Preprint of:
M. J. Fernandez, P. J. Sanchez-Cuevas, G. Heredia and A. Ollero, "Securing UAV communications using ROS with custom ECIES-based method,"
2019 Workshop on Research, Education and Development of Unmanned Aerial Systems (RED UAS), Cranfield, United Kingdom, 2019, pp. 237-246.

Securing UAV communications using ROS with custom ECIES-based method

Manuel J. Fernandez, Pedro J. Sanchez-Cuevas, Guillermo Heredia, and Anibal Ollero
GRVC Robotics Lab, University of Seville
Seville, Spain
manfer@ieee.org, psanchez16@us.es, guiller@us.es, aollero@us.es

***Abstract*—This paper is about an application of a method based on the ECIES (Elliptic Curve Integrated Encryption Scheme) to improve the security against malicious attacks of the UAVs (Unmanned Aerial Vehicles) communications system. This system is focused on improving the security conditions in extreme situations and preventing the aircraft for man-made incidents and cyber attacks. The paper briefly describes the different attacks that can affect to the operation of UAVs and the security methods that, nowadays, are used to guarantee the security during the operations. Moreover, it presents a solution to a strong vulnerability detected in the classical scheme used in UAV. This scheme uses ROS (Robot Operating System) as the core of the communication system to interconnect different devices and nodes in this paper, it is demonstrated that if an Intruder is able to enter in the local network of the UAV system, he/she is also able to impersonate the GCS (Ground Control Station) of the UAV and take control of it leading to an undesirable maneuver or even a dangerous crash against a building or a person. The security system proposed to avoid this consists of a simplified method based on ECIES sending packets, between UAV and GCS, which uses ECDSA (Elliptic Curve Digital Signature) and are ciphered in RSA (Rivest–Shamir–Adleman). Thus, it is possible to guarantee that the high level computer of the UAV is able to identify the identity of their GCS and prevent of being commanded by an unauthorized Intruder. Both, the vulnerability and the solution proposed have been experimentally tested and validated through software-in-the-loop simulations and in a outdoor scenario using a small UAV.**

***Index Terms*—UAV,security,software,drone,attack**

Section I. Introduction

The application range of UAV (Unmanned Aerial Vehicles) is significantly growing year after year [1]. However, most previous studies have been only focused from the operation point of view or showing experimental applications of this new technology, such as aerial manipulation [2] or inspection of infrastructure [3]. Nevertheless, there are more aspects that must be considered in real applications to guarantee that this technology can be applied in security conditions and commercially exploited. It means that it is not only important to reach the application goal, but also it should be secure and reliable in terms of communication. Moreover, most UAV developments lack a dedicated security system in their communication system leading to situations in which an attacker can be a dangerous and unwanted Intruder. These attacks are ranging from a MiTM (Man in The Middle) or DoS (Denial-of-Service) to a GPS spoofing or UAV hijacking. Their consequences in an unmanned system are critical and extremely dangerous.

In general, data security and security in the communications are key tasks in most UAV applications. For instance, this work is one of the task in the RESIST H2020 European project [4], in which the UAVs must autonomously fly while they accomplish inspection tasks in bridges or tunnels in extreme and post-disaster situations caused by both, natural and man-made hazards. The second case includes man-made incidents and cyber attacks.

Most well known autopilots, as PX4 [5] or Arducopter [6], use the MAVLink (Micro Aerial Vehicle Link) [7] protocol to exchange information with the GCS (Ground Control Station), while other autopilots, such as the DJI A3 [8], have not direct support for MAVLink communication and they simply offer a SDK (Software Developer Kit) with ROS (Robot Operating System) [9] support. In the case of MAVLink communication, it is usually established with a HLC (High Level Computer) which can be onboard or offboard, through the ROS environment using MAVROS. This acts as a rospackage which translates the MAVLink messages into ROS topics. Thus, the use of ROS is commonly used to ease the communication between the autopilot and any other device connected to it or that could be in the same network. ROS acts like a middleware to exchange messages between the GCS and the UAVs. It consists mainly in a publish/subscribe pattern messaging. Moreover, from the application point of view, it is convenient to use an abstraction layer to manage UAV through ROS. In this way, it is possible to upgrade the interaction level creating a graphic user interface which could be fastly and intuitively operated by a human. Some examples of these interfaces are QGroundControl [10] and Mission Planner [11]. Nevertheless, these are mainly focused on the cases use of an standard UAV customer. Other approaches tries to solve this problem from a generic point of view creating an UAL (UAV Abstraction Layer) [12] which receives and apply commands through ROS services.

To sum up, ROS is highly considered for robotic systems by default, especially in research experiments. However, it is

an insecure environment to be deployed in the commercial use of a product. In general, in a ROS environment, it is enough to know the IP address of the ROS-master to read the information from the nodes which are running in the remote computer or even to execute processes. There are known options that can be implemented to get secure communications in ROS, such as SROS [13], which includes TLS (Transport Layer Security) support for communications. Currently it is in a stopped development state and developers advice that it must not be considered in production-grade. Therefore, it can not be considered an official security standard of ROS per se. Moreover, the proposal of configure and install SROS as secure environment for ROS is not essential to guarantee that the commands to execute in the UAV comes only from the authorized GCS. Protecting by a simpler way just the GCS-UAV communication solves the lack of security for this case of use.

The main contribution of this paper is to implement a security method in the communication UAV-GCS without losing the versatility and ease of use of the ROS environment. The solution proposed is a simplified method based on ECIES (Elliptic Curve Integrated Encryption Scheme) [14]. It consists of sending signed packets using ECDSA (Elliptic Curve Digital Signature) [15] and ciphered using RSA (Rivest–Shamir–Adleman) [16] in the messages exchanged between UAV and GCS. This sign is created using a private key and it can be verified with its known correspondent public key. The cipher is realized using the RSA public key. During the experimental results, the lack of security of the classical approaches is shown through the experiments performed, where an intruder attacks during a mission. Then, the same situation is executed, applying the security system proposed, foiling the plans of the attacker. Experiments are presented in the paper both simulation and real flight.

