac Windows	Apache V2.0	Quad (Iris and Solo), Hex (Typhoon H480), Generic quad delta VTOL, Tailsitter, Plane, Rover,	Yes	Yes	Yes
jMAVSim	unmanned systems	Free	JAVA	Yes	Linux, MacOs, Windows	BSD 3	Multirotor/Quad	Yes	Yes	Yes

of any technical coverage of MAVLink except some online documentation resources.

ACKNOWLEDGMENT

This work is supported by the Robotics and Internet of Things Lab of Prince Sultan University, and Jinan University.

REFERENCES

[1] A. D. Team, "Ardupilot." http://ardupilot.org/about/. Accessed: 2019-06-19.

[2] P. D. Team, "Paparazzi uav." http://wiki.paparazziuav.org/wiki/Main_Page. Accessed: 2019-06-19.

[3] "Autopilot - hangar." https://autoflight.hangar.com/autopilot. Accessed: 2019-06-19.

[4] P. D. Team, "Pixhawk web page." www.pixhawk.org. Accessed: 2019-06-19.

[5] "Multiwii." http://www.multiwii.com/. Accessed: 2019-06-19.

[6] S. Balasubramanian, "MAVLink tutorial for absolute dummies (part-i)." https://dokumen.tips/documents/mavlink-tutorial-for-absolute-dummies-part-i-tfwqtf2r7mmw7hksau-u9iabkndo9apguoisocmavlink.html, 2015.

[7] "Ardupilot telemetry devices." http://ardupilot.org/copter/docs/common-telemetrylandingpage.html. Accessed: 2019-06-19.

[8] L. M. Andrew Tridgell, "MAVLink 2.0 packet signing proposal," October 2015.

[9] L. Meier, J. Camacho, B. Godbolt, J. Goppert, L. Heng, M. Lizarraga, et al., "MAVLink: Micro air vehicle communication protocol," Online]. Tillgänglig: http://qgroundcontrol. org/mavlink/start.[Hämtad 2014-05-22], 2013.

[10] "MAVLink Common Message Set Specifications." https://mavlink.io/en/messages/common.html. Accessed: 2019-06-19.

[11] Y.-M. Kwon, J. Yu, B.-M. Cho, Y. Eun, and K.-J. Park, "Empirical analysis of MAVLink protocol vulnerability for attacking unmanned aerial vehicles," IEEE Access, vol. 6, pp. 43203–43212, 2018.

[12] A. Shoufan, H. AlNoon, and J. Baek, "Secure communication in civil drones," in International Conference on Information Systems Security and Privacy, pp. 177–195, Springer, 2015.

[13] K. Yoon, D. Park, Y. Yim, K. Kim, S. K. Yang, and M. Robinson, "Security authentication system using encrypted channel on uav network," in International Conference on Robotic Computing (IRC), pp. 393–398, IEEE, April 2017.

[14] L. Pike, "Keynote talk i: Building a high-assurance unpiloted air vehicle," in 2013 Eleventh ACM/IEEE International Conference on Formal Methods and Models for Codesign (MEMOCODE 2013), pp. 33–34, IEEE, 2013.

[15] S. B. Lorenz Meier, Seung-Hyun Seo, "sMAVLink request for comments," August 2013. Accessed: 19 February 2014.

[16] M. Podhradsky, C. Coopmans, and N. Hoffer, "Improving communication security of open source UAVs: Encrypting radio control link," in International Conference on Unmanned Aircraft Systems (ICUAS), pp. 1153–1159, IEEE, 2017.

[17] M. Han, "Authentication and encryption of aerial robotics communication," Master's thesis, San Jose State University, 2017.

[18] J. A. Marty, "Vulnerability analysis of the mavlink protocol for command and control of unmanned aircraft," Air Force Institute of Technology Wright-Patterson Graduate School of Engineering and Management, 2013.

[19] N. Butcher, A. Stewart, and S. Biaz, "Securing the mavlink communication protocol for unmanned aircraft systems," Appalachian State University, Auburn University, USA, 2013.

[20] B. S. Rajatha, C. M. Ananda, and S. Nagaraj, "Authentication of mav communication using caesar cipher cryptography," in 2015 International Conference on Smart Technologies and Management for Computing, Communication, Controls, Energy and Materials (ICSTM), pp. 58–63, May 2015.

[21] Hamsavahini, Rashmi, Varun, Swaroop, V. S. Praneeth, and S. Narayana, "Development of light weight algorithms in a customized communication protocol for micro air vehicles," International Journal of Latest Research in Engineering and Technology, 2016.

