Securing Unmanned Aerial Vehicles by Encrypting MAVLink Protocol

1st Noshin Sabuwala
Electrical Department
Veermata Jijabai Technological Institute
Mumbai, India
nasabuwala_p21@el.vjti.ac.in

2nd Rohin D Daruwala
Electrical Department
Veermata Jijabai Technological Institute
Mumbai, India
rddaruwala@el.vjti.ac.in

***Abstract*—Unmanned Aerial Vehicles (UAVs) or drones and Ground Control Station (GCS) frequently use the lightweight Micro Air Vehicle Link (MAVLink) protocol for communication. It describes a series of communications sent back and forth between a GCS and a UAV. The communication provides data regarding the status of the UAV and orders for control sent by the GCS. However, the MAVLink protocol lacks security and is susceptible to several attacks, which poses serious risks to public safety. There is less research that offer remedies for this issue. To fill the gap, we talk about the security flaws in the MAVLink protocol in this paper and examine three security-integrated algorithms - ChaCha20, Encryption by Navid, and DMAV that researchers have proposed for MAVLink to protect the MAVLink messages that are sent back and forth between UAVs and GCSs. Using a simulated environment called Gazebo, a case study examines the methods used by the autopilot system, Ardupilot (a UAV), and QGroundControl (a GCS) to assess how well they perform in terms of packet transfer speed, memory utilisation, and CPU consumption. The results of the experiments demonstrate that ChaCha20 is more effective and performs better than other encryption algorithms. A resource-constrained drone's battery life and message secrecy can both be preserved by integrating ChaCha20 into MAVLink. This can be done without degrading MAVLink's performance and while using similar memory and CPU.**

***Index Terms*—Unmanned Aerial Vehicle (UAV), Communication Security, GCS, Encryption, MAVLink, Ardupilot, QGroundControl.**

I. Introduction

Unmanned vehicles are independent systems that are simple to programme to carry out tasks with or without human assistance. These devices can be aerial-based (UAV), ground (UGV), or underwater. A GCS, monitoring UAV's status and directing their activities, usually communicates with them wirelessly. UAVs contain specialised software and hardware known as autopilot that regulate the drone's mobility, keep an eye on its condition, and use telemetry communication to interact with GCSs. MAVLink is a popular lightweight message serialisation protocol for UAVs. MAVLink was made available under the LGPL licence by Lorenz Meier in 2009. As a Marshaling library, MAVLink serialises messages describing the system's states and the commands it must carry out as a stream of bytes, which is independent of the platform. The MAVLink protocol is lightweight because it uses binary serialisation, which has less overhead than alternative serialisation methods (e.g., JSON or XML). MAVLink allows for bidirectional communication between the UAV and GCS. Additionally, because MAVLink uses binary serialisation, its messages are typically minimal and may be securely delivered across a variety of wireless channels. Additionally, it provides message integrity and reliability by double-verifying the checksum in the header. This makes MAVLink the most popular protocol for communication between GCS and UAVs [1]. Despite its widespread popularity, MAVLink protocol has flaws and is vulnerable to a number of attacks, such as message forgery, spoofing, and denial of service attacks (DoS) [2]. The protocol doesn't use any security mechanisms or encryption algorithms, which is the main cause of these flaws. As a result, the GCS and UAV communicate via an unencrypted channel, making them vulnerable to many forms of attacks. A few articles have examined potential security fixes for the MAVLink protocol in the literature. To safeguard the connection between drones and GCSs, we propose strengthening the MAVLink's security in this article. This makes it possible to lessen hostile attacks. The suggested approach entails the implementation of three encryption algorithms proposed by researchers, namely, ChaCha20, encryption by Navid [3], and DMAV [4] to guarantee the confidentiality of the messages transmitted back and forth between GCS and UAV. We also assess how well they perform in terms of CPU consumption, memory usage, and packet transfer rate. The following are this paper's main contributions: We start by identifying the security risks to the MAVLink protocol. Next, we suggest three cryptographic techniques included MAVLink version for confidentiality of the information exchange between GCS and UAV. Then, we formulate a case study to implement the security measures in software in the loop simulator using MAVLink to show the feasibility of the techniques. We then compare the effectiveness of all the three techniques via performance assessment of our suggestion. The rest of this paper is arranged as follows: An overview of MAVLink protocol with its header is demonstrated in Section II. Section III presents the literature survey. Section IV discusses the security threats and vulnerabilities of MAVLink. Background of proposed cryptographic solutions to secure MAVLinkis described in Section V. Section VI showcases the case study