The paper is organized as follows: Section II exposes the main ways of a possible UAV external attack. Section III shows the situation proposed to be solved, with experimental demonstrations both in simulation and in a real flight. The core of the solution proposed is explained in Section IV. Then, the simulation and experimental results are showed in Section V followed by the conclusions in the last section.

SECTION II. SECURITY REVIEW IN UAV

This section aims to summarize different acts with malicious purpose that can affect UAVs. They are usually classified depending on if they affect to sensors or communications. For the case of sensors, it will be only considered the GPS, since this is the only onboard sensor which lets an external data input without modifying the environment, or being near the platform. That is, a potential door to be broken. In Figure 1 it is shown a common communication scheme configuration for real flights with some of the attack methods used in each one of these communications. They also will be commented along this section.

On the one hand, since the GNSS (Global Navigation Satellite System), as GPS (Global Positioning System) sensor, is the main source to close the position control loop

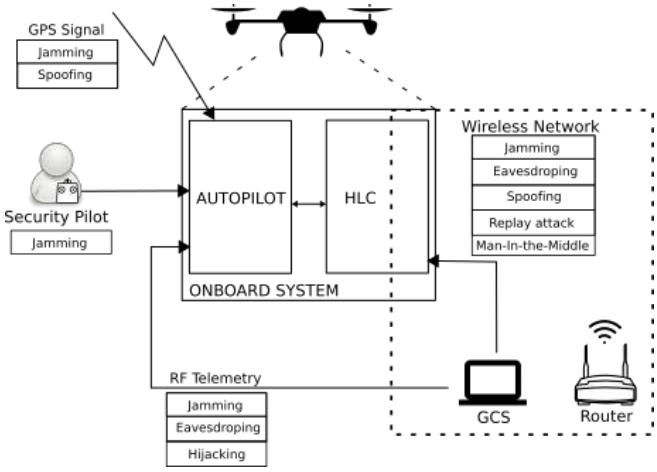

Figure 1. Scheme from possible attack ways and its more common techniques

of the UAV, it is one of the most commonly chosen to be attacked. GPS-based attacks can use spoofing signals [17] to fake an autonomous trajectory and guide the UAV to an undesirable area and, even land or crash it if the attacker wants. However, this is not only for UAVs, but it can also affect any system that uses a civilian GPS receiver, as unmanned surface marine systems or autonomous ground vehicles. Another vulnerability of this sensor is the GPS jamming, which can be used just to force GPS signal loss or as a previous attack to hijack the UAV through the spoofing method.

On the other hand, there are three main ways to establish a communication link with the UAV which also allows commanding the aerial vehicle to perform certain maneuvers. Thus, depending on the target, the attack method and its prevention measurements change:

- Radio Frequency Transmitter
  Mainly used by the pilot to control the UAV. In autonomous flights, the security pilot is essential to manually recover the control of the UAV in cases where the platform does not behave as it is expected or in emergency situations. If the pilot loses the RC link with the UAV due to an attack, the UAV platform can be completely controlled by the attacker, so the security in this role is critic. This can be initially done with a SDR (Software Defined Radio) jamming method [18]. Although commercial RF transmitter implements modulation methods to avoid possible attacks, the security methods in this communication is in continuous improvement [19].
- RF Telemetry communication
  The use of a pair of RF telemetry module is a common method to control the UAV by the GCS usually through the MAVLink protocol and using a specific software as QGroundControl [10] or Mission Planer [11]. Those programs can read and send MAVLink commands to the UAV autopilot directly through a telemetry module serial connec-

tion in the GCS. Some of these hardware modules have known potential vulnerabilities that can be exploited [20] such as the possibility of having an intruder in the communications. If it can hear the communication, depending of the security of the pair link the attacker, it could take the control of the aircraft and send commands hijacking the UAV. Thus, the use of modules specially prepared with security protocols in the physical layer of the transmission is highly recommended. In case of the attacker overpass the security layer of the telemetry module, the next wall to break is the security of the communication protocol. MAVLink, as it was commented in Section I, is a serial communication protocol used to send/receive messages between UAV and GCS in well known autopilots software as PX4 [5] or Ardupilot [6] among others. Prioritizing the lightweight in the communications, packets are sent in plain text without any security system, what implies to depend completely on the security implemented in the telemetry module. MAVLink v2 implements a symmetric key sign method to improve this lack of security, but it also adds specific ways where it can accept unsigned packets opening some ways to be exploited. Moreover, nowadays the v2 is not fully implemented in the autopilots previously mentioned because, although PX4 can use MAVLink v2, the signing functionality has not been implemented yet, so all packets are sent without sign. An alternative to avoid this problem would be, if there is a HLC onboard, disconnecting the RF telemetry module, so that the only way to establish a communication with the autopilot is through an onboard computer. To take this way is important to secure conveniently the HLC.

