SAMRIDDHI Volume 14, Issue 4, 2022 Print ISSN: 2229-7111 Online ISSN: 2454-5767

A Secure Communication Protocol for Unmanned Aerial Vehicles using IoT Protocols

Jayant kumar Dorave[1], Ritesh Sadiwala[2]

[1]Research Scholar, Department of ECE, R. K. D. F. University, Bhopal, India.

[2]Associate Professor, Department of ECE, R. K. D. F. University, Bhopal, India.

Abstract

IOT_MQTT is an open-source communication system for Unmanned Aerial Vehicles that is lightweight and extensively utilized. It is supported by a number of UAVs and autopilot systems, and it allows bidirectional communication. The findings demonstrate that our approach can encrypt IOT packets while maintaining performance and efficiency. The findings are confirmed in terms of transfer speed, performance, and efficiency when compared to other solutions in the literature such as smart dust and benchmarked against the original IoT-MQTT protocol.

Keywords: Unmanned aerial vehicles; IOT_MQTT protocol; drones security; UAVs communication.

SAMRIDDHI : A Journal of Physical Sciences, Engineering and Technology (2022); DOI: 10.18090/samriddhi.v14i04.13

Introduction

Unmanned Aerial Vehicles (UAVs) have grown in popularity in recent years. These unmanned aerial systems are being deployed worldwide as their military capabilities have been expanded to include commercial applications.[1] An Unmanned Aerial Vehicle (UAV) is a pilotless aircraft with no crew on board.

This work is distributed under a Creative Commons Attribution 4.0 International License, which allows for free use, distribution, and reproduction in any form as long as the original work is properly noted. [2,3] board They can function totally autonomously [4,5] through autopilot, which implies they can accomplish tasks without human intervention, or non-autonomously with the assistance of a remote control or a human pilot acting in runtime from a ground control station. [6] The military[7–9] uses it for warfare or reconnaissance, while citizens[10,11] utilise it for pleasure and amusement. Following this, there are several more growing uses of UAVs, including agricultural,[12,13] environmental protection,[14] search and rescue,[15] traffic monitoring,[16] delivery,[17] aerial mapping,[18] aerial photography,[19,21] and fire detection.[22]

Furthermore, digital behemoths like Facebook[23] and Tesla[24] are leveraging UAVs to offer internet connectivity to remote locations worldwide. UAVs are best suited. For "boring, filthy, and deadly"[25] missions, which denotes a circumstance in which it is exceedingly difficult for a human to access and use When a public communication network is disrupted, UAVs can help. Will give timely catastrophe notifications and aid in the speeding up of rescue or recovery activities They are capable. Carry medical supplies to remote locations UAVs may be used to quickly cover a large area. Region without

Corresponding Author: Jayantkumar Dorave, Research Scholar, Department of ECE, R. K. D. F. University, Bhopal, India., e-mail: jayant.dorve2805@gmail.com

How to cite this article: Dorave, J., Sadiwala, R. (2022). A Secure Communication Protocol for Unmanned Aerial Vehicles using IoT Protocols. *SAMRIDDHI : A Journal of Physical Sciences, Engineering and Technology,* 14(4), 79-88.

Source of support: Nil

Conflict of interest: None

jeopardising worker safety in situations such as poisonous gas invasions, Wildfires, for example, or wild animal monitoring. The most widespread and significant application of UAVs is in military purposes. Countries with modern Unmanned Aerial Systems outperform their competitors significantly due to their higher stealth, smaller size, and real-time capacity in hostile settings and border monitoring.[26] As a result, unmanned aerial vehicles (UAVs) may be deployed efficiently in a variety of settings to combat terrorism while minimizing human casualties. In this sense, real-time monitoring would need a variety of details obtained during UAV flight, such as a telemetric subsystem, payload subsystem, and directives broadcast from the ground station for UAV administration and mission performance.[27]

Unmanned Aerial Vehicles are part of an Unmanned Aerial System (UAS). A UAS consists of three major components: a UAV, a Base Station/Ground Control Station, and the communication system that connects them.[28] Various communication protocols, including as IOT_MQTT,[29] UAVCan,[30] and Uranus Link,[31] are commonly used to communicate between the UAV and the GCS. Among these protocols, the Micro Aerial Vehicle Link (IOT_MQTT) protocol is the most prevalent and frequently used communication protocol, with a large number of UAVs and the Ground Control Station supporting it.[32] Lorenz Meier created this protocol under the GPL licence in 2009.[33] IOT_MQTT enables UAV and GCS communication in both ways. The Ground Control Station transmits instructions and control messages to the UAV, while the UAV transmits telemetry and other status data to the GCS.[34] The IOT_MQTT protocol[35] is also used to connect UAVs over the internet.

Many UAVs and autopilot systems, like Ardupilot[36] and PX4,[37] use the IOT_MQTT protocol. These two open-source systems are the most advanced autopilots capable of controlling any unmanned vehicle, from unmanned aerial planes to unmanned submarines.[36,38]

IOT_MQTT is an open-source and cross-platform lightweight networking technology. There are three versions available: IOT_MQTT 1.0,[33] IOT_MQTT 2.0,[39] and a prototype version IOT_MQTT. For message integrity and authentication, IOT_MQTT 2.0 employs timestamped hash-based message authentication codes (HMAC).

IOT_MQTT is designed as a Marshalling library, which means that the system states' messages and the instructions required to run in a certain binary format are serialised (as bytes streams) irrespective of the platform. Compared to alternative serialisation methods such as XML and JSON, IOT_MQTT's binary serialisation methodology is lightweight and has minimal overhead.

The sIOT_MQTT draught version is a stable version that assures secrecy and integrity by encrypting important details using a symmetric key.[40] To the best of our knowledge, the sIOT_MQTT has not yet been deployed.

Furthermore, because of its Binary Serialization characteristics, IOT_MQTT messages are typically tiny and may be delivered across a variety of wireless networks, including Wi-Fi or even serial telemetric systems with low data rates. In the packet header, a double checksum verification ensures message durability and correctness. Because of these qualities, the IOT_MQTT protocol is the most extensively utilised by its peers for communication between unmanned systems and ground control stations (GCS).