978-1-6654-9291-1/22/$31.00 © 2022 IEEE

involving detailed simulated experimental results. Concluding remarks are provided in Section VII.

II. MAVLink Protocol

The open source, simple, and header only MAVLink protocol is mainly utilised for two-way information exchange linking UAVs and GCSs. Lorenz Meier made available MAVLink 1.0 in early 2009 and MAVLink 2.0 in early 2017 under an LGPL licence [1]. Two different types of MAVLink communications are sent: (i) messages from UAV to GCS i.e. state information messages, and (ii) messages from GCS to UAV i.e. control and command messages. The MAVLink protocol is made to be a lightweight protocol because it is used for instantaneous communication. Fig 1 presents the MAVLink 2.0 packet structure. Each data payload of an MAVLink communication has a header applied to it. The checksum is meant to confirm the message's integrity, which shouldn't be changed during transmission. MAVLink is a protocol with variable size. A MAVLink message has a maximum packet size of 297 bytes, and a minimum packet size of 11 bytes with the whole payload and signature. The payload length varies depending on the parameters transmitted during communication. The signature field enables message authentication and confirms that the message comes from a reliable source. The ID field on the packet serves to identify the different MAVLink message types, and the relevant data is contained in the payload. In the MAVLink protocol, various state and control messages are defined. The MAVLink protocol outlines the foundation for composition of message and the application-layer method for serialising messages. A object state or data structure is serialised when it is changed into a format that will be distributed or saved for later use. These messages are transmitted to the lower layers after serialisation to be sent over the network. TCP/IP, Wi-Fi, or low-bandwidth serial telemetry networks can all be used to transmit the MAVLink protocol over frequencies like 433, 868, and 915 MHz [1]. The second choice is to typically transmit MAVLink communications across IP networks using a Ethernet network interface or Wi-Fi. Depending on the setup of the application, both TCP and UDP links between the UAV and GCS are accepted by the MAVLink autopilot in the transport layer. For the UDP datagram protocol, a server-client connection is not required [5]. As a result, it is unreliable for sending messages. The benefit is that it offers a quick, lightweight alternative for loss-tolerant, real-time streaming communication. TCP is dependable for communication. The user must decide whether to utilise the TCP or UPD protocol based on the requirements. Binary serialised messages are used for information exchange between UAV and the GCS. The serialisation and deserialization of the message occur at both the ends since the communication is bidirectional. MAVLink serialisation is lighter and requires few transmission messages and is noticeably lighter than other serialisation techniques.