[22] M. S. Haque and M. U. Chowdhury, "A new cyber security framework towards secure data communication for unmanned aerial vehicle (uav)," in Security and Privacy in Communication Networks: SecureComm 2017 International Workshops, ATCS and SePrIoT, Niagara Falls, ON, Canada, October 22–25, 2017, Proceedings 13, pp. 113–122, Springer, 2018.

[23] C. Lin, D. He, N. Kumar, K.-K. R. Choo, A. Vinel, and X. Huang, "Security and privacy for the internet of drones: challenges and solutions," IEEE Communications Magazine, vol. 56, no. 1, pp. 64–69, 2018.

[24] J. Bian, R. Seker, and M. Xie, "A secure communication framework for large-scale unmanned aircraft systems," in 2013 Integrated Communications, Navigation and Surveillance Conference (ICNS), pp. 1–12, IEEE, 2013.

[25] R. Altawy and A. M. Youssef, "Security, privacy, and safety aspects of civilian drones: A survey," ACM Transactions on Cyber-Physical Systems, vol. 1, no. 2, p. 7, 2017.

[26] O. Zouhri, S. Benhadou, and H. Medromi, "A new adaptative security protocol for uav network," in Advances in Ubiquitous Networking 2, pp. 649–657, Springer, 2017.

[27] H. Shakhatreh, A. Sawalmeh, A. Al-Fuqaha, Z. Dou, E. Almaita, I. Khalil, N. S. Othman, A. Khreishah, and M. Guizani, "Unmanned aerial vehicles: A survey on civil applications and key research challenges," arXiv preprint arXiv:1805.00881, 2018.

[28] M. Verup and M. Olin, "Security models and exploitations in theory and practice for unmanned aerial vehicles," Master's thesis, Technical University of Denmark, 2016.

[29] R. Mitchell and R. Chen, "Adaptive intrusion detection of malicious unmanned air vehicles using behavior rule specifications," IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. 44, no. 5, pp. 593–604, 2014.

[30] A. Kim, B. Wampler, J. Goppert, I. Hwang, and H. Aldridge, "Cyber attack vulnerabilities analysis for unmanned aerial vehicles," in Infotech@ Aerospace 2012, p. 2438, 2012.

[31] A. Shoufan, "Continuous authentication of uav flight command data using behaviometrics," in 2017 IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC), pp. 1–6, IEEE, 2017.

[32] Z. Birnbaum, A. Dolgikh, V. Skormin, E. O'Brien, D. Muller, and C. Stracquodaine, "Unmanned aerial vehicle security using recursive parameter estimation," Journal of Intelligent & Robotic Systems, vol. 84, no. 1-4, pp. 107–120, 2016.

[33] H. Sedjelmaci, S. M. Senouci, and M.-A. Messous, "How to detect cyber-

attacks in unmanned aerial vehicles network?," in 2016 IEEE Global Communications Conference (GLOBECOM), pp. 1–6, IEEE, 2016.

[34] C. Rani, H. Modares, R. Sriram, D. Mikulski, and F. L. Lewis, "Security of unmanned aerial vehicle systems against cyber-physical attacks," The Journal of Defense Modeling and Simulation, vol. 13, no. 3, pp. 331–342, 2016.

[35] S. G. Casals, P. Owezarski, and G. Descargues, "Generic and autonomous system for airborne networks cyber-threat detection," in 2013 IEEE/AIAA 32nd Digital Avionics Systems Conference (DASC), pp. 4A4–1, IEEE, 2013.

[36] H. Sedjelmaci, S. M. Senouci, and N. Ansari, "Intrusion detection and ejection framework against lethal attacks in uav-aided networks: A bayesian game-theoretic methodology," IEEE Transactions on Intelligent Transportation Systems, vol. 18, no. 5, pp. 1143–1153, 2017.

[37] H. Sedjelmaci, S. M. Senouci, and N. Ansari, "A hierarchical detection and response system to enhance security against lethal cyber-attacks in uav networks," IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. 48, no. 9, pp. 1594–1606, 2018.

[38] D. Muniraj and M. Farhood, "A framework for detection of sensor attacks on small unmanned aircraft systems," in 2017 International Conference on Unmanned Aircraft Systems (ICUAS), pp. 1189–1198, IEEE, 2017.

[39] X. Liang, J. Zhao, S. Shetty, and D. Li, "Towards data assurance and resilience in iot using blockchain," in MILCOM 2017-2017 IEEE Military Communications Conference (MILCOM), pp. 261–266, IEEE, 2017.