Despite being strong and extensively used, the IOT_MQTT communication protocol lacks a subtle security mechanism, leaving it vulnerable to a variety of attacks, including denial of service (DDoS), eavesdropping, and man-in-the-middle attacks.[41,42] These flaws are obvious since the IOT_MQTT protocol does not encrypt communications in transit. This implies that binary communication between the GCS and the UAV occurs through an unencrypted channel, making it a prime target for various security attacks. As a result, the security of Unmanned Aerial Vehicles is jeopardized.

The fundamental contribution of this study is the inclusion of an extra security layer to the IOT_MQTT communication protocol to safeguard the binary-directed communication between the UAV and GCS. Three algorithms were developed as a result of our research. The additional contributions, as well as the three algorithms, are explained further down.

I created a method for relaunching recorded IOT_MQTT packets for attacks.

II. Created a method to extract useful information from intercepted IOT_MQTT packets.

III. Created a method to encrypt the IOT_MQTT packet to secure communication.

Table 1: Communication link attacks on UAVs

Security objectives	*System objectives*	*Attack methods*
Confidentiality	Ground control station	Virus
		Malware Key loggers
		Trojans
	UAV	Hijacking
	Communication link	Eavesdropping
		Man-in-the-middle
Integrity	Communication link	Packet injection
		Replay attack
		Man-in-the-middle
		Message detection
Availability	GCS	Denial of services (DoS)
	UAV	Fuzzing
	Communication	Jamming
		Flooding
		Buffer overflow Denial of services

Table 2: Security threats/attacks against IOT_MQTT protocol [3,55]

Security requirement	*Threats/attacks*	*Mitigations*
Confidentiality and privacy	Man-in-the-middle	Datalink encryption
	Eavesdropping	
	Identity spoofing Hijacking	
	Unauthorized access	
	Interception	
Integrity	Packet injection	Hash
	Man-in-the-middle	MAC (message authentication code)
	Fabrication	Authentication
	Message deletion	
	Message modification	
	Replay attack	
Availability	Command and control	Authentication
	Jamming	
	Routing attack	
	Denial of service	
	Flooding	
Authenticity	GCS Spoofing	Authentication
	Fabrication	

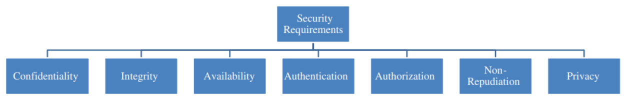


Figure 1: IOT_MQTT security requirements.

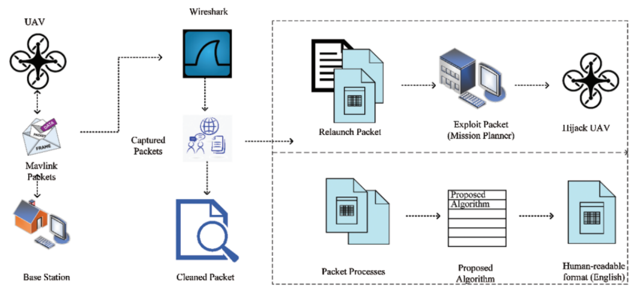
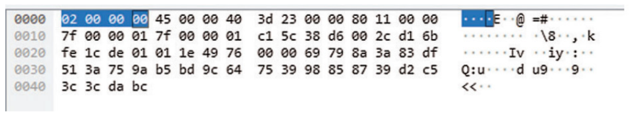


Figure 2: Threat model to exploit the IOT_MQTT vulnerabilities.



```
0000  02 00 00 00 45 00 00 40  3d 23 00 00 80 11 00 00   ····E··@ =#······
0010  7f 00 00 01 7f 00 00 01  c1 5c 38 d6 00 2c d1 6b   ········ ·\8··,·k
0020  fe 1c de 01 01 1e 49 76  00 00 69 79 8a 3a 83 df   ······Iv ··iy·:··
0030  51 3a 75 9a b5 bd 9c 64  75 39 98 85 87 39 d2 c5   Q:u····d u9···9··
0040  3c 3c da bc                                        <<··
```

Figure 3: Captured IOT_MQTT protocol packet structure.

- We carried out and evaluated the experiment in a simulated environment using Ardupilot and Mission planner, both of which use the same Autopilot software as actual UAVs.
- We secured the IOT_MQTT protocol, allowing UAVs and GCSs to communicate while maintaining performance and efficiency.

The remainder of the paper is structured as follows. The Introduction was delivered in Section 1. Section 2 contains an overview of the literature as well as relevant studies. Section 3 provides a full discussion of the IOT_MQTT protocol. Section 4 discusses the IOT_MQTT protocol's security concerns. The IOT_MQTT protocol was abused in Section 5 in terms of security attacks and vulnerabilities. In this section, two algorithms for exploiting IOT_MQTT protocol flaws are proposed. The proposed way to secure the IOT_MQTT protocol is presented in Section 6. This section demonstrates the encryption algorithm and our security strategy. Section 7 shows the experimental findings and our solution's performance and efficiency benchmarking against the original IOT_MQTT protocol. Finally, the paper's conclusion is shown.

Literature Review and Related Work

The danger to UAVs is frequently directed at the Unmanned Aerial System. It might be any of the three components, including the UAV, the Ground Control Station, or the communication link between the two.[43] The emphasis in this study is on communication link assaults, as illustrated in 1st Tab.

Because the use of unmanned systems has increased dramatically in recent years, their security has become critical.

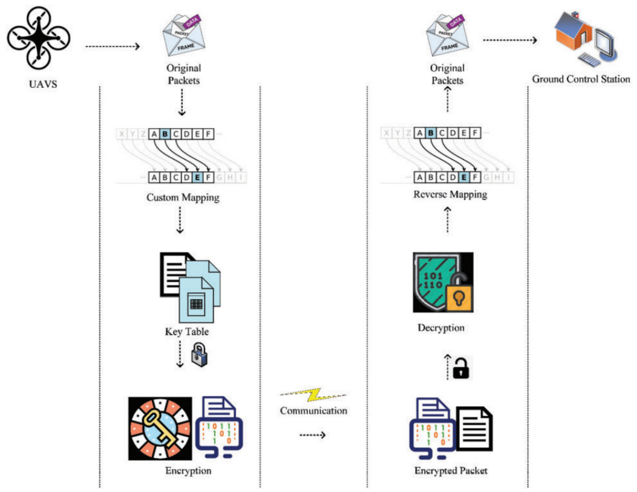


Figure 4: Proposed solution to enhance and secure the IOT_MQTT communication protocol.

