

Improving Drone Security in Smart Cities via Lightweight Cryptography

Raffaele Pizzolante(✉), Arcangelo Castiglione, Francesco Palmieri, Angelo Passaro, Rocco Zaccagnino, and Samanta La Vecchia

Department of Computer Science, University of Salerno,
Via Giovanni Paolo II, 132, 84084 Fisciano, Italy
rpizzolante@unisa.it

Abstract. In recent years, *Unmanned Aerial Vehicles (UAVs)* have become increasingly popular. It is estimated that the UAV market will continue to grow to reach $27.4 billion by 2030. The ability of UAVs to travel long distances and acquire and process information has made these devices suitable in many areas. On the other hand, their widespread use has made UAVs an attractive target for cyberattacks. More precisely, most of such attacks are targeted to the *MAVLink*, which is the main communication protocol used by UAVs.

In this paper, we first focus on the main MAVLink security protocol, namely, the *MAVSec* protocol. MAVSec uses cryptography to encrypt messages exchanged between the UAV and *ground control station (GCS)*. However, several security weaknesses still affect the MAVSec protocol, mainly resulting from the lack of authentication and key exchange between UAV and GCS. To overcome the limitations of this protocol, we introduce a novel and modular security architecture based on lightweight cryptography [10,16]. Finally, the effectiveness and efficiency of the proposed solution have been assessed through a proof of concept implemented using the *software in the loop (SITL)* simulator.

Keywords: Drone Security · Lightweight Cryptography · Unmanned Aerial Vehicles (UAVs) · MAVSec · MAVLink · Unmanned Aircraft Systems (UAS) · software in the loop

1 Introduction

In recent years, drones, also known as Unmanned Aerial Vehicles (UAVs), have been spread in many governmental and civilian applications, varying from disaster management to agricultural controlling and from data collection (e.g., image sensing, environment sensing, and so on) to traffic monitoring [8]. However, it is essential to point out that originally drones were used for several specific military aspects.

The spread of commercial drones is expected to continue its growth and, according to a report of Tractica [1], it is estimated that, in 2025, this market

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2023
O. Gervasi et al. (Eds.): ICCSA 2023 Workshops, LNCS 14109, pp. 99–115, 2023.
https://doi.org/10.1007/978-3-031-37120-2_7

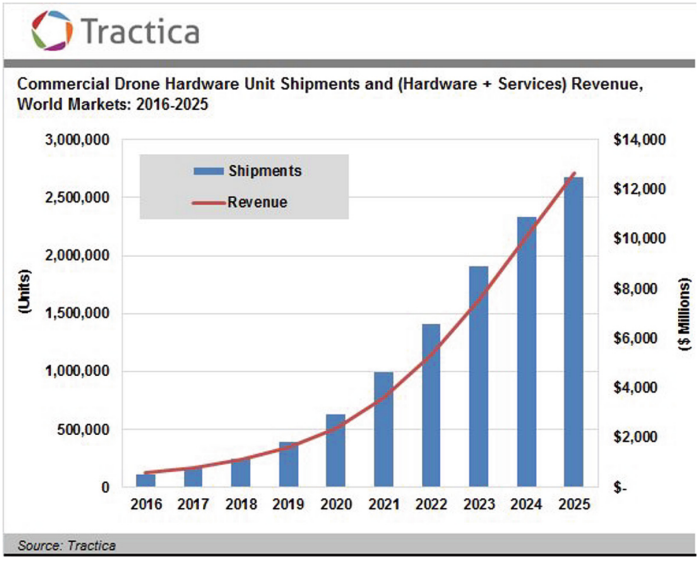


Fig. 1. Commercial Drone Hardware Unit Shipments and Revenue [1].

segment will reach a global income of $12.6 billion by 2025, as it can be observed from Fig. 1.

However, a drone, also known as *Unmanned Aerial Vehicle* (UAV), is a part of a more complex system denoted as *Unmanned Aerial System* (UAS). In detail, as shown in Fig. 2, a UAS is composed of the following key elements:

- One or more drones (UAVs);
- One or more radio controllers for piloting the drones;
- The *Ground Control Station* (GCS) or *Operation Control Unit* (OCU), which can be viewed as a control center that provides all the human facilities for the management of the UAVs;
- The communication links.

As discussed in [8], one of the most valuable and advanced fields of application for drones is related to smart cities. We can informally define a smart city as an urban region with advanced infrastructure and communications facilities [14]. Smart cities can offer many advantages for citizens, businesses, the government, and the environment. In this scenario, drones can be used in several fields of a smart city, such as traffic monitoring, controlling civilian security, and situation awareness. Drones could be more efficient than human-crewed planes. The abovementioned aspect is paramount since it can lead to relevant cost savings. Drones are also more flexible in many scenarios, such as hazard cases for people. In such cases, drones can provide effective and indispensable aid. Again, due to some of their physical characteristics (i.e., small size), drones can accurately perform specific actions close to a destination object or a small location.