[40] A. Kapitonov, S. Lonshakov, A. Krupenkin, and I. Berman, "Blockchain-based protocol of autonomous business activity for multi-agent systems consisting of uavs," in 2017 Workshop on Research, Education and Development of Unmanned Aerial Systems (RED-UAS), pp. 84–89, IEEE, 2017.

[41] V. Sharma, I. You, and G. Kul, "Socializing drones for inter-service operability in ultra-dense wireless networks using blockchain," in Proceedings of the 2017 International Workshop on Managing Insider Security Threats, pp. 81–84, ACM, 2017.

[42] N. K. M. C. S Aggarwal, M Shojafar, "A new secure data dissemination model in internet of drones," in The 53rd IEEE International Conference on Communications, (ICC 2019), IEEE, 2019.

[43] N. M. Rodday, R. d. O. Schmidt, and A. Pras, "Exploring security vulnerabilities of unmanned aerial vehicles," in NOMS 2016-2016 IEEE/IFIP Network Operations and Management Symposium, pp. 993–994, IEEE, 2016.

[44] H. Benkraouda, E. Barka, and K. Shuaib, "Cyber-attacks on the data communication of drones monitoring critical infrastructure," 2018.

[45] A. Davanian, F. Massacci, and L. Allodi, "Diversity: A poor man's solution to drone takeover..," in PECCS, pp. 25–34, 2017.

[46] I. García-Magariño, R. Lacuesta, M. Rajarajan, and J. Lloret, "Security in networks of unmanned aerial vehicles for surveillance with an agent-based approach inspired by the principles of blockchain," Ad Hoc Networks, vol. 86, pp. 72 – 82, 2019.

[47] T. Dietrich, O. Andryeyev, A. Zimmermann, and A. Mitschele-Thiel, "Towards a unified decentralized swarm management and maintenance coordination based on MAVLink," in 2016 International Conference on Autonomous Robot Systems and Competitions (ICARSC), pp. 124–129, May 2016.

[48] I. Zacarias, C. E. Leite, J. Schwarzrock, and E. P. de Freitas, "Control platform for multiple unmanned aerial vehicles," IFAC-PapersOnLine, vol. 49, no. 30, pp. 36 – 41, 2016. 4th IFAC Symposium on Telematics Applications TA 2016.

[49] M. Erdelj, O. Saif, E. Natalizio, and I. Fantoni, "UAVs that fly forever: Uninterrupted structural inspection through automatic uav replacement," Ad Hoc Networks, 2017.

[50] M. Aljehani and M. Inoue, "Communication and autonomous control of multi-uav system in disaster response tasks," in Agent and Multi-Agent Systems: Technology and Applications (G. Jezic, M. Kusek, Y.-H. J. Chen-Burger, R. J. Howlett, and L. C. Jain, eds.), (Cham), pp. 123–132, Springer International Publishing, 2017.

[51] A. Koubaa, B. Qureshi, M.-F. Sriti, A. Allouch, Y. Javed, M. Alajlan, O. Cheikhrouhou, M. Khalgui, and E. Tovar, "Dronemap planner: A service-oriented cloud-based management system for the internet-of-drones," Ad Hoc Networks, vol. 86, pp. 46 – 62, 2019.

[52] D. Soto-Guerrero, J. G. Ramírez-Torres, and J.-P. Gazeau, "Towards an autonomous airborne robotic agent," in Computational Kinematics (S. Zeghloul, L. Romdhane, and M. A. Laribi, eds.), (Cham), pp. 62–69, Springer International Publishing, 2018.

[53] R. G. Braga, R. C. da Silva, A. C. B. Ramos, and F. Mora-Camino, "Collision avoidance based on reynolds rules: A case study using quadrotors," in Information Technology - New Generations (S. Latifi, ed.), (Cham), pp. 773–780, Springer International Publishing, 2018.

[54] A. V. Rodrigues, R. S. Carapau, M. M. Marques, V. Lobo, and F. Coito, "Unmanned systems interoperability in military maritime operations: MAVLink to STANAG 4586 bridge," in OCEANS 2017 - Aberdeen, pp. 1–5, June 2017.

[55] M. M. Marques, "STANAG 4586-standard interfaces of uav control system (ucs) for nato uav interoperability," NATO Standardization Agency: Afeite, Portugal, p. 14, 2012.

[56] B. Uk, D. Konam, C. Passot, M. Erdelj, and E. Natalizio, "Implementing a system architecture for data and multimedia transmission in a multi-uav system," in Wired/Wireless Internet Communications (K. R. Chowdhury, M. Di Felice, I. Matta, and B. Sheng, eds.), (Cham), pp. 246–257, Springer International Publishing, 2018.