III. Literature Survey

UAVs are frequently the target of threats aimed at unmanned aerial vehicles. The target might be any of the three parts, the UAV, GCS, or the communication channel connecting them [6]. The focus of this study is on the attacks on communication links. The security of unmanned systems has become extremely important because of their rapid expansion. There has been a lot of work done in this topic and many researchers have contributed. The contributions primarily fall under two categories: Hardware and software. Numerous hardware and integrated solutions have been built for MAVLink's security. In a study proposed by [7], the authors suggest a hardware-implemented AES protocol to encrypt communication between the GCS and UAV. The study's primary concerns are authentication and secrecy. However, due to the additional hardware weight, the suggested hardware solution reduces system effectiveness, CPU usage, and energy consumption. On the other hand, the authors adopt Caesar cypher for message authentication and encryption of MAVLink data between UAV and GCS as part of their software solution contributions for the MAVLink protocol [8]. The absence of the study's results is one of the study's limitations. The fact that the secret Key is sent in plain text is another downside of their job. Additionally, the study has not been empirically evaluated. In [9], to guarantee data integrity, another author suggested using cryptographic encryption for authentication. These two studies, however, are merely proposals. Another study uses the encryption RC5 to make sure that communications are safe, however there are no specifics or evidence that the experiment was valid [10]. In [11], to protect MAVLink communication, the authors presented a MAVSec method. Four encryption algorithms—RC4, ChaCha20, AES-CBC, and AES-CTR were compared. According to their findings, ChaCha20 appears to be performing better than its competitors. However, they only encrypt the payload messages in their approach. The remainder of the packet is identical. In [3], the authors added an additional layer of security, which protects the entire packet. Additional research has been done to safeguard the MAVLink communication mechanism. However, most of the research is still in the early stages of development or are merely suggestions. In [8], [12], the MAVLink messages sent between the Micro Aerial Vehicles (MAV) and the GCS were encrypted by the creators using the Caesar cypher cryptography . They demonstrated that during the establishment phase, there was a distinct secret key transmission from the GCS to the drone. The key might be easily found by someone who is listening to the conversation, allowing them to compromise the entire system. The techniques involving Caesar encryption utilised in these publications are additionally acknowledged to be unreliable and susceptible to cryptanalysis. In this study, we integrate the encryption techniques into the UAV's source code to enable secure MAVLink protocol communication and performance assessment.

ACRONYMS	STX	LEN	INC FLAGS	CMP FLAGS	SEQ	SYS ID	COMP ID	MSG ID	PAYLOAD	CKA	CKB	SIGNATURE
RANGE	0xFD	1 byte	1 byte	1 byte	1 byte	1 byte	1 byte	3 bytes	0-255 bytes	1 byte	1 byte	13 bytes
DESCRIPTION	Start	Payload Length	Incompatibility flags	Compatibility flags	Packet sequence	Sender ID	Component ID	Message type	Actual data	Checksum with seed value A	Checksum with seed value B	Message authentication

Fig. 1. MAVLink 2.0 Packet structure with short description of fields.

IV. Security Threats

The communication protocol aids in the wireless channel connectivity between the GCS and UAV. Because the MAVLink protocol lacks established security protocols, this communication is unsafe in the context of the MAVLink protocol. The sole security check performed is to determine whether the packet is genuine and originates from a legitimate source. Confidentiality, Integrity, and Availability are some of the other security needs that are not natively supported. Thus, a variety of attacks, including modification (which violates the integrity), interception (which violates the confidentiality), and interruption, could be made against the MAVLink protocol (Attacks against the system's availability). Eavesdropping on channels enables interception. Unauthorized users are reading the message's content. The MAVLink doesn't encrypt the messages and does not have a covert security mechanism. This indicates that the information exchange between the UAV and GCS is insecure and easily vulnerable. Any hacker or assailant with the right transmission tool can listen in on the conversation and contact the UAV. The intrusive party can take advantage of this weakness to achieve their objectives, including hijacking the UAV entirely or inserting erroneous directives into an ongoing mission. To reduce this risk and ensure the confidentiality and integrity of the sent messages, encryption and authentication should be employed on the link. Modification is the process of changing the message. Because the MAVLink protocol doesn't guarantee integrity, a hacker may be able to essentially take the UAV away from its GCS. If there are no safeguards in place to protect information integrity, malicious network assaults may lead to information change and eventual invalidation. A message to or from a service is stopped when there is an interruption. A hostile party prevents the drone from receiving MAVLink control signals from the ground. The aircraft will enter a lost link condition because of the drone and GCS's communication being prevented. Strong authentication systems can be implemented to reduce the danger of downtime. Here, we're primarily concerned with attacking the UAV's communication links and preventing UAV hijacking. To capture the packets, we first used a Man-in-the-Middle attack. Several attacks are carried out using the intercepted MAVLink packets like making use of the MAVLink packet for active and passive attacks. Following that, we encrypt the communication using three different methods before comparing the algorithms and presenting our findings.