By relying on these considerations, it emerges that drones provide tangible benefits in several scenarios in the context of smart cities. On the other hand,

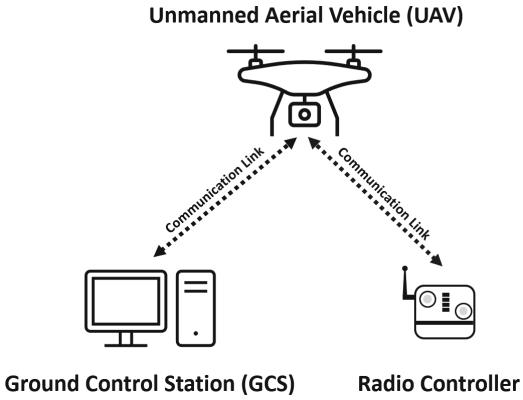


Fig. 2. The Elements of a UAS.

the drones and all the elements of a UAS system can be affected by several issues related to security and privacy [6,9,12,15], such as attacks on the flight controller and ground control station, attacks on the data link, etc. [6].

Motivation and Contribution. Most of such attacks are targeted to the *MAVLink*, i.e., the main communication protocol used by UAVs. The MAVLink protocol is a *lightweight* messaging protocol for communication with drones and among the components of onboard drones. This protocol ensures efficient, reliable, and extensible communication. Since the MAVLink protocol is widely used, it is affected by several security issues and attacks. [11,13]. MAVLink protocol implements a solution to solve authentication problems, including signatures in the MAVLink messages. However, this solution only solves a small portion of the issues affecting the MAVLink since, in general, the GCSs do not implement signature mechanisms. Other solutions have been proposed to improve the security of MAVLink, and most of them rely on symmetric cryptography. However, due to the hardware constraints and the limited computational capabilities, such solutions could not be suitable for all devices. Allouch et al. [5] proposed MAVSec, a solution based on cryptography that can solve security problems related to confidentiality and privacy. However, even the MAVSec is affected by some safety and scalability issues that do not make it entirely suitable for all modern scenarios where drones are used.

This paper addresses MAVSec issues by improving the communication link security between GCS and UAVs in smart city ecosystems.

First, we focus on the encryption algorithms used by MAVSec, which are insecure or unsuitable for devices with limited hardware capabilities. Furthermore, we observe that the problem of *key exchange* between the UAV and the GCS is not addressed in MAVSec, and this causes a lack of portability. Finally, found out that are no mechanisms that guarantee the *authentication* between the involved entities (i.e., GCS and UAV).

Starting from the considerations given above, in this work, we envision a novel solution that takes into account the following aspects:

- We integrate *lightweight cryptography* [7] in the MAVLink protocol and compare the obtained results with respect to the standard MAVSec protocol;
- We focus on the key exchange problem and address it with *lightweight key exchange* techniques. In particular, we first analyze such techniques to find the best of them in terms of performance, and then we integrate them into the MAVLink protocol.

Organization. In Sect. 3, we present the proposed solution, highlighting its underlying ideas and the adopted design choices; in Sect. 4, we focus on the experimental evaluation of our proposal; Finally, in Sect. 5 we conclude the paper, besides providing future research directions.

2 Background

This section highlights the key points and basic concepts to provide all the necessary background to understand better and contextualize our proposal. In particular, we outline the basic concepts and ideas underlying the MAVLink V2 protocol. More precisely, we focus first on the main aspects of the MAVLink V2, showing its motivations, logical functioning, and exchanged messages. Finally, we focus on the MAVLink V2 security requirements, highlighting the most common attacks proposed against this protocol.

2.1 MAVLink

A UAV can be viewed as an autonomous platform that can be easily programmed to perform missions with or without the intervention of a pilot. Using wireless technologies, a UAV typically communicates with a Ground Control Station (GCS). A GCS can monitor the status and control the actions of one or more UAVs. Furthermore, the communications between the GCS and the UAV occur through ad-hoc hardware and/or software called *autopilot*. More precisely, the autopilot is embedded into the UAV and allows the control of the UAV movements.

Some software has been proposed in the market to perform autopilot functions. Some of these software are:

- *ArduPilot* by *3DR*;
- *Paparazzi UAV* developed by *Ecole Nationale de l'Aviation Civil (ENAC)*;
- *Hangar autopilot*.

One of the most used autopilot software is *ArduPilot*, an open-source project. ArduPilot represents the basis of different types of autonomous systems, and its core is the *Micro Air Vehicle communication Link* (*MAVLink*) protocol [2,11].

The MAVLink protocol, released in 2009 by Lorenz Meier under the lesser general public license (LGPL), is a *lightweight* messaging protocol for communication with drones and among the components of onboard drones. Until now, two versions of MAVLink have been developed, MAVLink V1 and MAVLink V2.

2.2 MAVLink Insecurity

In this section, we report some possible attacks that can be performed.