- Network communications
  The last way to establish the communication with the platform is through the HLC computer, which is usually a single-board computer (i.e. Raspberry-Pi [21] or Intel NUC [22]) onboard and connected to the autopilot. From the autopilot side, this computer acts like a bridge which links the GCS and the autopilot through a wireless network. This computer receives external orders from the GCS over the network, processes the data and sends them to the autopilot. In case of the autopilots that uses MAVLink protocol, this is usually by autopilot and the onboard computer to maintain a transparent communication for the rest of the network. Since the autopilot is not directly connected to the network, the main target for an attacker is this device, particularly the communication software that will be running to exchange information with the GCS. Thus, the possible attacks and the measures to avoid them are mainly the same that can be applied to any other remote computer in a wireless network (i.e. blocking not used ports, establishing SSH keys and disabling password-based SSH connections, etc). This communication is as secure as the security measures taken in the onboard computer. However, this kind of architectures usually considers the use of ROS as communication software between UAV and GCS and, as it was aforementioned in Section I, in this situation appears an important lack of security and vulnerability. ROS uses TCPROS protocol, based on the standard TCP/IP sockets, to send the messages, but they are not ciphered and guaranteed an access control to let who can or not use the services available in the ROS node running in the HLC of the UAV. These services are initially deployed by MAVROS [23], a ROS package which acts like ROS node interface between the autopilot and the ROS environment. It reads and sends MAVLink packets from the autopilot and offers this information to be easily consulted trough topics and services in the ROS environment. The ease of communication of ROS is not only between nodes running in the same machine. Assuming the case where there is a computer running ROS, with just changing an environment variable, a remote machine can read topics and use the services of the first without the user of the first knows it with the naked eye. The services offered by MAVROS let to manage the UAV and to realize unmanned mission or just for manual operations, as command changes in the flight mode or a landing maneuver among others. Thus, it is crucial to protect the inappropriate access to this services. There are several possible solutions to this problem depending on the security level needed of the final application. It could be to reject all external requests, or even implement a security method to be executed by all nodes. For the case presented in Section III, it will be enough with adding the security step in the GCS node and UAV node as it is explained in Section IV. Thus, the rest of the network can be connected freely to UAV node and to be informed about the current situation but with the guarantee that they can not interact with the UAV.

SECTION III. SCENARIO DESCRIPTION

In the RESIST project [4], UAVs fly autonomously inspecting bridges and tunnels in extreme events, where the security in communications is a key point to guarantee the overall security in a post-disaster situation. Figure 2 shows a simplified scenario of the mentioned application. The external network that appears called REDComm Network (Rapid Emergency Deployment mobile Communication infrastructure) [24] is a communications infrastructure prepared to be used in crisis situations. Through this network, the flight plan mission is received in the GCS. This acts like a secure bridge between that external network and the UAV to establish a last-step confirmation of the mission before being sent to the platform.

ROS is the choice selected as middleware because it is a publisher/subscriber environment which facilitates the communications and the deployment of nodes. However,

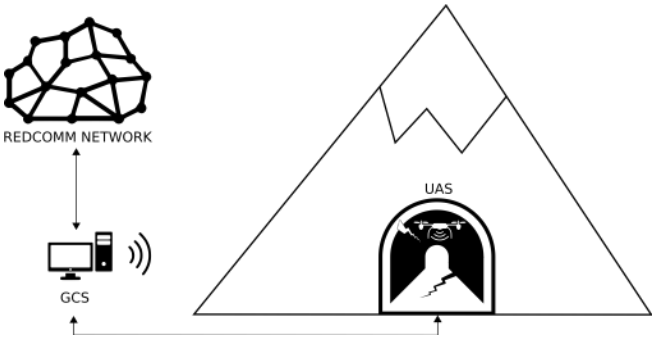

Figure 2. Simplified RESIST performance scenario

ROS does not implement security measures. The hardware/software architecture used is similar to the one presented in [25], as it is shown in Figure 3. Moreover, the use of the UAV Abstraction Layer (UAL) [12] provides a more ease and friendly user-interface to manage the UAV than MAVROS.

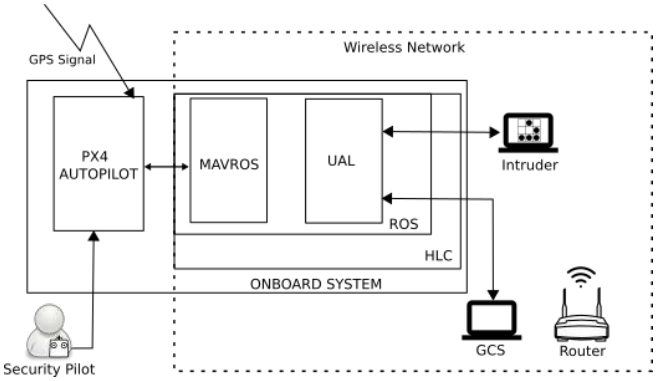

Figure 3. Simplified performance scenario

For the simulation case, a scheme like the one shown in Figure 4 has been set where two computers with Ubuntu Operative System have been used connected in the same local network. The computer one (IP 192.168.1.138) is in charge of running the PX4 autopilot in a SITL (Software In the Loop) simulation with Gazebo simulator [26] jointly with ROS and UAL. This computer will be the same who manages the UAV, so it will also act like the GCS of the system. A second computer in the same network (IP 192.168.1.141) will act as the "Intruder" which it is assumed that has violated the security of the REDComm network and has accessed to the local network where the HLC of the UAV is connected.

To make it as real as possible, it is assumed that the Intruder does not know any information about the UAV node. Then, the Intruder gets access to the network and expects an excess of truthfulness by the UAV manager. As it was aforementioned and thanks to the ROS vulnerabilities, the intruder can execute a basic port scanning order, for example, with the nmap tool, to look for the master port communication (by default it is 11311) because an IP with

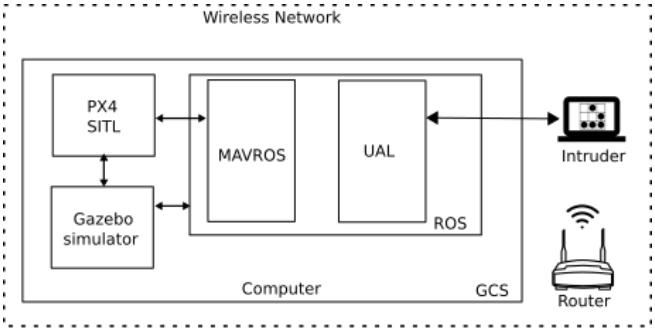

Figure 4. Simulation blocks-scheme of the simulation experiments

that port opened could be the ROS master node. It uses avahi daemon with avahi-resolve-address to get the name of the remote machine and add it to the hosts file of the system. ROS uses this file, through the Operative System, to know correspondences between the IP and its Domain Name to know, when consults *ROS_MASTER_URI* variable, where the master is located. That local system variable is set as *http://RemoteMachine:11311*, where RemoteMachine is the hostname of the remote machine found. A way to check if that machine is a ROS master node is trying to list the topics. Hereinafter, the intruder can consult any topic and use the services available.