V. Securing MAVLink Protocol

This section provides an improved MAVLink version using encryption techniques to address the confidentiality flaws in the MAVLink. MAVLink communications have a header with a non-encryptable MAVLink ID. A heartbeat message sent from UAV to GCS ensures that the UAV is prepared and functional before transferring any parameter through the payload. To make sure data is successfully obtained by the GCS, the checksum is computed after encrypting the payload using the key generated during the authentication process. The encrypted payload is sent by the UAV in a message to the ground station. Following receipt of the communication, the GCS first verifies the checksum before decrypting the contents. On both the UAV and the GCS, the encryption algorithms ChaCha20, Encryption by Navid, and DMAV are developed. To obtain encrypted data, the payload is provided as an input to the encryption process. The encrypted data is decoded at the receiver's end.

A. Background Of Algorithms

1) ChaCha20: Four constants— 0x61707865, 0x3320646e, 0x79622d32, and 0x6b206574, a 256-bit key(Key), a 32-bit initial counter(Count), and a 96-bit nonce(Nonce) are used by the algorithm to create an initial matrix (I). [13]. Each entry of the 4x4 matrix used to represent the 512-bits of I matrix is regarded as an unsigned integer of 32-bit. Additionally, I is organised in little-endian form.

The ChaCha20 cypher suite is displayed by algorithm 1. The values of I are first used to initialise an operation matrix (O). The internal state O is subjected to 10 double-rounds of the ChaCha20 algorithm, with each double-round consisting of eight quarter-round function applications. When every state has been updated in O, the rounds are considered complete. The quarter round (QR) operation's output is saved in the location as the input occupied state. By combining the I and O, the keystream is finally acquired. To reach the following I, the counter value is additionally increased by one for processing each block of plaintext [14]. A 32-bit word carry-less addition is what the QR algorithm designates as an addition. The result is an identically long encrypted message. The process for decryption is the same. The key is expanded using the ChaCha20 block function to create a keystream. To reveal the plaintext, the ciphertext is then XORed with the keystream [11].

2) Encryption by Navid: Authors [3] presented a fix for the MAVLink protocol's runtime security via the algorithm 2. It replaces the character's ASCII by applying unique mapping

Algorithm 1 ChaCha20 Algorithm

Require: $Key \in (0,1)^{256}, Nonce \in (0,1)^{96}, Count \in (0,1)^{32}, PlainText \in (0,1)^{*}$
Ensure: $CipherText = ChaCha20(Key, Nonce, Count, PlainText)$

$I \leftarrow Init(Key, Nonce, Count)$
for $a \leftarrow 1to([PlainText/512])$ **do**
    O ← I
    for `b ← 1 to 10)` **do**
        O[0,4,8,12] ← QR(O[0,4,8,12])
        O[1,5,9,13] ← QR(O[1,5,9,13])
        O[2,6,10,14] ← QR(O[2,6,10,14])
        O[3,7,11,15 ← QR(O[3,7,11,15]
        O[0,5,10,15] ← QR(O[0,5,10,15])
        O[1,6,11,12] ← QR(O[1,6,11,12])
        O[2,7,8,13] ← QR(O[2,7,8,13])
        O[3,4,9,14] ← QR(O[3,4,9,14])
    end for
    $Sl \leftarrow Serial(O + I)$
    for `c ← 1 to 512` **do**
        $CipherText[512(a-1)+(c-1)] \leftarrow PlainText[512(a-1)+(c-1)] \oplus S[c-1]$
    end for
    $I[12] \leftarrow I[12] + 1$
end for
return $CipherText$