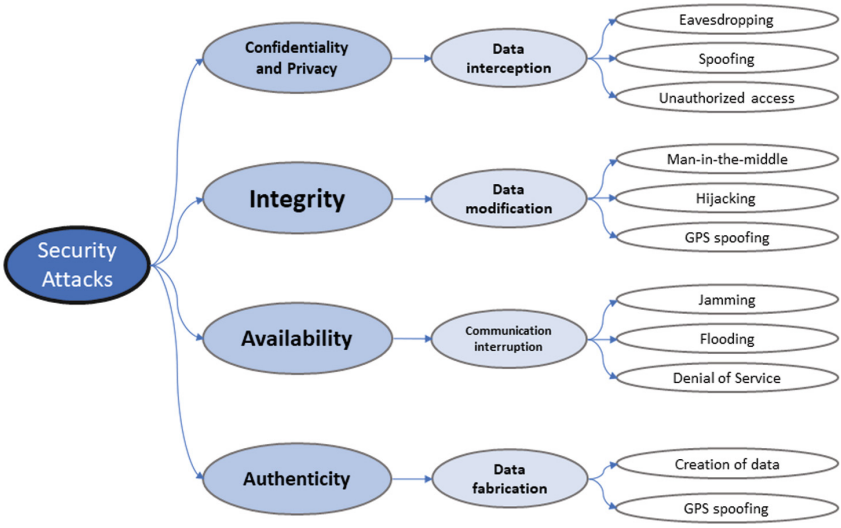


Fig. 3. The MAVLink Security Attacks.

As we showed in Fig. 3, the attacks can be classified according to the following categories:

- **Confidentiality and privacy attacks**: An attacker gains unauthorized access to confidential and sensitive information by intercepting data, commands, or messages exchanged between one or more UAVs and a GCS. In the following, we report some of the most important attacks in this category:
  - *Eavesdropping*: If encryption techniques are not put in place, an attacker can easily intercept communication between a UAV and a GCS, obtaining information of different types through traffic analysis;
  - *Spoofing*: MAVLink uses System IDs to identify elements of a network. However, due to the lack of encryption, such IDs are transmitted in cleartext. Therefore, an attacker could easily use such IDs for malicious purposes;
  - *Unauthorized access*: This attack occurs when an attacker gains access to the UAV and/or GCS.
- **Integrity attacks**: The integrity of MAVLink can be compromised by modifying the sent data. The following attacks are some examples of integrity attacks:
  - *Man-in-the-middle (MITM)*: An attacker can listen to the communication and reconstruct the exchanged commands. Again, the attacker can modify the control and telemetry data, then send this altered data to the GCS or UAV;

  - *Hijacking*: Whenever a MITM attack is successful, an attacker can send arbitrary commands to the UAV, making it believe the GCS sent them;
  - *GPS spoofing*: An attacker can send false GPS coordinates to the GCS. In this manner, the GCS believes that the UAV is in a different position from the real one.
- **Availability attacks**: Attacks that compromise the availability of MAVLink break the link used for data exchange between a UAV and the GCS. Such attacks cause an interruption of communication. In the following, we highlight some of these types of attacks:
  - *Jamming*: Breaking the communication link between the UAV and the GCS, the drone enters a *lost-link*. Then, the controller is no longer able to send commands. In this way, an attacker can try to take control of the drone;
  - *Flooding*: An attacker sends many packets (SYN, UDP, or ICMP) to the UAV and/or the GCS to overload them/it. In this manner, the GCS and/or the UAV are/is unable to communicate correctly;
  - *Denial of Service*: As in the previous attack, the UAV and/or the GCS receive(s) many packets. However, in this case, an attacker aims to obtain a break in the communication between the parties (i.e., UAV and GCS).
- **Authenticity attacks**: These attacks, which try to make the GCS/UAV believe that the forged data is authentic, can comprise the following actions
  - *Creation of data*: An attacker can analyze the exchanged information between the UAV and the GCS, trying to create forged messages to be sent to one of the two parties;
  - *GPS spoofing*: The UAV connects to the GCS via wireless links to exchange data and commands. However, an attacker could successfully spoof one or more MAVLink command(s) since the wireless environment is open.

3 The Proposed Solution

Although the MAVSec V2 protocol represents an attractive solution, it is still affected by some limitations, which make its use not fully compliant with the security requirements defined in the previous section. In particular, from a critical analysis of the MAVSec protocol, it emerged that it is affected by the following limitation.

First, we focus on the encryption algorithms used by MAVSec, i.e., *RC4* and *AES*. Actually, RC4 cannot guarantee a good level of security. On the other hand, as discussed in the previous section, AES is unsuitable for devices with limited hardware capabilities.

Furthermore, we observe that the problem of *key exchange* between the UAV and the GCS is not addressed in MAVSec. More precisely, the keys are hard-coded in the source code, and this causes a lack of portability. Finally, no mechanisms guarantee the *authentication* between the involved entities (i.e., GCS and UAV).

Starting from the considerations given above, in this work, we envision a novel solution that takes into account the following aspects:

- We integrate *lightweight cryptography* [7] in the MAVLink protocol and compare the obtained results with respect to the standard MAVSec protocol;
- We focus on the key exchange problem and address it with *lightweight key exchange* techniques. In particular, we first analyze such techniques to find the best of them in terms of performance, and then we integrate them into the MAVLink protocol.