The steps to discover the ROS master can be automatically executed just with a bash script[1], as in Figure 5. The same procedure could be applied in the real flight.

```
Found -> 192.168.1.138  grievous        grievous
Add to hosts file?
1) Yes
2) No
#? 1
Set as ROS_MASTER_URI?
1) Yes
2) No
#? 1
Found 1 ROS host available
```

Figure 5. Output of the bash script to discover ROS master automatically

Now, if the GCS commands the takeoff of the UAV and start a mission, the state of the UAV, along with more extra information, is published in topics. Then, the intruder can monitor the position of the UAV (Figure 6) and decide to apply a non controlled maneuver, such as landing (Figure 7) in a dangerous area. In Figure 7, it is shown a screen capture from Wireshark while it was sniffing the network. The top side of the figure indicates the IP source and the IP destination of the packet. The bottom side shows the data contained in the packet transmitted and highlight the command "/ual/land". A call to the ROS service to land the UAV has arrived to the master and it is been applied.

As it is shown in Figure 8 ROS does not inform that someone strange is monitoring neither that an Intruder is

1. https://gist.github.com/manfer33/5ba2c3d467170f9b71c6b5d76e78fd3e

```
header:
  seq: 98462
  stamp:
    secs: 3286
    nsecs: 964000000
  frame_id: "odom"
pose:
  position:
    x: 0.0276872012764
    y: -0.0169933885336
    z: 1.91556882858
  orientation:
    x: 0.0102642220816
    y: -0.00995239757336
    z: -0.0150648430912
    w: -0.999784317462
---
```

Figure 6. Topic from the ROS master node consulted in the Intruder computer

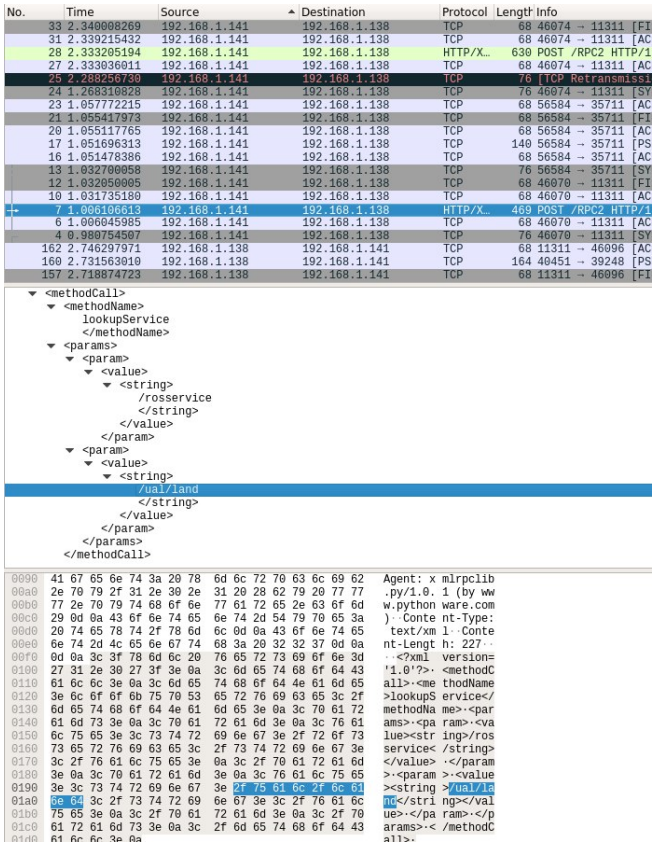

Figure 7. Intruder message detected usign Wireshark

using their services. The GCS can only observe how the UAV executes an non commanded maneuver.

```
INFO] [1564373469.115333608, 13.880000000]: UAL 1 ready!
INFO] [1564374896.393102208, 1439.628000000]: Parameter [MPC_TKO_SPEED] v
INFO] [1564374896.694561526, 1439.930000000]: Set flight mode [OFFBOARD]
INFO] [1564374896.694612938, 1439.930000000]: Trying to set [OFFBOARD] mo
INFO] [1564374896.996317412, 1440.232000000]: Set flight mode [OFFBOARD]
INFO] [1564374896.996365569, 1440.232000000]: Trying to set [OFFBOARD] mo
INFO] [1564374903.903424524, 1447.132000000]: Flying!
INFO] [1564375273.032991719, 1816.018000000]: Set flight mode [AUTO.LAND]
INFO] [1564375273.033045607, 1816.018000000]: Trying to set [AUTO.LAND] m
INFO] [1564375273.334075516, 1816.318000000]: Set flight mode [AUTO.LAND]
INFO] [1564375273.334116607, 1816.318000000]: Trying to set [AUTO.LAND] m
INFO] [1564375273.334142096, 1816.318000000]: Landing...
INFO] [1564375276.836299876, 1819.818000000]: Landed!
```

Figure 8. ROS Master does not inform about the presence of the external call

SECTION IV. SOLUTION PROPOSED

The solution proposed in this paper (Figure 9) is applied in the ROS nodes of GCS and UAV without changing the regular way of using ROS. GCS consists of a ROS python node that encrypts, signs and sends the message to the UAL node receiver running in the HLC. For that solution, the UAV receiver is an enhanced version of a UAL service which decrypts, verifies the signs of the message and, if the checks are correct, applies the command received. This method allows us creating a secure channel to send commands to control the UAV just with a regular ROS installation with an UAL. Having UAL in GCS is advisable for access to the messages to send automatically. It lets an easy way to any other entity in the network to receive information about the state of the platform and the mission. Also, if any entity wants to know how is going the mission, it only needs to have installed ROS and set ROS_MASTER_URI variable with the UAV IP. Considering that the main use will be in emergency situations, the fast to access to that information is key to inform about the situation. However, it is necessary to guarantee that despite everyone can heard, just one, the GCS, can command the UAV.