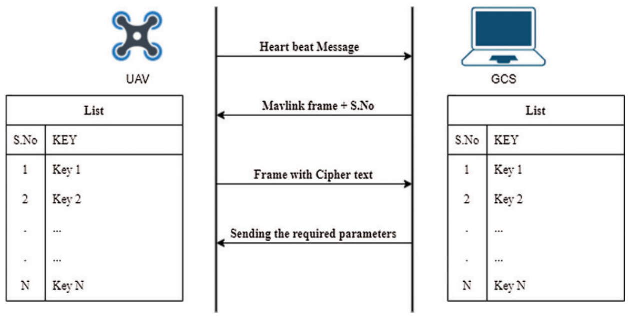


Figure 5: Lists of serial number and secret key for encryption.

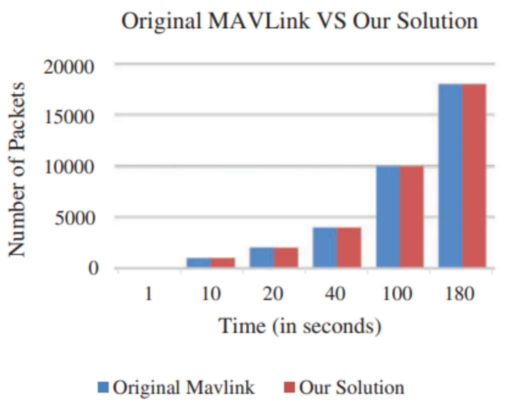


Figure 6: Number of packets transmission (in seconds).

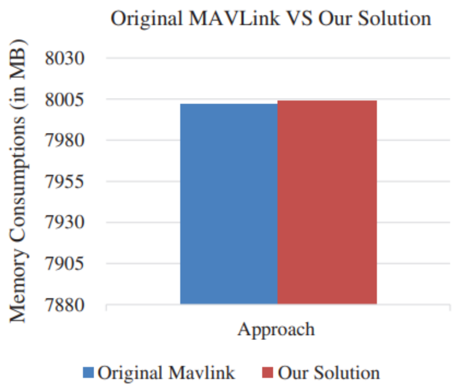


Figure 7: Memory consumptions (in MBs).

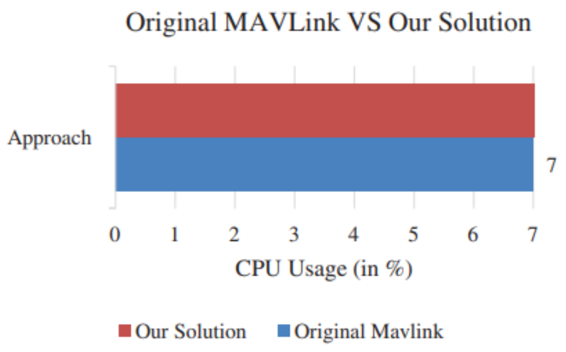


Figure 8: CPU processing (in percentage).

Many researchers have made significant contributions to this topic, and considerable work has been completed. The contributions can be classified into two categories. There are two types of hardware: 1) hardware and 2) software.

Many embedded and hardware security mechanisms have been created to safeguard the IOT_MQTT protocol. The researchers employed extra encrypted communication channels with Raspberry Pi's assistance to address the proposed UAV security issue in the study.[44] In this method, the hardware must interact with GCS to retake control in the event of an assault. The disadvantage of this strategy is the time delay between the GCS and the Raspberry and the increased CPU utilization. Another limitation of the research is that it is only a hypothesis that has yet to be tested on actual UAVs. Our research uses a case study to test the answer, and the findings are presented in Section 7.

In another work [45], the authors use a suggested AES protocol with hardware implementation to encrypt communication between the GCS and UAV. The study's primary focus is on secrecy and authentication. However, due of the added hardware weight, the suggested hardware solution degrades the system's efficiency, CPU, and energy usage.

In contrast, as part of the IOT_MQTT protocol software solutions, the authors recommend employing Caesar cypher cryptography for data encryption and authentication of IOT_MQTT messages between the ground station and the UAV [46]. One disadvantage of this study is that the outcomes were not provided in the study. Another disadvantage of their technique is that the secret Key is sent in plain text. Four successful cryptographic methods were used in a study[47] to mitigate the confidentiality issues in the IOT_MQTT Protocol using efficient symmetrical key encryption algorithms. The four algorithms presented are Rabbit stream cypher, Salsa20 stream cypher, and XXTEA stream cypher. They can easily encrypt IOT_MQTT transmissions while maintaining the privacy of GCS and UAV communications. The study papers [46,48] make advantage of to encrypt IOT_MQTT signals between GCS and UAV for cryptographic data reasons, the Caesar cryptography technique is used. The secret Key, on the other hand, is provided to the UAV in plain text during the setup procedure in this arrangement. It is simple to locate the Key when capturing the packet. As a result, it is quite simple to breach its security. Furthermore, no empirical evaluation of the work is provided. Our research effort puts the answer into action on a case study, and the outcomes are shown below. Another encryption, RC5, is employed in another investigation to ensure that communication is safe, although there are no details or confirmation of the experiment.[49] Our research secured communication between UAVs and GCSs, as well as assessed and confirmed their functionality. clear\sresults. Another study[50] advises adding a digital signature to the data packed using the UAV's Private Key. Another author advocated cryptographic encryption for authentication.[51] Ensure the data's integrity. However, both of these investigations are only proposals.[47] The author performed a vulnerability study and proposed a cryptographic technique to secure the IOT_MQTT protocol without specifying which algorithm will be used. Other authors[37] offered a solution called MAVSec is used to protect IOT_MQTT communication. They compared four encryption techniques, among which AES-CBC, AES-CTR, RC4, and ChaCha20 are examples of encryption algorithms. Based on their results, ChaCha20 appears to outperform others in terms of performance. However, the encryption is only applied to the payload messages in their suggested technique. The remainder of the package is identical. We added an extra layer of security to our technology, which safeguards the whole packet. Several further investigations have been conducted to ensure the security of the IOT_MQTT communication protocol. Nonetheless, most of the research either offered answers or is in the early stages of development.