This section presents the proposed solution, highlighting its underlying ideas and the adopted design choices. We remark that the proposed solution is modular and can be adapted according to the different operational scenarios. In particular, our solution, based on lightweight cryptography, can be modeled by the following four phases:

1. **Certificate Generation**: In this phase, the certificates for the GCS and the UAV are generated by the GCS;
2. **Certificate Broadcasting**: In this phase, the UAV broadcasts its certificate;
3. **Certificate Management**: The GCS receives a certificate and checks its validity. Then, the following two cases may occur:
   - If the received certificate is not valid, it is ignored;
   - Otherwise, the GCS generates the shared key and sends its certificate to the UAV (if the certificate is valid). The UAV then checks the certificate's validity and generates the shared key. The certificate exchange process is shown in Fig. 4
4. **Secure Data Exchange**: Once both the UAV and the GCS have generated the shared key, the exchange of encrypted messages begins.

3.1 Used Technologies

To realize the proposed solution, we use the following technologies:

- *MAVLink v2*;
- *QGroundControl (QGC)* [3]: QGC is an open-source and well-known cross-platform GCS. QGC is entirely developed in C++ and fully supports MAVLink.
- *SITL (Software in the Loop) Simulator* [4]: SITL Simulator permits to simulate Plane, Copter, or Rover without any specific hardware. More precisely, SITL is a build of the autopilot code that allows testing the behavior of a UAV without using any ad-hoc hardware.

3.2 Certificate Generation

In our solution, the GCS plays the role of the *Certificate Authority (CA)*. The certificates for the GCS and UAVs are created by an ad-hoc *generator* that we will describe below.

In detail, we include two structures in the proposed generator:

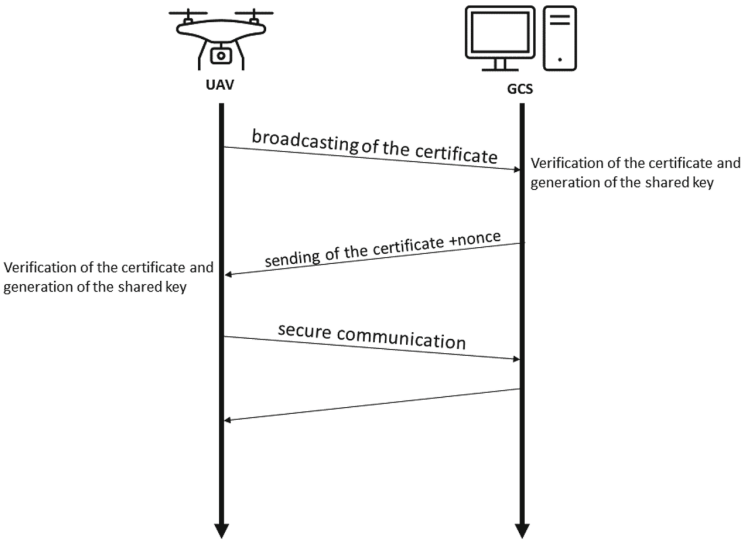


Fig. 4. Exchange of certificates.

- The first one, denoted as *info_s*, contains the information needed to identify the drone. In the following, we report some of the most relevant information contained in such a structure:
  - Serial number of the certificate;
  - Name and identification (ID) of the device;
  - Entity that has signed the certificate and validity of this certificate;
- The second one, denoted as *mavlink_device_certificate*, contains the certificate and other information, such as the *private* and *public key* of the issuer and the *sign* of the certificate. Notice that MAVLink uses these two pieces of information, which are not transmitted when the certificate is sent.

First of all, the certificate for the GCS is generated. In particular, the generator obtains the necessary information to create the certificate (i.e., the issuer, the subject, the device name, and the validity period). Then it generates the pair of keys, which will be used to generate the *shared key*. Subsequently, once the certificate for the GCS is signed, the certificate for the UAV can be generated.

3.3 Certificate Broadcasting

To broadcast the certificates on the network, we define and include two new messages in the MAVLink protocol: one message is intended for the UAV, and the other for the GCS. We remark that to include such new messages in the MAVLink protocol, we need to define their structures in an XML file (i.e., *common.xml*) within the *message_definitions* element. We also define and implement the features needed to interact with certificates and integrate them into

the MAVLink protocol. Again, regarding the MAVLink implementation, the features mentioned above have been included in the *mavlink_helpers.h* file. At the same time, the auxiliary structures have been included in *mavlink_types.h*.

In detail, we extended the MAVLink with several new functions. In the following, we outline the most important of them:

- *mavlink_read_certificate*: This function enables reading a certificate from a given file and storing its information;
- *mavlink_check_remote_certificate*: This function checks a remote certificate. First of all, this function checks the validity of the date and then the sign of the certificate;
- *mavlink_get_remote_key*: This function enables access to the keys agreed with remote devices;
- *mavlink_set_remote_key*: This function generates the *shared_key* between two devices, and stores it into the *remote_keys* by also updating the status;
- *mavlink_compute_iv*: This function generates a pseudo-random IV for a specific remote key;
- *mavlink_set_iv*: This function receives an IV and saves it for the specific remote key.

Once we have modified the implementation of the MAVLink protocol, we also modified ArduPilot and QGC according to such new features for the management of the certificates.