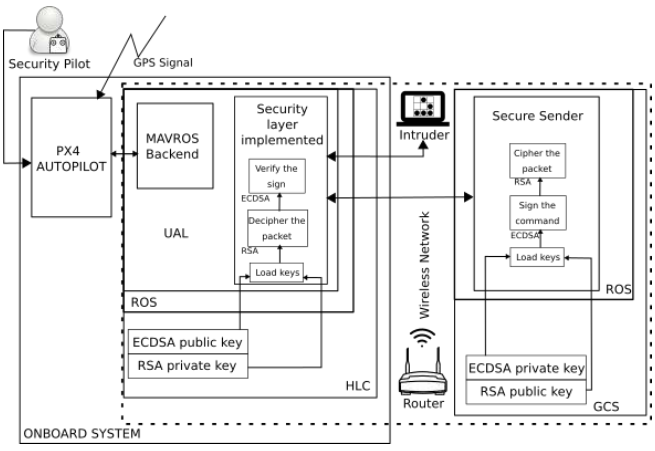

Figure 9. System blocks-scheme with sign verification implemented

The solution proposed is based on both RSA and ECDSA algorithms working together. They are asymmetric

key algorithms commonly known in cryptography and security. These algorithms use a pair of keys, public and private, that are generated using mathematical one-way functions. Public key can be shared freely but the private key must be keep private to maintain the security of the communication. RSA is used to encrypt the information to send. Its structure has been defined by the authors as it is shown in Figure 10. Public-key algorithm has been used for encryption, in contrast of symmetric, due to a versatility of use. By this way, this solution add the possibility of use more than one UAV in an easier way. The GCS does not need to have a symmetric key for each platform, it only needs to consult the correspondent public key. On the other hand, if the GCS has a problem a GCS backup does not need to depend of a symmetric key preloaded. In any case, the GCS only needs to keep properly the ECDSA private key.

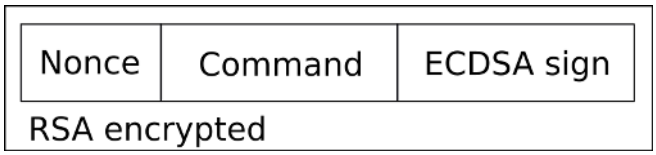

Figure 10. Structure of the packet sent from the GCS

In order to ensure that only one entity will be able to manage the UAV, the GCS and UAV will share between them (offline of the network) their correspondent public keys before starting the mission. In case of a failure in the GCS, or if it would be inoperative during the mission, it has been implemented a service in UAL which returns the RSA public key of UAV. Thus, it is recommendable to add more than one ECDSA public key during the preflight tasks to allow managing the UAV from a backup GCS.

A. ECDSA signing

ECDSA is a variant of Digital Signature Algorithm which uses elliptic-curve cryptography. It is used to verify the truthfulness of the information transmitted. ECDSA is chosen because it is more secure and faster than RSA [27]. It can get the same level of security as RSA with smaller keys, but it can not be used for cipher, so it is also necessary the use of RSA for that purpose.

The ECDSA key pair files are generated in the GCS through of openssl [28] software. To generate them the algorithm calculates math operations over the curve whose results depend on the curve selected. For this case, the curve secp256k1 has been selected. As example, this is the same curve used in the signing operations of Bitcoin and Ethereum blockchain and it is defined in Standards for Efficient Cryptography (SEC) [29].

The plain text information to be sent is signed with the ECDSA private key, allocated in the GCS. This sign is generated using the ECDSA private key, the message to sign and a random number. The sign is sent with the rest of information. Once the packet is received, the UAV node can verify that the sign received correspond to that message because it knows the ECDSA public key from GCS and the curve used. When this step is confirmed, the command is executed. Otherwise, the packet is rejected and the GCS is informed of a possible cyber-attack. By this way, only in the GCS can be generated the ECDSA key pair. In the packet, it has been added a nonce to avoid the possibility of a replay attack.

B. RSA encryption

RSA public key is used to encrypt the packet signed through math operations with the message to cipher. The algorithm is based on the factorization of the product of two large prime numbers. Just the private key owner can decrypt the message and read it. Thus, only the UAV will generate the RSA key pair. This key pair is generated usign the openssl software again. The public key is based on two large prime numbers, which form part of the private key. GCS encrypts data using UAV public key and sends it to the UAV. When it arrives, it is decrypted with the private key allocated in the onboard platform. The math operations applied with the RSA private key let recover the original message. NIST (National Institute of Standards and Technology) recommends an RSA key length of 2048 bits. This specification is not a problem for the Raspberry Pi as HLC in computation terms because generally there are not multiple transmissions per second of commands in an autonomous flight of UAV. The messages to be decrypted are calls to ROS services (i.e. execute a take off or go to waypoint) processed when they are received.

SECTION V. EXPERIMENTS

This section is focused on the demonstration and validation of the solution proposed in Section IV. First, it is presented a detailed description of the system used to accomplish the experiments, including both the software and the hardware. Second, two situations in which an Intruder acts maliciously, are presented and described to remark the vulnerabilities of a system that does not include a security system in their communications. Finally, the results of using the ECIES-based security method proposed in this paper are presented. Complete videos of the experiments can be consulted in [2].