[57] S. A. Hadiwardoyo, E. HernÃąndez-Orallo, C. T. Calafate, J. C. Cano, and P. Manzoni, "Experimental characterization of uav-to-car communications," Computer Networks, vol. 136, pp. 105 – 118, 2018.

[58] S. Atoev, K. Kwon, S. Lee, and K. Moon, "Data analysis of the MAVLink communication protocol," in 2017 International Conference on Information Science and Communications Technologies (ICISCT), pp. 1–3, Nov 2017.

[59] P. śmigielski, M. Raczyński, and Á. Gosek, "Visual simulator for MAVLink-protocol-based uav, applied for search and analyze task," in Federated Conference on Computer Science and Information Systems (FedCSIS), pp. 1177–1185, Sep. 2017.

[60] J. Sun, B. Li, C.-Y. Wen, and C.-K. Chen, "Design and implementation of a real-time hardware-in-the-loop testing platform for a dual-rotor tail-sitter unmanned aerial vehicle," Mechatronics, vol. 56, pp. 1 – 15, 2018.

[61] F. Fabra, C. T. Calafate, J. C. Cano, and P. Manzoni, "Ardusim: Accurate and real-time multicopter simulation," Simulation Modelling Practice and Theory, vol. 87, pp. 170 – 190, 2018.

[62] L. Pannocchi, C. Di Franco, M. Marinoni, and G. Buttazzo, "Integrated framework for fast prototyping and testing of autonomous systems," Journal of Intelligent & Robotic Systems, Dec 2018.

[63] M. C. Nielsen, O. A. Eidsvik, M. Blanke, and I. Schjølberg, "Constrained multi-body dynamics for modular underwater robots - theory and experiments," Ocean Engineering, vol. 149, pp. 358 – 372, 2018.

[64] I. A. Mason, V. Nigam, C. Talcott, and A. Brito, "A framework for analyzing adaptive autonomous aerial vehicles," in Software Engineering and Formal Methods (A. Cerone and M. Roveri, eds.), (Cham), pp. 406–422, Springer International Publishing, 2018.

[65] U. R. Mogili and B. B. V. L. Deepak, "Review on application of drone systems in precision agriculture," Procedia Computer Science, vol. 133, pp. 502 – 509, 2018. International Conference on Robotics and Smart Manufacturing (RoSMa2018).

[66] F. Schill, A. Bahr, and A. Martinoli, Vertex: A New Distributed Underwater Robotic Platform for Environmental Monitoring, pp. 679–693. Cham: Springer International Publishing, 2018.

[67] Y. G. Kabaldin, D. A. Shatagin, A. V. Kiselev, M. V. Zhelonkin, and A. A. Golovin, "Drone-based autonomous robot diagnostic system for gas and oil pipelines in the arctic and far north," Russian Engineering Research, vol. 38, pp. 677–679, Sep 2018.

[68] H. Freimuth and M. König, "Planning and executing construction inspections with unmanned aerial vehicles," Automation in Construction, vol. 96, pp. 540 – 553, 2018.

[69] L. Rodríguez, J. A. Cobano, and A. Ollero, "Architecture of a flight endurance enhancement system for maritime operations with fixed wing uas," in ROBOT 2017: Third Iberian Robotics Conference (A. Ollero, A. Sanfeliu, L. Montano, N. Lau, and C. Cardeira, eds.), (Cham), pp. 171–182, Springer International Publishing, 2018.

[70] T. Riviere, H. G. Ayala, and J. Hajek, "The extension and implementation of the autonomous movement framework," in Proceedings of the 6th Annual Conference on Research in Information Technology, RIIT '17, (New York, NY, USA), pp. 7–10, ACM, 2017.

[71] M. S. Menegol, J. F. Hübner, and L. B. Becker, "Coordinated uav search and rescue application with jacamo," in Advances in Practical Applications of Agents, Multi-Agent Systems, and Complexity: The PAAMS Collection (Y. Demazeau, B. An, J. Bajo, and A. Fernández-Caballero, eds.), (Cham), pp. 335–338, Springer International Publishing, 2018.

[72] M. K. Al-Sharman, B. J. Emran, M. A. Jaradat, H. Najjaran, R. Al-Husari, and Y. Zweiri, "Precision landing using an adaptive fuzzy multi-sensor

data fusion architecture," Applied Soft Computing, vol. 69, pp. 149 – 164, 2018.