Algorithm 2 Encryption by Navid Algorithm

Require: $MAVLink\_Packets$
Ensure: $Encrypted\_MAVLink\_Packets$

Initiate Drone_Object
$D_0 \leftarrow PacketData$
$Interval \leftarrow IV$
if $PacketCount == Interval$ **then**
    Reset(count)
    Swap(KeyList)
end if
Pick RandomIndex
FetchKey ← Random(I)
HexString ← Convert(FetchKey)
for `num ← EveryChar` **do**
    if $Char == Alpha$ **then**
        CaesarCipherEncrypt(Char, Key)
    else
        if $Char == Number$ **then**
            CaesarCipherEncrypt(Char, Key + 2)
        end if
    end if
    $CustomMapping(Char) \rightarrow EncryptedString$
    return Encrypted_MAVLink_Packets
end for

to the MAVLink message first. It outputs an ASCII string made up of arbitrary characters. Only using the same mapping method can this be reversed. Their strategy provides the idea of lists from the perspectives of the GCS and UAV. The list stores serial number and key. The key is the Caesar cypher Key for message encryption, while the serial number is a representational number to match the Key. For instance, if the UAV side selects the Serial Number 3's Key, the GCS should decode the message using the Serial Number 3's Key as well. The serial number is shared instead of sharing the Key during communication. As a result, on packet interception, the intruder will only have the serial number, which has no value in decrypting the cypher. The serial number, which requires four bytes, is inserted to the beginning of the packet. Following the serial number's addition, the Caesar cypher encryption is used to protect the selected serial number in the list based on the Key. Following conversion, it is delivered to the GCS as bytes. The first four bytes of each packet, which carry the serial number, are taken as the UAV receives the data. At UAV, the list is checked against the serial number. The Key is found using the serial number, and the message is then decrypted using a reverse Caesar cypher. The character string that is obtained after decrypting the data is again decrypted using the mapping reverting procedure.

The serial numbers are generated at random from the table for each request while transmitting UAV and GCS data. It lets you to use a separate key to encrypt each packet and change the key for each and every request. As a result, the subsequent MAVLink packet won't have the same serial number and fresh key that was used to encrypt the previous one. The keys and the serial numbers are shuffled at random in the list at both ends in parallel after the first 100 requests are fulfilled, making the lists identical on both sides. Communication is impossible if the list on one side is not updated after each iteration and given to the other as the UAV/GCS will be regarded as unauthorized.

3) DMAV: Authors [4] presented a novel technique for creating a safe connection between GCS and drones that use the binary bits based dynamic DNA Encoding alongwith the lightweight GIFT technique. The data is initially encrypted using GIFT [15], which is a simple encryption technique and is made more secure by using binary bits based dynamic DNA Encoding. The operations that make up each round in the DMAV encryption algorithm are as follows:

1) Sbox of DMAV Block Cipher: The n-bit block (n = 64) is split into 4 bits and becomes the input value of the Sbox(4-bit). The Sbox of DMAV block cipher is presented in Table 3 in [16].
2) A (adenine), C (cytosine), G (guanine), and T (thymine) are the four nucleic acid bases that make up a DNA sequence. T and A are complementary, and C and G are complementary. As in binary system, 1 and 0 are complimentary, 11 and 00, and 10 and 01 are also complementary. DNA sequences perform addition and subtraction in the group of integers modulo 2 (Z_2) similar to the conventional addition and subtraction. For instance, 01-11=10 and 11+10=01 [17].
3) Permutation of DMAV Block Cipher: In the permuta-

tion, DMAV-64 replaces the $P_{64}(i^{th})$ bit of block B with the i^{th} bit of block B. Details on the permutation of DMAV-64 are shown in Table 4 in [16].