Overview of the IOT_MQTT Protocol (IOT_MQTT System Architecture)

The IOT_MQTT protocol defines the architecture for message composition and how messages are serialized on the application layer. The serialization process includes turning a data structure or object state into a later stored or disseminated format. Following serialization, these messages are sent to the lower levels, namely the transport and physical layers, for transmission via the network. Because of its lightweight design, it can accept a variety of transport layers and media. Wi-Fi, TCP/IP, or low-bandwidth serial telemetry networks can broadcast the IOT_MQTT protocol across sub-GHz frequencies such as 433, 868, and 915 MHz [52].

The second method is to broadcast IOT_MQTT communications across IP networks using a Wi-Fi or Ethernet network interface. Depending on the application's setup, the IOT_MQTT autopilot accepts both UDP and TCP links between the ground control station and the UAV at the transport layer. The datagram protocol UDP does not need a connection between the client and the server [53]. As a result, it is untrustworthy in terms of message delivery. The benefit is that it offers a quick, light alternative weight for streaming real-time, loss-tolerant communication. TCP, unlike UDP, is a connection-oriented protocol, which means it contains a method for recognizing that the request has been sent [54]. This suggests that TCP is a dependable communication protocol. The user must decide whether to utilize the UPD or TCP protocol based on their needs.

Binary serialized messages are used to communicate between the UAV and the Ground Station. Because the communication is bidirectional, the message is serialized and deserialized at both the sender and receiver ends. IOT_MQTT serialization employs fewer transmission messages and is substantially lighter than other serialization systems. There are two versions of the IOT_MQTT protocol available: IOT_MQTT 1.0 and IOT_MQTT 2.0 [52]. IOT_MQTT is also available in sIOT_MQTT form. To the best of our knowledge, the sIOT_MQTT has not yet been deployed.

Security Issues of IOT_MQTT Protocol

Unmanned Systems research and development is still in its early stages, and much work is being done in this field. Simultaneously, hackers and attackers see it as a chance to exploit new vulnerabilities and breach the security of these systems for a variety of reasons. Many researchers have contributed to Unmanned Systems security at various levels to solve security difficulties and challenges. One of the challenges with such solutions is that they are still in the early phases of execution and are either suggested work or have not yet been implemented. We need to understand the security difficulties before we can provide a solution to exploit the IOT_MQTT protocol's weaknesses. The security problems of the IOT_MQTT protocol are classified into two categories: 1) Security Requirements and 2) Security Threats/Attacks. This will aid practitioners and researchers in developing security frameworks and threat models for Unmanned Aerial Vehicles in the future.

Security Requirements

Overall, substantial study has been done on the security of unmanned aerial systems, but less work has been done on communication level security, specifically on the IOT_MQTT protocol. The medical slogan "prevention is better than cure" is well suited for the security requirements. It is critical to understand the security needs and avoid these undesirable scenarios in order to avoid security risks and assaults. To protect the connection between the UAV and GCS and avoid attacks, the IOT_MQTT's security needs are summarised as confidentiality, integrity, availability, authentication, non-repudiation, authorization, and privacy.[3] The IOT_MQTT security requirements are depicted in Figure 1.

Security Threats

The UAV and GCS communicate over a wireless channel with the assistance of the communication protocol. This communication is susceptible in the case of the IOT_MQTT protocol because the IOT_MQTT protocol lacks basic security protocols. The only security check is that it determines whether the packet is real and originated from an authentic source. The other security criteria, such as secrecy, are not natively supported. The IOT_MQTT lacks a sophisticated security mechanism and does not encrypt communications. That implies the connection between the UAV and the Ground Control Station is insecure and readily hacked. Any hacker or attacker equipped with a suitable transmitter device can intercept the communication and connect with the UAV. The attacker can use this vulnerability for their intended purpose, such as injecting bogus orders into a current operation or totally hijacking the UAV. These assaults are further categorized based on their outcome as follows. Table. 2 shows the categorization.

Based on the aforementioned security issues, we describe our threat model and attack the IOT_MQTT communication protocol's flaws. The threat model is divided into two phases. 1) to take advantage of the IOT_MQTT packet and use it in active attacks 2) Take advantage of the IOT_MQTT packets and subsequently use them for passive attacks. Two methods are created for this goal and are presented in the next section. The threat model is depicted in Figure. 2 below. As mentioned in Table 1, our goal here is to protect the UAV from communication link assaults and to hijack the UAV. In our experiment, we used our suggested technique to attack the IOT_MQTT vulnerabilities. To capture the packets, we first used a Man-in-the-middle assault. After capturing the packets, Algorithm 1 relaunches the recorded IOT_MQTT packets for a replay attack. This may be used for two things:1) If the mission is still running, restart the packets for a replay or eavesdropping assault. If the goal is to inject fake data, a false injection attack can also be carried out by introducing bogus data into the intercepted packets. Based on Algorithm 1, our experiment hijacked the UAV and gained complete control of it. Similarly, Algorithm 2 is designed to comprehend the collected packets of communication between the UAV and GCS. It may be used to initiate passive assaults.

Exploiting the IOT_MQTT Protocol

The experiment is carried out in a simulated environment with the help of ArduPilot Software in the Loop (SITL) and a simulated UAV. The Adrupilot SITL employs the same autopilot as a real UAV. It can control a plan or a land rover without utilizing any hardware and can reproduce the actual UAV in a simulated environment. Mission Planner is chosen for the Ground Control Station, therefore when we open Mission Planner at this time, it immediately connects to the IOT_MQTT protocol is used by UAVs. If it does not connect automatically, click the connect button in the top right corner of the Mission Planner programme to connect manually. After successful execution, the Unmanned Aerial Vehicle may be viewed on the Mission Planner map as swell. Here we may design a new mission, load an existing mission, and execute other mission-related actions.

Capturing the IOT_MQTT Packets

Intruders use specific transmitters to intercept IOT_MQTT packets. Because we are the packets are caught on the local Wi-Fi network while simulating using the packet capturing programme. Wireshark. In Wireshark, we filtered the port numbers to get only the one we wanted. IOT_MQTT. A competent individual may easily recognize the IOT_MQTT packet structure. Once the packet structure has been recognized, it is simple to determine the port from which the packets are being sent. Coming in and then filtering only that port to obtain IOT_MQTT packet metadata.