[73] J. J. Acevedo, M. García, A. Viguria, P. Ramón, B. C. Arrue, and A. Ollero, "Autonomous landing of a multicopter on a moving platform based on vision techniques," in ROBOT 2017: Third Iberian Robotics Conference (A. Ollero, A. Sanfeliu, L. Montano, N. Lau, and C. Cardeira, eds.), (Cham), pp. 272–282, Springer International Publishing, 2018.

[74] S. Primatesta, L. S. Cuomo, G. Guglieri, and A. Rizzo, "An innovative algorithm to estimate risk optimum path for unmanned aerial vehicles in urban environments," Transportation research procedia, vol. 35, pp. 44–53, 2018.

[75] H. D. Mathias and V. R. Ragusa, "Micro aerial vehicle path planning and flight with a multi-objective genetic algorithm," in Proceedings of SAI Intelligent Systems Conference (IntelliSys) 2016 (Y. Bi, S. Kapoor, and R. Bhatia, eds.), (Cham), pp. 107–124, Springer International Publishing, 2018.

[76] Q. Yuan, J. Zhan, and X. Li, "Outdoor flocking of quadcopter drones with decentralized model predictive control," ISA Transactions, vol. 71, pp. 84 – 92, 2017. Special issue on Distributed Coordination Control for Multi-Agent Systems in Engineering Applications.

[77] QGroundControl, "Open source mav ground control station." http://qgroundcontrol.org, 2010. Accessed: 2019-06-19.

[78] M. Planner, "Ground control station for apm(ardupilotmega)." http://planner.ardupilot.com/wiki/mission-planner-overview/. Accessed: 2019-06-19.

[79] Ardupilot, "APM 2.0 planner." http://planner2.ardupilot.com/, 2014. Accessed: 2019-06-19.

[80] Ardupilot, "Mavproxy." https://github.com/ArduPilot/MAVProxy. Accessed: 2019-06-19.

[81] Benemann, "Droidplanner 2." https://play.google.com/store/apps/details?id=org.droidplanner, 2014. Accessed: 2019-06-19.

[82] G. S. Software, "Ugcs pc mission planning." https://www.ugcs.com/. Accessed: 2019-06-19.

[83] A. I. Hentati, L. Krichen, M. Fourati, and L. C. Fourati, "Simulation tools, environments and frameworks for uav systems performance analysis," in 2018 14th International Wireless Communications Mobile Computing Conference (IWCMC), pp. 1495–1500, June 2018.

[84] A. R. Perry, "The flightgear flight simulator," in Proceedings of the USENIX Annual Technical Conference, 2004.

[85] P. Cao, X. Hu, and G. Zhang, "Interface research and flight control based on flightgear," in 2017 12th IEEE Conference on Industrial Electronics and Applications (ICIEA), pp. 397–402, June 2017.

[86] M. Mueller, V. Casser, J. Lahoud, N. Smith, and B. Ghanem, "Ue4sim: A photo-realistic simulator for computer vision applications," 2017.

[87] A. Meyer, "X-plane." Available: http://www.x-plane. com. Accessed: 2019-06-19.

[88] R. Garcia and L. Barnes, Multi-UAV Simulator Utilizing X-Plane, pp. 393–406. Dordrecht: Springer Netherlands, 2010.

[89] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "Airsim: High-fidelity visual and physical simulation for autonomous vehicles," in Field and service robotics, pp. 621–635, Springer, 2018.

[90] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "Airsim: High-fidelity visual and physical simulation for autonomous vehicles," in Field and Service Robotics (M. Hutter and R. Siegwart, eds.), (Cham), pp. 621–635, Springer International Publishing, 2018.

[91] T. A. Johansen and K. Klausen, "The path to Autonomous Inspection using an Unmanned Aerial Vehicle Brage Gerdsønn Eikanger," Master's thesis, Norwegian University of Science and Technology, June 2017.

[92] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566), vol. 3, pp. 2149–2154, IEEE, 2004.

[93] Gazebo, "Robot simulation made easy." http://gazebosim.org, 2014. Accessed: 2019-06-19.

[94] M. Read, C. Möslinger, T. Dipper, D. Kengyel, J. Hilder, R. Thenius, A. Tyrrell, J. Timmis, and T. Schmickl, "Profiling underwater swarm robotic shoaling performance using simulation," in Conference Towards Autonomous Robotic Systems, pp. 404–416, Springer, 2013.

[95] A. Babushkin, "Jmavsim." https://pixhawk.org/dev/hil/jmavsim. Accessed: 2019-06-19.