4) Add Round Key of DMAV Block Cipher: In the DMAV-64 block cipher, k_0 and k_1 (32-bit total) are selected from the key ($K = k_7, ..., k_0$). k_0 and k_1 are used as U and V of the round key as follows, $RK = U||V = u_{15}...u_0||v_{15}...v_0 (U = k_1, V = k_0)$. The round key is exclusive-ored with the block B, where U is XORed to bl_{4i+1} and V is XORed to bl_{4i}.
5) Constant XOR of DMAV Block Cipher: Round constants C given in Table 5 are used in DMAV-64 ciphers. Single bit and round constants (C = c5c4c3c2c1c0) are XORed to block B as: $bl_{n-1} \leftarrow bl_{n-1} \oplus 1, bl_{23} \leftarrow bl_{23} \oplus c_5, bl_{19} \leftarrow bl_{19} \oplus c_4, bl_{15} \leftarrow bl_{15} \oplus c_3, bl_{11} \leftarrow bl_{11} \oplus c_2, bl_7 \leftarrow bl_7 \oplus c_1, bl_3 \leftarrow bl_3 \oplus c_0$.
6) Keyschedule of DMAV Block Cipher: In DMAV-64 block ciphers, the Keyschedule updates key ($K = k_7, ..., k_0$) and extracts the round key from the updated key K. The notation ($>>> i$) denotes a right rotation operation (i-bit). $k_7||k_6||...||k_1||k_0 \leftarrow k_1 >>> 2||k_0 >>> 12||...||k_3||k_2$

Algorithm 3 DMAV Algorithm

Require: $MAVLinkPayload\&keybin32bit$
Ensure: $EncryptedMAVLinkPayload$

```
for Each Block (16bit) do
    Hex_Conversion to 4-bit
    Change Key bit applying Logistic_Map
    for Each Iteration do
        Cells_Sub
        Bits_Permutation
        DNA_Permutation_By_ChaoticMap
        Round_Key_Addition
        Constant_Addition
        Key_Updation
    end for
end for
```

VI. Case Study

This section gives a thorough analysis of the MAVLink protocol's performance while using the encryption algorithms, both in terms of MAVLink packet transmission speed and resource utilisation, like memory consumption rate and CPU processing time. We explain which algorithm is preferable to utilise considering the specified performance criteria after analysing the results.

A. Integration & Simulation Environment Settings

The Software-In-The-Loop (SITL), Ardupilot, which utilises similar MAVLink communication protocol and similar autopilot as the real drone, is employed with a simulated UAV as the experimental testbed. The application of a virtual drone generalises directly to that of a real drone. SITL simulator allows us to fly a plane, a helicopter, or a rover without any gear. To incorporate encryption techniques into the communication stream sent between the GCS and drone's autopilot, we have assembled the source code of Ardupilot. In addition, we employed the QGroundControl ground station, a C++-based open-source GCS programme created by Lorenz Meier. We also implemented the encryption techniques into the QGroundControl so that it can decrypt the cypher stream received and retrieve the actual MAVLink message, enabling safe communication between the Ardupilot and the QGroundControl. A free GCS programme called MAVproxy connects the GCS to the fictitious UAV. Gazebo has been used by us for simulations. We utilised the UDP protocol and port 14551 to connect to the SITL.

B. Results

We compare our method to the original, insecure MAVLink protocol for benchmarking. The experiment's findings indicate that even if the MAVLink packet is intercepted, it is impossible to decrypt the communicated information. Table I shows the algorithm's parameters used in this experiment. We compare the CPU consumption, memory utilisation, and the number of packets transferred by the secure MAVLink protocol utilising cryptography with those of the unsecure original MAVLink protocol to better analyse the impact of an encryption.

TABLE I
Algorithms' Parameters

Algorithm	*Key Size*	*Block Size*
ChaCha20	256 bits	Same as Payload Length
Encryption by Navid	1 bit	1 bit
DMAV	128 bits	64 bits

1) Transfer Speed (Packets Count): As anticipated, fewer packets will be transferred because the MAVLink-DMAV requires additional processing time. This can be explained by the fact that the bit permutation that makes up the linear layer of DMAV's implementation in software makes it appear complicated and ineffective. Because of its cypher type, the MAVLink-Encryption by Navid transfers more packets. MAVLink-ChaCha20 transfers more packets than either of the other two. Compared to MAVLink-ChaCha20, unsecure MAVLink protocol delivers more packets, however the difference is typically insignificant. This can be clearly seen in Fig 2.