A. System description

The flight tests have been carried out with a DJI F550 hexarotor frame, mounting DJI 2312E rotors, DJI 9x4.5inc propellers and a 4S 5300 LiPo battery for the propulsive system (Figure 11). The autopilot is a Pixhawk [30] running PX4 flight stack (v1.8.0) and the HLC is a Raspberry Pi, with Ubuntu Mate 16.04 as Operative System. The GCS is a common laptop which also runs Ubuntu 16.04 and has the function of monitoring and remotely control de high level task of the platform during an autonomous operation. Both the GCS and HLC use ROS Kinetic as ros version. In contrast, the Intruder uses Ubuntu 18.04 and ROS Melodic to show that the vulnerability does not depend on a specific version but it is a regular behaviour own of ROS. UAL (v2.2)

2. https://hdvirtual.us.es/discovirt/index.php/s/24K5Qpm9SaLnwF5

is used to establish the communication with the autopilot in the Raspberry Pi. The Intruder will work in other computer far away from the sight of the GCS operator. The architecture implemented follows the one presented previously in Figure 9.

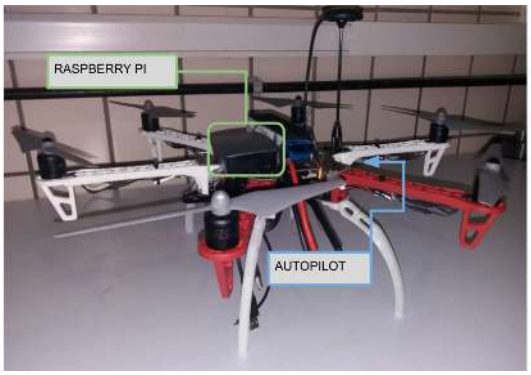

Figure 11. DJI F550 used in flight experiments

Moreover, as it has been commented in Section III, and has shown in Figure 12, before real flights Gazebo has been used as a simulation environment by PX4 running in SITL.

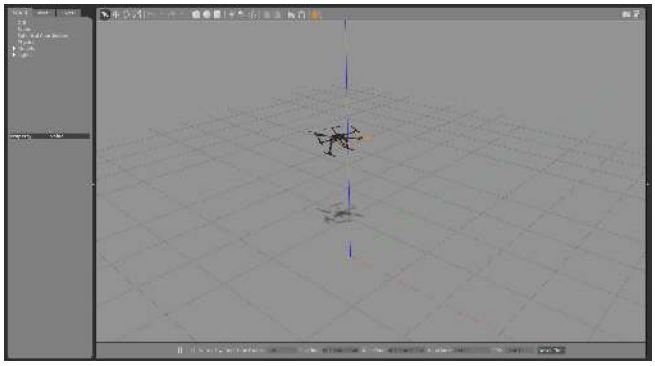

Figure 12. Gazebo as simulation environment

B. *Exploited cases*

During the experiments it has been applied the same steps than in simulation in Figure 5 to know the information of the master and get access to topics and services.

1) Hijacking. Once the Intruder access to the topics and services, it can monitor the UAV and hijack it during a trajectory commanding a new waypoint non controlled by the GCS. In this case, the Intruder also commands a land maneuver when the platform is out of the control of the pilot and the GCS.

As it shows in Figure 13, the GCS commanded initially a waypoint but during the operation the Intruder not only is able to take the control of the UAV, but also sent a new waypoint to divert the original trajectory followed by a land maneuver. The Figure 13 shows a sequence of the experiment where the green line is the path to the waypoint sent by the GCS and the red line is the path to the waypoint of the Intruder.

Figure 13. Video frames with the hijacked trajectory

In Figure 14 it is represented part of the data logged by the autopilot. The figure shows the path with the setpoint to reach and the position of the UAV along the flight. The desirable trajectory to follow is the green line and the commanded by Intruder is the red line.

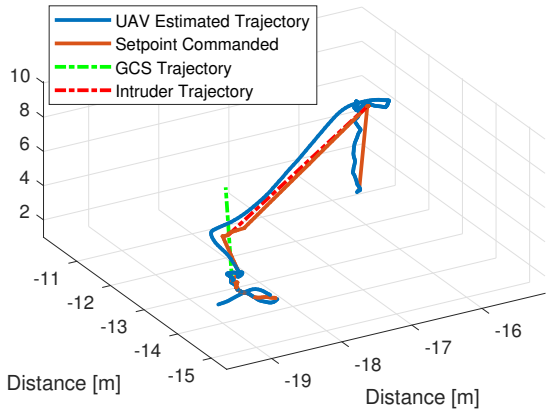

Figure 14. Trajectory of UAV hijacked

The Figure 15 shows the last part of the experiment where the Intruder execute a landing operation when UAV arrives to the waypoint.

2) Crash. One of the most extreme situation of hijacking will be if instead of commanding waypoints or a landing maneuver the Intruder marks as objective a place where the UAV crashes against a surface or a person. In the real flight test the UAV was commanded to impact with a obstacle during a hover state. In this case it was the security net of an outdoor flight field (Figure 16). The result is shown in Figure 17.