The detailed structure of the IOT_MQTT packet is seen in Figure 3 above, which was collected using Wireshark. The data shown here is in binary and hexadecimal formats. The packet is stored in a text file that has been properly formatted. Algorithm 1 is then designed to extract relevant information from this packet and utilize it for an attack to gain unauthorized access to the UAV.

```
Algorithm 1: Launching the captured packet for attacks
Input: Captured_Packet
Output: Hijack_Drone
Procedure (Exploiting_Mavlink)
$D_0 \leftarrow$ Read (Caputred_Packets)
$D_1 \leftarrow$ Read_Bytes ( $D_0$ )
Socket $\leftarrow$ Define (Port $_{num}$ )
While ( $i \leftarrow$ Read ( $D_1$ )
Packet $\leftarrow$ Get ( Length, add, port )
Send ( Packet ) $\rightarrow$ Port ( Socket )
G CS ( Mission_Planner ) $\leftarrow$ Received (Packets)
End Procedure
```

Java code is used to implement Algorithm 1. When the code is executed, it reads the contents from the IOT_MQTT packet collected with Wireshark and saves it to a text file. It reads the buffers and then begins transferring data to a specified port number (14450 in our case). It continues to send data as long as it is reading data from the packet. Because we are utilizing a mission planner, the data was received by the Mission Planner via the given port number. All of the UAV information from which the packet was obtained, such as mission data and GPS position, is now visible. If it's During an ongoing operation, the UAV can entirely hijack from Mission Planner because it now has complete control. Furthermore, any other intentional assault, such as eavesdropping or GPS tracking, can be undertaken of the unmanned aerial vehicle. Spoofing, or the insertion of false mission data.

Converting the Packet into a Human-Readable Form

The following step is to take advantage of the IOT_MQTT protocol's flaw, which is that it is not secure and the data transferred is not encrypted. We created Algorithm 2 for this purpose, which translates the information acquired by Wireshark to a human-readable format in plain text and extracts the useful information from it. Java Coding is also used to implement this approach.

```
1    Algorithm 2: Getting secret information in plain text
     Input: Captured_Packet
     Output: Hijack_Drone
     Procedure (Exploiting_Mavlink)
1    Create Drone Object
2    DefineMavlinkMessageHandler
3    $D_0 \leftarrow read(Datablock\_Packets)$
4    While (Connection! = 0)
5    Packet ← Get ( Length, addr, port )
6    Response (Mavlink_Library) ← read (Mavlink_Messages)
7    For(i ← Read (Packet))
8    $D_1 \leftarrow Parse(Packet) + 0x00ff$
9    $D_1 \leftarrow Parse\ (Hex\_Char)$
10   $DroneObject \leftarrow Ready\ (D_0, D_1)$
```

In this approach, a connection is formed first, and then a IOT_MQTT message handler is defined, which verifies the IOT_MQTT messages and determines which sort of IOT_MQTT message is being handled. The connection is then tested to see if it is linked to the drone item. As long as the connection is established, it will first read and store the data block. Then there's our library, which recognizes the IOT_MQTT message. The data is then processed, and the hex value 0x00ff is assigned.

The actual hex value is received after a check to see what sort of IOT_MQTT packet it is. Then

The data is parsed using the derived hex characters. An object is created for the IOT_MQTT packet.

Determines whether or not the packet is null (real packet). If the packet is genuine, it will unpack the IOT_MQTT packet. And converts the data to human-readable form all characteristics, such as may be retrieved from here. Sensor data, roll and pitch, radio frequency, GPS, and so on.

Proposed Solution

This part presents a solution for the IOT_MQTT protocol's runtime security for an ongoing mission between a UAV and a GCS. Our strategy is based on cryptography. Mechanism, as well as our mapping approach, the original IOT_MQTT protocol is enhanced with a security layer. Figure 4 depicts the total suggested model. We encrypt the information in the IOT_MQTT packet and ensure that if the packet is captured using Wireshark or any other transmitting device, 1) it cannot be restarted to take control of the UAV, i.e., the captured packet is worthless to the intruder. 2) Encrypt the packets so that even if an intruder intercepts the packet, he or she will not be able to extract useful information from it. For our answer, we use a case study. As previously indicated, the experiment is carried out in a virtual setting. We assume that the current mission lasts a minimum of ten minutes and a maximum of three hours, which is typically the case for civilian UAV applications. The UAV and GCS are linked.

First, we use custom mapping to substitute the character's ASCII in the IOT_MQTT packet. The data is made up of bytes in binary and hexadecimal formats. It will return an ASCII string made up of random characters. This can only be reversed using the same mapping process. Our As seen in Figure 5, the technique introduces the idea of lists on both the UAV and GCS sides. The three are two columns in the list: a serial number and a key. A serial number is a number that represents something. The list to match the Key, whereas the key column includes the Caesar cipher's real Key the channel over which the message will be encrypted. For example, if the Key against serial number 2 is selected from the UAV side, it signifies that the GCS should decrypt the message using the same serial number 2. Instead of sharing the Key, the serial number is sent in communication. As a result, even if the packet is intercepted, the intruder will only have the serial number, which is useless for decrypting the encryption. The serial number is appended to the beginning of the transmission, taking up four bytes. Once the serial number is inserted, the Caesar cypher encryption is performed to the packet depending on the Key and the serial number in the list that was picked. It is subsequently translated to bytes and transmitted to the GCS.

When the data is received by the UAV, the first four bytes of each packet are collected since they carry the serial number. The serial number is compared to a list on the UAV side. The Key is determined based on the serial number, and the message is decrypted using a reverse Caesar cypher. When we decrypt the data, we obtain a character string that is still encrypted with our custom mapping and can only be reverted using our mapping reverting approach. We have also included encryption in the Mission Planner to enable secure connection between the SITL UAV's autopilot and the Ground Control Station to decrypt the acquired packet and extract the original IOT_MQTT transmission. To carry out this encryption procedure, an algorithm is created. Java is used to implement the code. The pseudo-code for Algorithm 3 is presented below in two steps: Algorithm 3 for encrypting and transmitting packets and Algorithm 4 for receiving and decrypting packets.

```
Algorithm 3: Encrypting the mavlink packet
Input: Mavlink_Packets
Output: Encrypted_Mavlink_Packets
Procedure (Encryption_Mavlink)
1    CreateDroneObject
2    $D0 \leftarrow Packet_{Data}$
3    inveral ← IV
4    $if\ (Packet_{Count} = Interval)$
5    Reset(counter)
6    Swap(Key_List)
7    PickRandomIndex
8    Fetch_Key ← Rand(Index)
9    Hex_String ← Convert(Fetch_Key)
10   For(n ← Each_Char)
11   if (Char == Alphabet)
12   Unpack(Packet)
13   Received_Data(Binary, Hex)
14   Plain_Text ← Convert(Binary, Hex)
15   Switch
16   Custom_Mapping(Char) → Encrypted_String
17   Sec_Inf ← Get(GPS, payload, alt, etc., )
18   End procedure
```

```
Algorithm 4: Receiving and decrypting packet
Input: Mavlink_Packets
Output: Decrypted_Mavelink_Packets
  Procedure(Decryption_mavelink)
1   CreateDroneObject
2   D0 ← Packet_Data
3   inveral ← IV
4   if (Packet_Count = Interval)
5   Reset(counter)
6   Swap(Key_List)
7   S_number ← Fetch(Packet)//Fetch the serial number form first 4 bytes
8   Fetch_Key ← List(S_Number)
9   Remove first 4 Bytes
10  For(n ← Each_Char)
11  Switch
12  Custom_Reverse_Mapping(Char) → Decrypted_String
13  if (Char == Alphabet)
14  Caesarcipher_Decrypt(Char, Key)
15  Else if (Char == Num)
16  Caesarcipher_Decrypt(Char, Key − 2)
17  Original Mavlink Packet
18  End Procedure
```