3.4 Certificate Management

Besides introducing the new certificate management functions, we modified the ArduPilot and the QGC to allow the certificate exchange. We briefly outline the basic ideas behind the introduced modifications in the following. For what concerns the ArduPilot software, we highlight that it is equipped with a scheduler. Such a scheduler works in the following manner: each time it finds a message, it tries to send it by calling the corresponding function. For example, when the *HEARTBEAT* message is found in the scheduler, the scheduler invokes the *send_heartbeat* message and tries to send this message.

Starting from the considerations given above, we provide the scheduler with a new message, denoted as *CERTIFICATE*. In addition, we include the *send_certificate* function in the scheduler. These modifications allow the scheduler of the ArduPilot software to adequately manage the sending of a certificate when the *CERTIFICATE* message is received.

Furthermore, we modified the QGC software component for receiving incoming messages. The basic idea behind this modification is that when a GCS receives a *certificate* message, it verifies if a key has already been set for the device that has sent such a message. At this point, if the key is not set, the GCS *deserializes* the received message and checks if the certificate is valid. If the certificate is valid, the GCS sets the shared key and the IV. Finally, the GCS reads its certificate, which is serialized so that it can be sent to the UAV. In this manner, the exchange of certificates can be completed.

3.5 Secure Data Exchange

To exchange data securely between the GCS and UAV, we modified three functions in the MAVLink implementation. In detail, we redefine the following functions included in the *mavlink_helpers.h* file:

1. *mavlink_finalize_message_buffer*: This function is used by the GCS to compute the checksum of the payload;
2. *_mav_finalize_message_chan_send*: This function is similar to the *mavlink_finalize_message_buffer*, but it is used by the UAV;
3. *mavlink_frame_char_buffer*: This function parses the received message.

In detail, we modify the functions *1.* and *2.*, enabling them to encrypt the payload before the checksum computation. More precisely, the ID is checked before the encryption of the payload. Then, if the ID denotes a *heartbeat* message or matches one of the two certificates, the payload is not encrypted. We remark that a heartbeat message is a message the UAV sends to the GCS to inform that it is alive and contains some information. When the ID check has been carried out, the function obtains the remote key and performs the encryption.

Finally, we modify function *3.* to enable the decryption of the received message. However, we remark that this encryption can occur only after the checksum's correct verification.

4 Experimental Test Results

In this section, we focus on the experimental evaluation of our proposal. In particular, we highlight and discuss the performance of the proposed solution, comparing it with some other proposals defined in state-of-the-art.

4.1 Testing Environment and Benchmark

The used testing environment is composed of a laptop with the following hardware characteristics:

- *CPU*: Intel i5-8250U, with 4 cores and 8 threads;
- *RAM Capacity*: 8 GB.

The SITL and QGC were run on the laptop described above during the testing phase. In particular, the SITL simulator has been run on a virtual machine (VM) with the following assigned resources: 2 CPU threads and 3192 MB of RAM. On the other hand, the GCS software has been run on the host.

Moreover, to carry out the testing activities, we define a *mission* composed of three waypoints. This mission is used as a test for all the considered algorithms, as shown in Sect. 4.1.

In Fig. 5, we report the three waypoints, denoted as *T*, *2*, and *3*, characterizing the mission used for the testing activity. The UAV starts from *T* and proceeds towards *2* and *3*. Finally, the UAV returns to the point *T*.

The mission characterized above has been used in two phases:

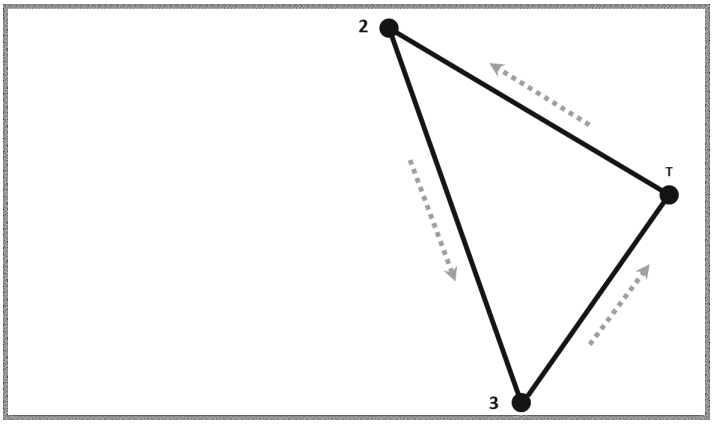


Fig. 5. The three waypoints characterizing the evaluated mission.

1. To determine the best encryption algorithm for memory and energy consumption. In this case, the shared key has been hard-coded within the code. We discuss the details and the motivations behind this phase in Sect. 4.1;
2. To evaluate the overall performance of the proposed solution, taking into account both the encryption and key exchange phases. The details are outlined in Sect. 4.1.

Encryption. This section focuses on the first phase of the testing activity. In particular, we focus on analyzing the encryption algorithms that use a shared key hard-coded in the source code. For each evaluated encryption algorithm integrated into the MAVLink protocol, we completed the mission 10 times. Then, using the *pidstat* utility provided by the SITL simulator, for each second, we collect the CPU consumption and memory usage for the process related to our test. Furthermore, we estimate the energy consumption from the GCS for each flight session. Again, we evaluate the collected results and show them through several graphs. Then, we also provide some comparisons based on the achieved results.