2) Memory Consumption: According to the graph in Fig 3, the MAVLink-DMAV uses the highest memory resources due to the bit permutation, which necessitates the usage of more registers. Fig 3 shows that there is almost any difference in memory utilisation between the MAVLink-ChaCha20 and the unencrypted MAVLink protocol.

3) CPU Utilization: According to the simulation results shown in Fig 4, MAVLink-DMAV uses the most CPU power, while MAVLink-ChaCha20 uses less. This is due to the fact that CPU friendly ARX (Addition-Rotation-XOR) instructions constitute the foundation of ChaCha20. In contrast, the Sbox and Mixcolumns computations in the DMAV use binary fields

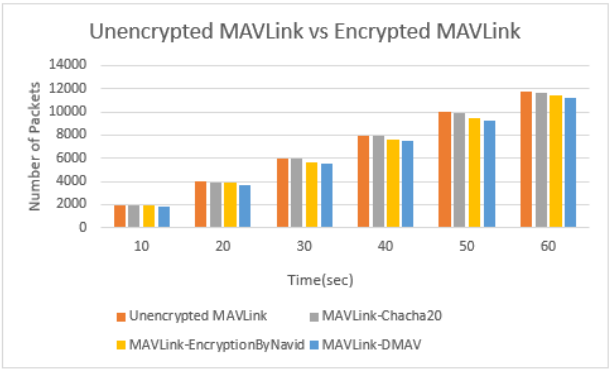

Fig. 2. Packet Transfer Rate Comparison of Unencrypted and Encrypted MAVLink

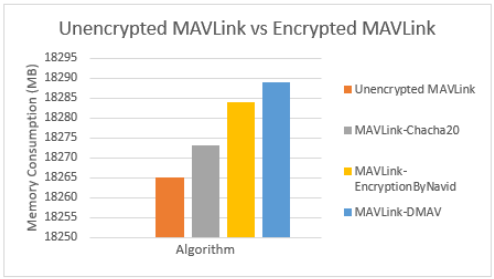

Fig. 3. Memory Consumption Comparison of Unencrypted and Encrypted MAVLink

and are typically put into use as a look-up table to improve efficiency.

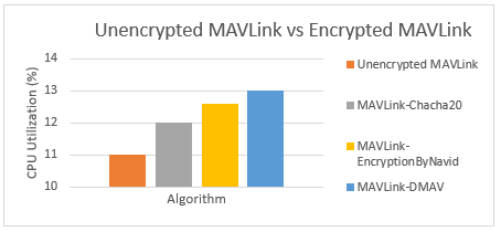

Fig. 4. CPU Utilization Comparison of Unencrypted and Encrypted MAVLink

We can gauge the effectiveness of the encryption scheme by comparing the performance of the secured and unsecure MAVLink protocols. Our set of simulation findings sought to present that ChaCha20 outperforms other encryption techniques with respect to performance and efficiency. Hence, it may be regarded as a standard encryption technique MAVLink's message security and ensure its confidentiality. It does so without impairing its performance, using less CPU power and memory to preserve memory and conserve battery life for drones with limited resources.

VII. Conclusion

We reviewed the MAVLink protocol's vulnerability and security risks in this work, we then suggested various cryptographic ways to address these threats. Ardupilot SITL, a virtual UAV that employs the same autopilot as actual UAVs, for simulation and providing the results. Through the integration of the encryption techniques into the MAVLink source code, the experimental study was made possible. We demonstrated through performance evaluation that ChaCha20 may be used for MAVLink's security since it upholds message secrecy without degrading performance.

References

[1] A. Koubaa, A. Allouch, M. Alajlan, Y. Javed, A. Belghith, and M. Khalgui, "Micro Air Vehicle Link (MAVlink) in a Nutshell: A Survey," *IEEE Access*, vol. 7, pp. 87 658–87 680, 2019.

[2] 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. 43 203–43 212, aug 2018.