When transferring UAV and GCS data, the serial numbers are drawn at random from a table for each request. It allows you to change the key for each request and encrypt each packet with a separate key. This implies that the following IOT_MQTT packet will not have the same serial number but a new serial number with a new key to encrypt the data. After completing a hundred queries, the serial numbers and Key in the list are mixed randomly in parallel on both sides so that the lists are comparable on both sides. If the list at one end is not updated after each iteration and transmitted to the other, communication cannot occur, and the UAV/GCS will be regarded as illegal.

Performance Evaluation and Benchmarking

This section thoroughly examines the IOT_MQTT protocol's efficiency in conjunction with our encryption approach for the protocol's security. Furthermore, the output is rated in terms of resource usage, such as CPU processing and memory consumption rate.

We compare our suggested method to the original, unsecure IOT_MQTT protocol. In contrast, our method ensures that communication between the UAV and GCS is safe and that the information provided is not compromised. As a result, our method encrypts the IOT_MQTT packet without compromising its performance or efficiency.

The experiment involves running a PC with an Intel 2.6 GHz Core i7 CPU and 8 GB of RAM. The operating system is Microsoft Windows 10 (64-bit), and the software includes Mission Planner 1.3.74, Ardupilot version 3.2.1, and SITL UAV copter. UDP port 14550 is used to link the UAV.

Security

The experiment results reveal that the packet is encrypted using our encryption approach. When a packet is seized, the information communicated cannot be recovered. To put this to the test, we examined the communication between the UAV and GCS using our IOT_MQTT encryption approach. The secured IOT_MQTT packets are first collected using Wireshark. The captured packets are then sent to Algorithm 1 to launch attacks. The findings reveal that when the packets are transmitted, the mission planner does not identify them since the packet's data cannot be retrieved. Furthermore, Algorithm 2 is used to determine whether or not the packets can be translated into a human-readable format; the results are negative. This indicates that the packets are worthless for replicating to start attacks. They are encrypted to protect the communication between the UAV and the GCS.

Transfer Speed (Packets Count)

The experimental findings demonstrate that our technique sends nearly the same number of packets per second as the original IOT_MQTT. There is only a small difference of one or two packets up and down, which is insignificant. Figure 6 depicts the number of packets transferred per second by the original IOT_MQTT protocol and our method.

Memory Consumption

Memory usage is another significant measure for evaluating performance. The findings reveal that our approach consumes about the same amount of RAM as the original IOT_MQTT packet, as seen in Figure 7.

CPU Usage

Memory usage is another significant measure for evaluating performance. The findings reveal that our approach consumes about the same amount of RAM as the original IOT_MQTT packet, as seen in Figure 8.

Conclusion

This study develops and implements a novel way to secure the IOT_MQTT communication protocol. The method relies on a cryptographic encryption algorithm and custom mapping. To protect the entire packet, an extra security layer is added to the IOT_MQTT communication protocol. A new list idea is added, communicating a serial number rather than the secret Key for encryption. The Key is verified against the serial number of both sides, and the communications are encrypted and decrypted. The findings are obtained by modelling the environment using a virtual UAV via Ardupilot SITL, which employs the same autopilot as the actual Planes and UAVs. The results indicate that our method secures communication while preserving the original protocol's performance and efficiency. To validate the results in terms of transfer speed, performance, and efficiency, the suggested method is compared to existing literature, such as MAVsec, and benchmarked against the original insecure IOT_MQTT protocol. The present scope and limitations of the work are that it is best suited for missions lasting from 10 minutes to 3 hours. In the future, we are focusing on making the protocol adaptive \sand self-deciding to apply different degrees of encryption based on the mission needs.

References

[1] Y. Zeng, R. Zhang and T. J. Lim, "Wireless communications with unmanned aerial vehicles: Opportu- nities and challenges," IEEE Communications Magazine, vol. 54, no. 5, pp. 36–42, 2016.

[2] J. Janousek and P. Marcon, "Precision landing options in unmanned aerial vehicles," in 2018 Int. Interdisciplinary PhD Workshop, Poland, pp. 58–60, 2018.

[3] N. A. Khan, S. N. Brohi and N. Z. Jhanjhi, "UAV's applications, architecture, security issues and attack scenarios: A survey," in Intelligent Computing and Innovation on Data Science. Berlin, Germany: Springer, pp. 753–760, 2020.

[4] M. H. Choi, B. Shirinzadeh and R. Porter, "System identification-based sliding mode control for small- scaled autonomous aerial vehicles with unknown aerodynamics derivatives," IIEEE/ASME Transactions on Mechatronics, vol. 21, no. 6, pp. 2944–2952, 2016.

[5] B. G. Maciel-Pearson, S. Akçay, A. Atapour-Abarghouei, C. Holder and T. P. Breckon, "Multi-task regression-based learning for autonomous unmanned aerial vehicle flight control within unstructured outdoor environments," IEEE Robotics and Automation Letters, vol. 4, no. 4, pp. 4116–4123, 2019.

[6] W. Kwon, J. H. Park, M. Lee, J. Her, S.-H. Kim et al., "Robust autonomous navigation of unmanned aerial vehicles (UAVs) for warehouses' inventory application," IEEE Robotics and Automation Letters, vol. 5, no. 1, pp. 243–249, 2019.