Notice that even though the CBC mode is also supported for *Simon* and *Speck* ciphers, it has not been possible to test it due to some limitations related to modifying the payload size.

From Fig. 6 and Fig. 7, we can observe that there are no relevant differences concerning memory and energy consumption among all the considered algorithms. Again, as we can observe from Fig. 8, the *Trivium* algorithm offers the worst performance in terms of CPU usage. Conversely, all the other evaluated encryption algorithms perform significantly better than *ChaCha20*. In particular, *Speck128/192* in CTR mode offers the best performance. Therefore, our solution uses *Speck128/192* as an encryption algorithm.

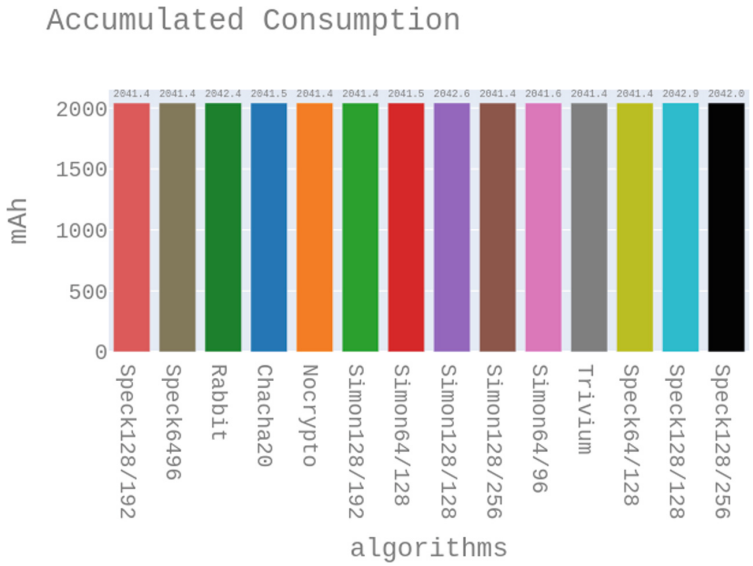


Fig. 6. Encryption algorithms energy consumption.

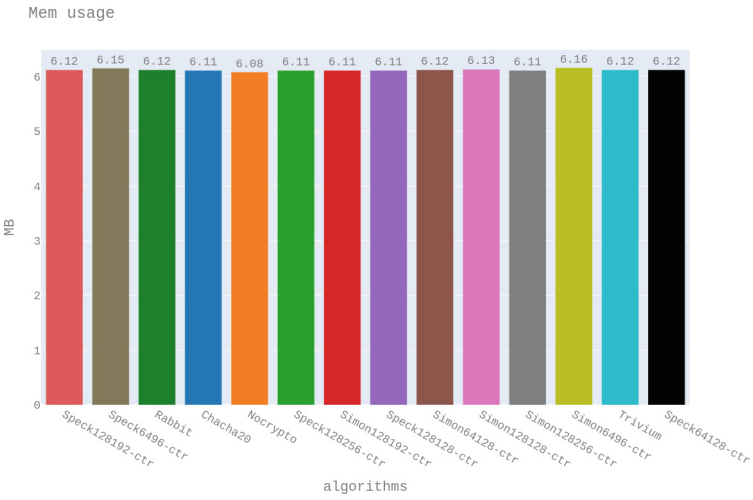


Fig. 7. Encryption algorithms memory consumption.

Key Exchange and Encryption. Once we have determined the best encryption algorithm regarding CPU consumption, i.e., the *Speck128/192*, we performed an evaluation similar to the one outlined in Sect. 4.1. In particular, to analyze the overall performance of our proposal, we repeated 10 times the mission, using the *Speck128/192* algorithm preceded by a key exchange phase.

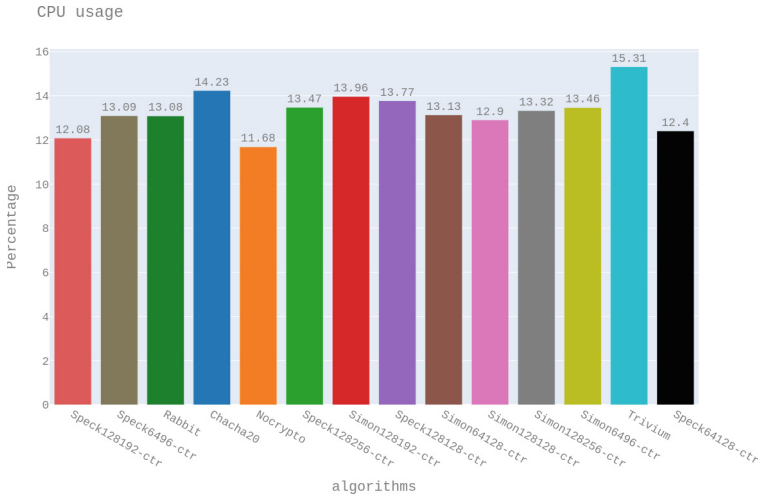


Fig. 8. Encryption algorithms CPU consumption.