In Figure 18, the behaviour of the autopilot IMU gyroscope has been represented at the bottom of the figure while in the top is showed the transmitter log. The space highlighted is the offboard mode, where the UAV is flying completely autonomous through external references of the GCS or the Intruder. At the end of this graph, the gyroscope registers a great disturbance which corresponds with the

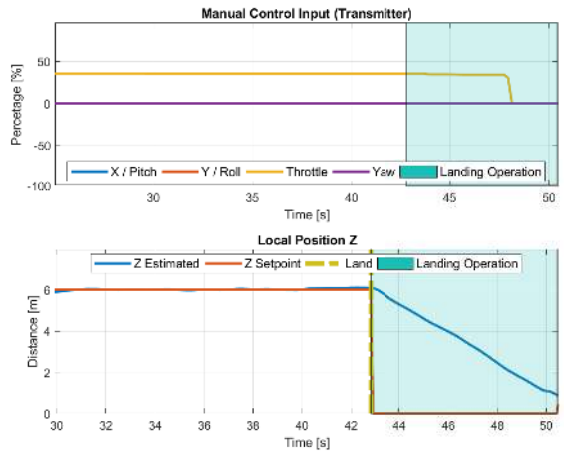

Figure 15. Autonomous undesirable landing operation

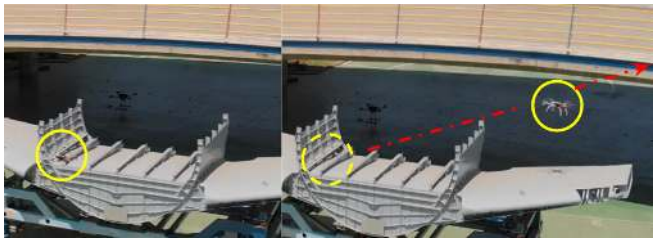

Figure 16. UAV hijacked to be crashed

time of the crash. In Figure 19 is presented the path followed by the UAV until it crashes against the net.

C. Implemented solution

To verify that the proposed solution is valid, it has been implemented in a new ROS service to execute a secure landing operation. In this case, the Intruder can still access to the information and monitor the UAV. Nevertheless, if it tries to manage the platform, the system rejects their requests of

Figure 17. Result of the crash

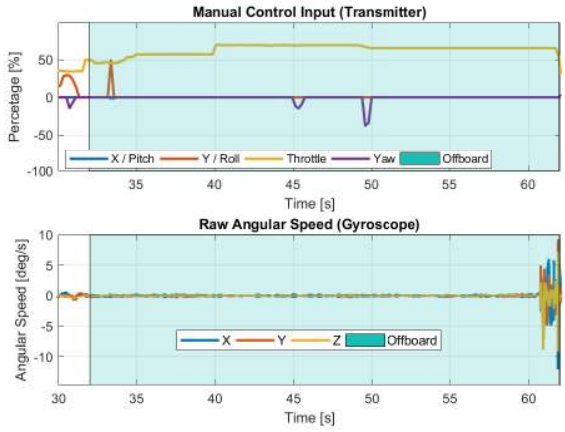

Figure 18. Log from sensor in impact time

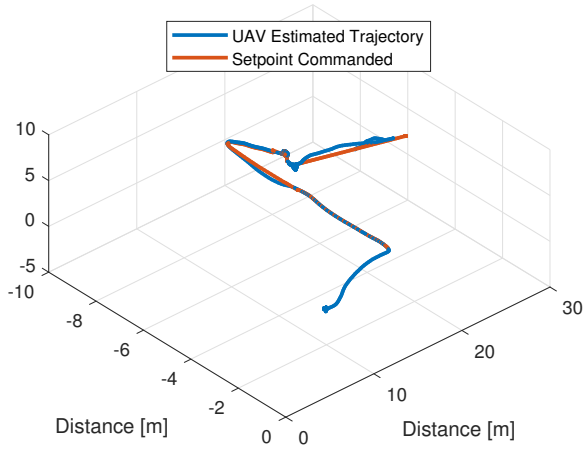

Figure 19. Log from sensor in impact time

commanding the UAV because thanks to the security system implemented, it is not recognized as an authorized UAV manager. This reject response can be observed in Figure 20.

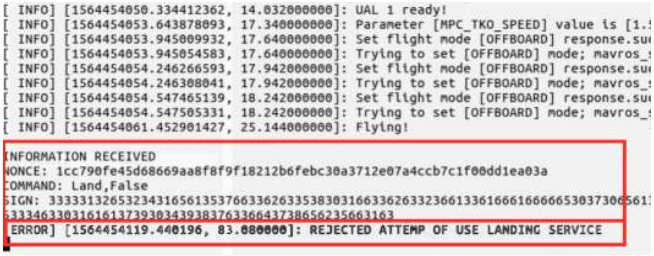

Figure 20. ROS reject landing operation

This experiment has considered even that the Intruder knows the security mechanism used in the communications and it generates it own key-pairs (ECDSA and RSA). Although the message could be correctly signed, it has not

been signed with the GCS private key. Thus, the check with the GCS public key results in error responding with a False, as it shows in Figure 21.

Figure 21. Intruder gets a fail in the try of command the UAV

SECTION VI. CONCLUSIONS

This paper has been briefly reviewed the current security systems implemented in UAVs and it has analyzed one of the most common vulnerabilities proposing a method to solve it. This security method aims to protect the managing of the UAV from possible intruders attacks and guarantees that the message received comes from the authorized GCS. The solution proposed has been demonstrated in real flights experiments along the Section V. It has been designed to reject any packet or modified replay packet sent by a Intruder with a verification step. By this way, it is possible to send a secure message point to point in an insecure environment as ROS.

Thus, the solution proposed based on the ECIES curves has been shown as a feasible technique to increase the security in the communication systems which use ROS, without losing the versatility and easiness provided by the ROS environment.

A potential improvement of this work consist of adding more than one public key in the HLC to be prepared in case of the GCS fails or it results attacked. Although in this paper is demonstrated the secure use of this implementation, to have any other backup GCS (with its own key pairs) is highly recommendable.

This method would be used in multi-UAV communications too, both for different commands for each UAV and for a multicasting communication. As the communication system is ROS, the security implemented in this paper is independent of the way that the messages arrive to the UAVs.

As future work, the authors plan to implement this method as UAL ROS services and study other security techniques which can improve the overall security of the system.