[7] T. Samad, J. S. Bay and D. Godbole, "Network-centric systems for military operations in urban terrain: The role of UAVs," Proc. of the IEEE, vol. 95, no. 1, pp. 92–107, 2007.

[8] V. Roberge, M. Tarbouchi and G. Labonté, "Fast genetic algorithm path planner for fixed-wing military UAV using GPU," IEEE Transactions on Aerospace and Electronic Systems, vol. 54, no. 5, pp. 2105–2117, 2018.

[9] D. Orfanus, E. P. de Freitas and F. Eliassen, "Self-organization as a supporting paradigm for military UAV relay networks," IEEE Communications Letters, vol. 20, no. 4, pp. 804–807, 2016.

[10] M. Saleh, N. Jhanjhi and A. Abdullah, "Proposing a privacy protection model in case of civilian drone," in Proc. 2020 22nd Int. Conf. on Advanced Communication Technology, Phoenix Park, South Korea, pp. 596–602, 2020.

[11] G. Quiroz and S. J. Kim, "A confetti drone: Exploring drone entertainment," in Proc. 2017 IEEE Int. Conf. on Consumer Electronics, Las Vegas, USA, pp. 378–381, 2017.

[12] D. Murugan, A. Garg and D. Singh, "Development of an adaptive approach for precision agriculture monitoring with drone and satellite data," IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, vol. 10, no. 12, pp. 5322–5328, 2017.

[13] A. M. Jawad, H. M. Jawad, R. Nordin, S. K. Gharghan, N. F. Abdullah et al., "Wireless power transfer with magnetic resonator coupling and sleep/active strategy for a drone charging station in smart agriculture," IEEE Access, vol. 7, no. 1, pp. 139839–139851, 2019.

[14] I. Bor-Yaliniz, S. S. Szyszkowicz and H. Yanikomeroglu, "Environment-aware drone-base-station place- ments in modern metropolitans," IEEE Wireless Communications Letters, vol. 7, no. 3, pp. 372–375, 2017.

[15] A. N. Chaves, P. S. Cugnasca and J. Jose, "Adaptive search control applied to search and rescue operations using unmanned aerial vehicles (UAVs)," IEEE Latin America Transactions, vol. 12, no. 7, pp. 1278–1283, 2014.

[16] N. A. Khan, N. Z. Jhanjhi, S. N. Brohi, R. S. A. Usmani and A. Nayyar, "Smart traffic monitoring system using unmanned aerial vehicles (UAVs)," Computer Communications, vol. 157, no. 1, pp. 434– 443, 2020.

[17] D. Wang, P. Hu, J. Du, P. Zhou, T. Deng et al., "Routing and scheduling for hybrid truck-drone col- laborative parcel delivery with independent and truck-carried drones," IEEE Internet of Things Journal, vol. 6, no. 6, pp. 10483–10495, 2019.

[18] A. Tariq, S. M. Osama and A. Gillani, "Development of a low cost and light weight UAV for photogrammetry and precision land mapping using aerial imagery," in Proc. 2016 Int. Conf. on Frontiers of Information Technology, Islamabad, Pakistan, pp. 360–364, 2016.

[19] A. Gurtner, D. G. Greer, R. Glassock, L. Mejias, R. A. Walker et al., "Investigation of fish-eye lenses for small-UAV aerial photography," IEEE Transactions on Geoscience and Remote Sensing, vol. 47, no. 3, pp. 709–721, 2009.

[20] I. Mademlis, V. Mygdalis, N. Nikolaidis, M. Montagnuolo, F. Negro et al., "High-level multiple-UAV cinematography tools for covering outdoor events," IEEE Transactions on Broadcasting, vol. 65, no. 3, pp. 627–635, 2019.

[21] E. Natalizio, N. R. Zema, E. Yanmaz, L. D. P. Pugliese and F. Guerriero, "Take the field from your smartphone: Leveraging UAVs for event filming," IEEE Transactions on Mobile Computing, vol. 19, no. 8, pp. 1971–1983, 2019.

[22] G. D. Georgiev, G. Hristov, P. Zahariev and D. Kinaneva, "Forest monitoring system for early fire detection based on convolutional neural network and UAV imagery," in Proc. 2020 28th National Conf. with International Participation, Sofia, Bulgaria, pp. 57–60, 2020.

[23] C. Chen, C. Grier, A. Malfa, M. Booen, C. Xia et al., "High-speed optical links for UAV applications," in Free-Space Laser Communication and Atmospheric Propagation XXIX. vol. 10096. California, United States, 1009615–1009626, 2017.

[24] J. Russell, "Facebook is reportedly testing solar-powered internet drones again—this time with Air- bus, techcrunch.com," 2019. [Online]. Available: https://techcrunch.com/2019/01/21/facebook-airbus- solar-drones-internet-program/ (Accessed Feb. 28, 2021).

[25] E. Salamí, C. Barrado and E. Pastor, "UAV flight experiments applied to the remote sensing of vegetated areas," Remote Sensing, vol. 6, no. 11, pp. 11051–11081, 2014.

[26] S. Berrahal, J.-H. Kim, S. Rekhis, N. Boudriga, D. Wilkins et al., "Border surveillance monitoring using quadcopter UAV-aided wireless sensor networks," Journal of Communications Software and Systems, vol. 12, no. 1, pp. 67–82, 2016.

[27] L. Krichen, M. Fourati and L. C. Fourati, "Communication architecture for unmanned aerial vehicle system," in Int. Conf. on Ad-Hoc Networks and Wireless, pp. 213–225, 1st ed., Chap. 1, Sec. 1. USA, Cham, Springer, 2018.

[28] J. Gertler, "US unmanned aerial systems," Library of Congress Washington DC Congressional Research Service, vol. 1, no. 1, pp. 1–10, 2012.

[29] N. A. Khan, N. Jhanjhi, S. N. Brohi and A. Nayyar, "Emerging use of UAV's: Secure communication protocol issues and challenges," in Drones in Smart-cities: Security and Performance, 1st ed., vol. 1. pp. 37–55, Chap. 3, Sec. 1. Turkey: Elsevier, 2020.

[30] U. development Team, "UAVCAN Communication Protocol," 2014. [Online]. Available: https://uavcan. org/Specification/1._ Introduction/ (Accessed Aug. 28, 2019).