[3] N. A. Khan, N. Z. Jhanjhi, S. N. Brohi, A. A. Almazroi, and A. A. Almazroi, "A Secure Communication Protocol for Unmanned Aerial Vehicles," *Computers, Materials & Continua*, vol. 70, no. 1, pp. 601–618, sep 2021.

[4] G. E. Kassim and S. H. Hashem, "DMAV: Enhanced MAV Link Protocol Using Dynamic DNA Coding for Unmanned Aerial Vehicles," *International Journal of Online and Biomedical Engineering (iJOE)*, vol. 18, no. 11, pp. 4–16, aug 2022.

[5] F. T. AL-Dhief, F. T. AL-Dhief, N. Sabri, N. M. A. Latiff, N. N. N. A. Malik, M. A. A. Albader, M. A. Mohammed, R. N. AL-Haddad, Y. D. Salman, M. K. A. Ghani, and O. I. Obaid, "Performance Comparison between TCP and UDP Protocols in Different Simulation Scenarios," *International Journal of Engineering & Technology*, vol. 7, no. 4.36, pp. 172–176, dec 2018.

[6] J. Whelan, A. Almehmadi, J. Braverman, and K. El-Khatib, "Threat Analysis of a Long Range Autonomous Unmanned Aerial System," *2020 International Conference on Computing and Information Technology, ICCIT 2020*, sep 2020.

[7] A. Shoufan, H. Alnoon, and J. Baek, "Secure communication in civil drones," *Communications in Computer and Information Science*, vol. 576, pp. 177–195, 2015.

[8] B. S. Rajatha, C. M. Ananda, and S. Nagaraj, "Authentication of MAV communication using Caesar Cipher cryptography," *2015 International Conference on Smart Technologies and Management for Computing, Communication, Controls, Energy and Materials, ICSTM 2015 - Proceedings*, pp. 58–63, aug 2015.

[9] R. Altawy and A. M. Youssef, "Security, Privacy, and Safety Aspects of Civilian Drones," *ACM Transactions on Cyber-Physical Systems*, vol. 1, no. 2, nov 2016.

[10] Butcher N, Stewart A, and Biaz S, "Securing the mavlink communication protocol for unmanned aircraft systems," 2013.

[11] A. Allouch, O. Cheikhrouhou, A. Koubaa, M. Khalgui, and T. Abbes, "MAVSec: Securing the MAVLink protocol for Ardupilot/PX4 unmanned aerial systems," *2019 15th International Wireless Communications and Mobile Computing Conference, IWCMC 2019*, pp. 621–628, jun 2019.

[12] R. Hamsavahini, S. Varun, and P. VS, "Development of light weight algorithms in a customized communication protocol for micro air vehicles," pp. 73–79, 2016.

[13] Y. Nir and A. Langley, "ChaCha20 and Poly1305 for IETF Protocols. RFC 8439," 2018.

[14] R. Serrano, C. Duran, M. Sarmiento, C. K. Pham, and T. T. Hoang, "ChaCha20-Poly1305 Authenticated Encryption with Additional Data for Transport Layer Security 1.3," *Cryptography 2022, Vol. 6, Page 30*, vol. 6, no. 2, p. 30, jun 2022.

[15] S. Banik, S. K. Pandey, T. Peyrin, Y. Sasaki, S. M. Sim, and Y. Todo, "GIFT: A small present: Towards reaching the limit of lightweight encryption," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 10529 LNCS, pp. 321–345, 2017.

[16] K. Jang, G. Song, H. Kim, H. Kwon, H. Kim, and H. Seo, "Efficient Implementation of PRESENT and GIFT on Quantum Computers," *Applied Sciences 2021, Vol. 11, Page 4776*, vol. 11, no. 11, p. 4776, may 2021.

[17] Q. Zhang, L. Guo, and X. Wei, "Image encryption using DNA addition combining with chaotic maps," *Mathematical and Computer Modelling*, vol. 52, no. 11-12, pp. 2028–2035, dec 2010.