SECTION VII. ACKNOWLEDGEMENT

This work has been supported by the ARTIC Project, funded by the Spanish Ministerio de Economía, Industria, y Competitividad (RTI2018-102224-B-I00), the H2020 RESIST Project, funded by the European Commission (H2020-MG-2017-769066) and by the Spanish Ministerio de Educación, Cultura, y Deporte, FPU Program. Also, the authors would like to acknowledge the support given by the GRVC members, particularly to Rafael Salmoral and Rebeca Perez for their contribution during the experiments.

REFERENCES

[1] K. Valavanis and G. Vachtsevanos, "Handbook of unmanned aerial vehicles," in. Jan. 2015, pp. 383–384, ISBN: 978-90-481-9706-4. DOI: 10.1007/978-90-481-9707-1_135.

[2] A. Suarez, M. Fernandez, M. Perez, G. Heredia, and A. Ollero, "Lightweight and Compliant Long Reach Aerial Manipulator for Inspection Operations," pp. 6746–6752, 2018. DOI: 10.1109/IROS.2018.8593940.

[3] P. J. Sanchez-Cuevas, P. Ramon-Soria, B. Arrue, A. Ollero, and G. Heredia, "Robotic system for inspection by contact of bridge beams using uavs," *Sensors*, vol. 19, no. 2, p. 305, 2019.

[4] (). Resilient transport infrastructure to extreme events (resist) h2020 european project, [Online]. Available: http://www.resistproject.eu/aboutresist.

[5] L. Meier, D. Honegger, and M. Pollefeys, "Px4: A node-based multithreaded open source robotics framework for deeply embedded platforms," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015. DOI: 10.1109/ICRA.2015.7140074.

[6] (). Ardupilot project, [Online]. Available: http://ardupilot.org/ardupilot/.

[7] (). Micro aerial vehicle link (mavlink), [Online]. Available: https://mavlink.io/en/.

[8] (). Dji a3 - official web page, [Online]. Available: https://www.dji.com/bg/a3/info.

[9] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: An open-source robot operating system," in *ICRA Workshop on Open Source Software*, 2009.

[10] (). Mission planer - ground control station software, [Online]. Available: http://ardupilot.org/planner/.

[11] (). Qground control - ground control station for the mavlink protocol, [Online]. Available: http://qgroundcontrol.com/.

[12] F. Real, A. Torres-Gonzalez, P. R. Soria, J. Capitán, and A. Ollero, "Ual: An abstraction layer for unmanned vehicles," in *2nd International Symposium on Aerial Robotics (ISAR)*, 2018.

[13] R. White, H. I. Christensen, and M. Quigley, "SROS: securing ROS over the wire, in the graph, and through the kernel," *CoRR*, vol. abs/1611.07060, 2016. arXiv: 1611.07060. [Online]. Available: http://arxiv.org/abs/1611.07060.

[14] V. Shoup, "A proposal for an iso standard for public key encryption.," *IACR Cryptology ePrint Archive*, vol. 2001, p. 112, Jan. 2001.

[15] D. Johnson, A. Menezes, and S. Vanstone, "The elliptic curve digital signature algorithm (ecdsa)," *Int. J. Inf. Secur.*, vol. 1, no. 1, pp. 36–63, Aug. 2001, ISSN: 1615-5262. DOI: 10.1007/s102070100002. [Online]. Available: http://dx.doi.org/10.1007/s102070100002.

[16] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, 1978.

[17] A. Ranganathan, H. Ólafsdóttir, and S. Capkun, "SPREE: Spoofing Resistant GPS Receiver," 2016. arXiv: 1603.05462. [Online]. Available: http://arxiv.org/abs/1603.05462.

[18] K. Pärlin, M. M. Alam, and Y. Le Moullec, "Jamming of uav remote control systems using software defined radio," 2018. DOI: 10.1109/ICMCIS.2018.8398711.

[19] G. Miká and A. Németh, "Scfdm based communication system for uav applications," 2015. DOI: 10.1109/RADIOELEK.2015.7129014.

[20] (). Hijacking quadcopters with a mavlink exploit, [Online]. Available: https://hackaday.com/2015/10/15/hijacking-quadcopters-with-a-mavlink-exploit/.

[21] (). Raspbery pi - official web page, [Online]. Available: https://www.raspberrypi.org/.

[22] (). Intel nuc - official intel webpage, [Online]. Available: https://www.intel.com/content/www/es/es/products/boards-kits/nuc.html.

[23] (). Mavros - mavlink to ros gateway with proxy for ground control station, [Online]. Available: https://github.com/mavlink/mavros.

[24] (). Rapid emergency deployment mobile communication infrastructure (redcomm), [Online]. Available: http://www.redcomm-project.eu/description.html.

[25] I. Mademlis, A. Torres-González, J. Capitán, R. Cunha, B. Guerreiro, A. Messina, F. Negro, C. Le Barz, T. Gonçalves, A. Tefas, *et al.*, "A multiple-uav software architecture for autonomous media production,"

[26] 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, Sep. 2004, 2149–2154 vol.3. DOI: 10.1109/IROS.2004.1389727.

[27] A. V. Mota, S. Azam, B. Shanmugam, K. C. Yeo, and K. Kannoorpatti, "Comparative analysis of different techniques of encryption for secured data transmission," in *2017 IEEE International Conference on Power, Control, Signals and Instrumentation Engineering (ICPCSI)*, Sep. 2017, pp. 231–237. DOI: 10.1109/ICPCSI.2017.8392158.

[28] (). Openssl - robust, commercial-grade and general-purpose cryptography library, [Online]. Available: https://www.openssl.org/.

[29] P. Hess, "Sec 2: Recommended elliptic curve domain parameters," 2000.

[30] (). Pixhawk flight controller, [Online]. Available: https://docs.px4.io/en/flight%5C_controller/pixhawk.html.