[31] V. Kriz and P. Gabrlik, "Uranuslink-communication protocol for uav with small overhead and encryption ability," IFAC-PapersOnLine, vol. 48, no. 4, pp. 474–479, 2015.
[32] Y.-M. Kwon, "Vulnerability analysis of the IOT_MQTT protocol for Unmanned Aerial Vehicles," Ph.D. dissertation. Univerity of Daegu Gyeongbuk Institute of Science and Technology (DGIST), 2018.
[33] S. Atoev, K.-R. Kwon, S.-H. Lee and K.-S. Moon, "Data analysis of the IOT_MQTT communication protocol," in Proc. 2017 Int. Conf. on Information Science and Communications Technologies, Tashkent, Uzbekistan, pp. 1–3, 2017.
[34] S. Veena, S. Vaitheeswaran and H. Lokesha, "Towards the development of secure mavs," in Proc. ICRAMAV-2014 (3rd Int. Conf.), India, pp. 1–10, 2014.
[35] A. Koubâa and B. Qureshi, "Dronetrack: Cloud-based real-time object tracking using unmanned aerial vehicles over the internet," IEEE Access, vol. 6, no. 1, pp. 13810–13824, 2018.
[36] AD Team, "Ardupilot project," 2009. [Online]. Available: https://ardupilot.org/index.php/about (Accessed Feb. 28, 2021).
[37] A. Allouch, O. Cheikhrouhou, A. Koubaa, M. Khalgui and T. Abbes, "MAVSec: Securing the IOT_MQTT protocol for ardupilot/PX4 unmanned aerial systems," in Proc. 2019 15th Int. Wireless Communications & Mobile Computing Conf., Tangier, Morocco, pp. 1–9, 2019.
[38] PD Team, "PX4 Autopilot," 2012. [Online]. Available: https://px4.io/ (Accessed Feb. 28, 2021).
[39] A. Tridgell and L. Meier, "IOT_MQTT 2.0 packet signing proposal," Scientific report, 2015. [Online]. Available: https://lists.linuxfoundation.org/pipermail/dronecode-tsc/2015-October/000171.html.
[40] G. William, B. Elizabeth, B. Mike, S. Daniel, S. Ryan et al., "Challenges of securing and defending unmanned aerial vehicles," in National Cyber Summit (NCS) Research Track, 1st ed., vol. 1. New York, USA: Springer, pp. 119–138, 2020.
[41] N. Hoque, D. K. Bhattacharyya and J. K. Kalita, "Botnet in DDoS attacks: Trends and challenges,"
[42] IEEE Communications Surveys & Tutorials, vol. 17, no. 4, pp. 2242–2270, 2015.
[43] G. Vasconcelos, G. Carrijo, R. Miani, J. Souza and V. Guizilini, "The impact of DoS attacks on the AR. Drone 2.0," in Proc. 2016 XIII Latin American Robotics Symp. and IV Brazilian Robotics Symp., Recife, Brazil, pp. 127–132, 2016.
[44] J. Whelan, A. Almehmadi, J. Braverman and K. El-Khatib, "Threat Analysis of a long range autonomous unmanned aerial system," in Proc. 2020 Int. Conf. on Computing and Information Technology, Tabuk, Saudi Arabia, pp. 1–5, 2020.
[45] K. Yoon, D. Park, Y. Yim, K. Kim, S. K. Yang et al., "Security authentication system using encrypted channel on UAV network," in Proc. 2017 First IEEE Int. Conf. on Robotic Computing, Taichung, Taiwan, pp. 393–398, 2017.
[46] A. Shoufan, H. AlNoon and J. Baek, "Secure communication in civil drones," in Int. Conf. on Information Systems Security and Privacy, cham, Springer, pp. 177–195, 2015.
[47] B. S. Rajatha, C. M. Ananda and S. Nagaraj, "Authentication of MAV communication using Cae- sar Cipher cryptography," in 2015 Int. Conf. on Smart Technologies and Management for Computing, Communication, Controls, Energy and Materials, Avadi, India, pp. 58–63, 2015.
[48] J. A. Marty, "Vulnerability analysis of the IOT_MQTT protocol for command and control of unmanned aircraft," MS dessertation, Air Force Institute of Technology, 2013.
[49] R. Hamsavahini, S. Varun and S. Narayana, "Development of light weight algorithms in a customized communication protocol for micro air vehicles," International Journal Of Latest Research In Science And Technology, vol. 1, no. 1, pp. 73–79, 2016.
[50] N. Butcher, A. Stewart and S. Biaz, Securing the IOT_MQTT communication protocol for unmanned aircraft systems. in Technical Report # CSSE 1402 (2014). USA: Appalach State University. Auburn University, 2013.
[51] J. Bian, R. Seker and M. Xie, "A secure communication framework for large-scale unmanned aircraft systems," in Proc. 2013 Integrated Communications, Navigation and Surveillance Conf., Herndon, VA, USA, pp. 1–12, 2013.
[52] R. Altawy and A. M. Youssef , "Security, privacy, and safety aspects of civilian drones: A survey,"
[53] ACM Transactions on Cyber-Physical Systems, vol. 1, no. 2, pp. 7, 2017.
[54] A. Koubaa, A. Allouch, M. Alajlan, Y. Javed, A. Belghith et al., "Micro air vehicle link (IOT_MQTT) in a nutshell: A Survey," IEEE Access, vol. 7, no. 1, pp. 87658– 87680, 2019.
[55] F. T. AL-Dhief, N. Sabir, N. M. A.Latif, N. N. N. A. Malik, M. A. A. Albader et al., "Performance comparison between Tcp And Udp protocols in different simulation scenarios," International Journal of Engineering & Technology, vol. 7, no. 4.36, pp. 172–176, 2018.
[56] W. Goralski, "The illustrated network: How TCP/IP works in a modern network," in The Illustrated Network: How TCP/IP Works in a Modern Network Morgan Kaufmann, 2 ed., Chennai, India: Elsevier, 2017.
[57] Anis Koubaa, "MAVSec: Securing the IOT_MQTT protocol for ardupilot and PX4 unmanned aerial systems," 2019. [Online]. Available: https://www.slideshare.net/AnisKoubaa/mavsec-securing-the-IOT_MQTT- protocol-for-ardupilot-and-px4-unmanned-aerial-systems (Accessed Mar. 17, 2021).