The key exchange phase adds only a slight initial overhead with respect to using only the encryption, as it can be observed from Figs. 9 and 10. Therefore, the overall performance of our proposal is not negatively affected by such a phase.

4.2 Correctness

Finally, after choosing the encryption algorithm and applying the key exchange, we intercept and analyze the traffic between UAV and GCS to verify the correct functioning of the proposed solution.

Starting from traffic analysis, we can draw the following considerations:

- *Traffic generated without the proposed solution*: Analyzing the *HOME_POSITION* message shown in Fig. 11 (a), it is possible to locate the waypoint from which the drone has taken off. This information is leaked due to the lack of encryption.
- *Traffic generated using the proposed solution*: Analyzing the *HOME_POSITION* message shown in Fig. 11 (b), it is not possible to obtain information on the waypoint thanks to our solution.

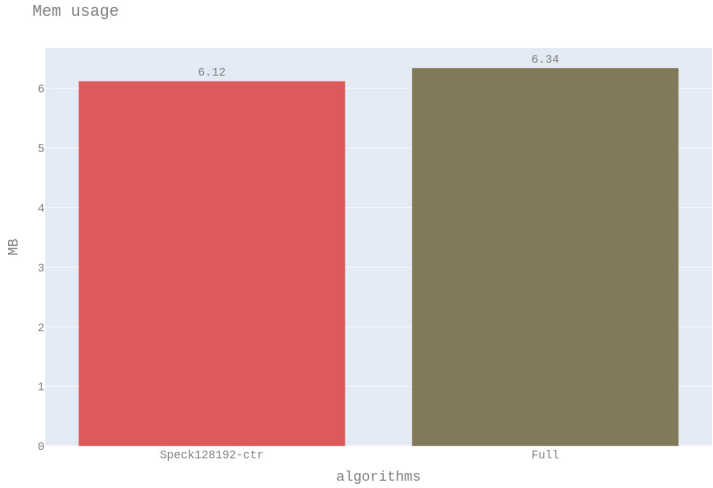


Fig. 9. Memory consumption of the proposed solution.

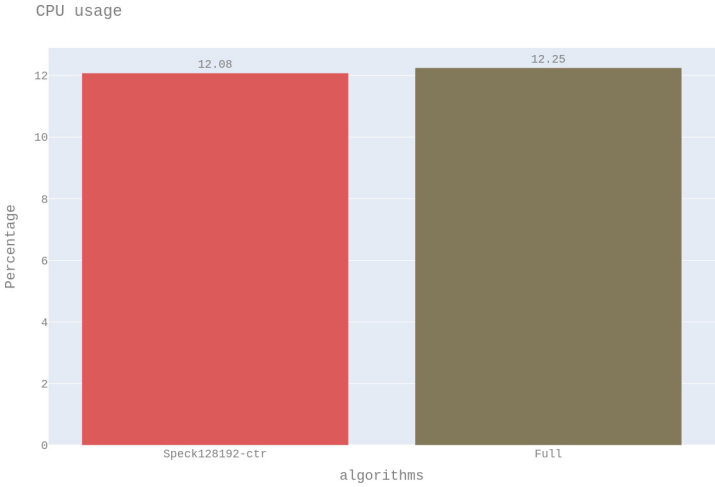
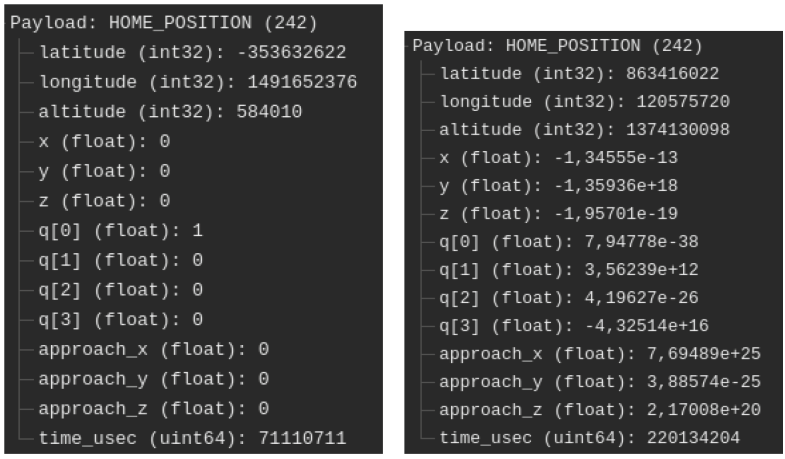


Fig. 10. CPU consumption of the proposed solution.



(a) An example of an individual figure sub-caption.

(b) A slightly shorter sub-caption.

Fig. 11. The HOME_POSITION Message: Before and After Encryption.

5 Conclusions and Future Directions

In recent years, drones, also known as Unmanned Aerial Vehicles (UAVs), have found fields of application in many environments. As discussed in [8], one of the most valuable and advanced fields of application for drones is related to smart cities [17]. However, a drone, also known as *Unmanned Aerial Vehicle* (UAV), is a part of a more complex system denoted as *Unmanned Aerial System* (UAS). On the other hand, drones and all the elements of a UAS system can be affected by several issues related to security and privacy, such as attacks on the flight controller and ground control station, attacks on the data link, etc. Most of such attacks are targeted to the *MAVLink*, which is the main communication protocol used by UAVs. The MAVLink protocol is a *lightweight* messaging protocol for communication with drones and among the components of onboard drones. Since the MAVLink protocol is widely used, it is affected by several security issues and attacks. MAVLink protocol implements a solution to solve authentication problems, including signatures in the MAVLink messages. However, this solution only solves a small portion of problems affecting the MAVLink since the GCSs do not provide implementations of signature mechanisms. Other solutions have been proposed to improve the security of MAVLink, and most of them rely on symmetric cryptography. However, due to the hardware constraints and the limited computational capabilities, such solutions could not be suitable for all devices. Allouch et al. [5] proposed MAVSec, a solution based on cryptography that can solve security problems related to confidentiality and privacy. However, even the MAVSec is affected by some safety and scalability issues that do not make it fully suitable for all modern scenarios where drones are used.

This paper addresses MAVSec issues by improving the communication link security between GCS and UAVs in smart city ecosystems.

As a future research development, we would like to define an intelligent model that allows establishing, dynamically and adaptively, the best possible combination of security algorithms that guarantees the best compromise between communication security and drone lifetime. Furthermore, we would like to test the feasibility of using safety routines implemented directly onboard the drone.

Acknowledgement. This work was partially supported by project *"SEcurity and RIghts in the CyberSpace (SERICS)"* (PE00000014) under the NRRP MUR program funded by the EU - NGEU.

References

1. Artificial intelligence - omdia - informa. https://omdia.tech.informa.com/topic-pages/artificial-intelligence
2. Mavlink developer guide. https://mavlink.io
3. QGC - qgroundcontrol - drone control. https://qgroundcontrol.com/
4. SITL simulator (software in the loop) - dev documentation. https://ardupilot.org/dev/docs/sitl-simulator-software-in-the-loop.html
5. Allouch, A., Cheikhrouhou, O., Koubâa, A., Khalgui, M., Abbes, T.: Mavsec: securing the mavlink protocol for ardupilot/px4 unmanned aerial systems. In: 2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC), pp. 621–628. IEEE (2019)
6. Altawy, R., Youssef, A.M.: Security, privacy, and safety aspects of civilian drones: a survey. ACM Trans. Cyber-Phys. Syst. **1**(2), 1–25 (2016)
7. Biryukov, A., Perrin, L.P.: State of the art in lightweight symmetric cryptography (2017)
8. Dai Nguyen, H.P., Nguyen, D.D.: Drone application in smart cities: the general overview of security vulnerabilities and countermeasures for data communication. Dev. Future Internet Drones (IoD): Insights Trends Road Ahead **332**, 185 (2021)
9. Dey, V., Pudi, V., Chattopadhyay, A., Elovici, Y.: Security vulnerabilities of unmanned aerial vehicles and countermeasures: an experimental study. In: 2018 31st International Conference on VLSI Design and 2018 17th International Conference on Embedded Systems (VLSID), pp. 398–403 (2018)
10. Duguma, D.G., Kim, J., Lee, S., Jho, N.S., Sharma, V., You, I.: A lightweight D2D security protocol with request-forecasting for next-generation mobile networks. Connect. Sci. 1–25 (2021)
11. Koubâa, A., Allouch, A., Alajlan, M., Javed, Y., Belghith, A., Khalgui, M.: Micro air vehicle link (MAVLink) in a nutshell: a survey. IEEE Access **7**, 87658–87680 (2019)
12. Krishna, C.G.L., Murphy, R.R.: A review on cybersecurity vulnerabilities for unmanned aerial vehicles. In: 2017 IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR), pp. 194–199 (2017). https://doi.org/10.1109/SSRR.2017.8088163
13. Kwon, Y.M., Yu, J., Cho, B.M., Eun, Y., Park, K.J.: Empirical analysis of mavlink protocol vulnerability for attacking unmanned aerial vehicles. IEEE Access **6**, 43203–43212 (2018)

14. Madakam, S., Ramaswamy, R.: 100 new smart cities (India's smart vision). In: 2015 5th National Symposium on Information Technology: Towards New Smart World (NSITNSW), pp. 1–6. IEEE (2015)
15. Rodday, N.M., Schmidt, R.D.O., Pras, A.: Exploring security vulnerabilities of unmanned aerial vehicles. In: NOMS 2016–2016 IEEE/IFIP Network Operations and Management Symposium, pp. 993–994 (2016)
16. Xiao, L., Xie, S., Han, D., Liang, W., Guo, J., Chou, W.K.: A lightweight authentication scheme for telecare medical information system. Connect. Sci. 1–17 (2021)
17. Zheng, Z., Zhou, Y., Sun, Y., Wang, Z., Liu, B., Li, K.: Applications of federated learning in smart cities: recent advances, taxonomy, and open challenges. Connect. Sci. 1–28 (2021)