Cryptography-based military UAV communication using RSA with a responsive GUI

[*]This report has been made using LaTeX

K. Eswar, V. Haswanth Raj, R. Balakrishna, Ch. Rahul

I. Introduction

As aviation technology gradually advances, the use of UAVs is gradually expanding in environments where are difficult for humans to perform directly and in dangerous environments. However, due to the security vulnerabilities of UAVs, illegal and malicious attacks against UAVs such as eavesdropping on communication data and obtaining UAV controls are also increasing. One of the solutions to prevent such attacks are to encrypt UAV's communication data and stored sensitive information. Therefore, we propose a drone security module to protect the communication data and storage information of the UAV, and briefly present the hardware design and software driver design of the drone security module. The drone security module is connected to the flight control computer or mission computer through a USB interface, and encrypts the control signal and telemetry data from the UAV to the ground control station.

Recently, a UAV(Unmanned Aerial Vehicle), also called as a drone has been used in various fields such as delivery, cartography, meteorological observation, agriculture, facility inspection, and taking a video. However, there are also attacks that take control of the UAVs and use them for illegal activities such as terrorism. In addition, attacks in which intercept and acquire telemetry information such as sensing data and location information transmitted from the UAV to the ground control station through communication channels have been reported [3,4,5]. In other words, UAVs without security functions can be used for criminal activities such as safety threats, security threats, information theft, and invasion of privacy. To prevent these cybersecurity vulnerabilities, special purpose UAVs operated by state agencies or military UAVs for surveillance, reconnaissance and attack purposes require data encryption technology for UAVs to protect not only communication data but also sensitive information stored in the UAV

II. Key sharing with the help of RSA

The base station of a UAV acts as a control center for many drones hence a separate communication channel is required for each drone and the base station. Hence, for a UAV to communicate it needs to have accessibility to the public key of the base station to encrypt important messages like GPS coordinates etc.

There should be a secure way of communication between the UAV and the base station. The base station stores all it's UAVs' public keys for sending encrypted messages. If the desired UAV is in the range of the base station the base station uses the existing public key of the UAV which the base station has and encrypts the message and then sends the signal to all probable directions of the UAV. Once, UAV receives the message it can decrypt it and can have secure communication. To have two-way communication the base station encrypts its own public key and sends it to the desired UAV and once the UAV gets the public key of the base station it can communicate back and forth. This type of cryptographic encryption for communication takes place, especially in military UAVs.

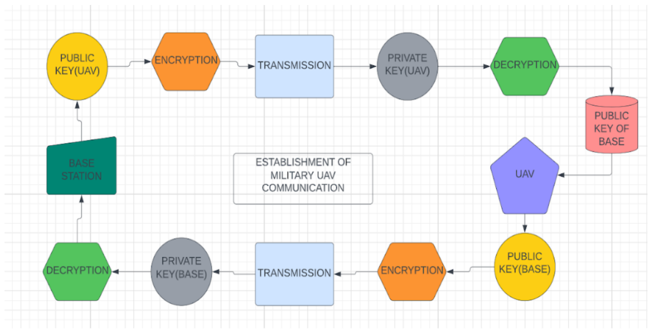


Fig. 1. Project flow chart

The public key of the control station is not made available to all of the base station's drones for security purposes hence the public key is sent to the desired UAVs only and the public key of the UAV is safely present with the base station. Hence, well round security is achieved.

As RSA is one of the most commonly known asymmetric encryption techniques it is used in this project other encryption standards like AES are also used for military UAV communication. RSA is chosen because it is the most common and it is tough to crack.

A. *Working of RSA*

RSA typically includes operations with prime numbers. The larger the primes the stronger the algorithm gets. Hence, two different prime numbers (p,q) are selected and their product (n) is calculated along with its **euler's toeint fucntion**(ϕ). Then a random number (e) between one to the product of the selected primes is taken such that the GCD of the random number and the euler's toeint function is one. Hence, we would acquire our public key. The multiplicative inverse of public key *mod* the euler's toeint function gives us the private key. This is the basic working of RSA which is explained in the following equations.

$$n = p * q \tag{1}$$

$$\phi(n) = (p-1) * (q-1) \tag{2}$$

$$GCD(\phi(n), e) = 1 \tag{3}$$

$$(e * d) \mod (\phi(n)) = 1 \tag{4}$$

$$PU = e, n \tag{5}$$

$$PR = d, n \tag{6}$$

$$C = M^e \mod (n) \tag{7}$$

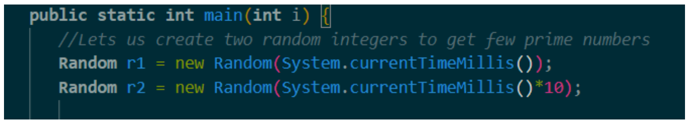
$$M = C^d \mod (n) \tag{8}$$

Equations (7) and (8) mathematically express the encryption and decryption using the keys generated using the RSA algorithm.

B. Code Snippets

NOTE: The entire code is written in *JAVA*.

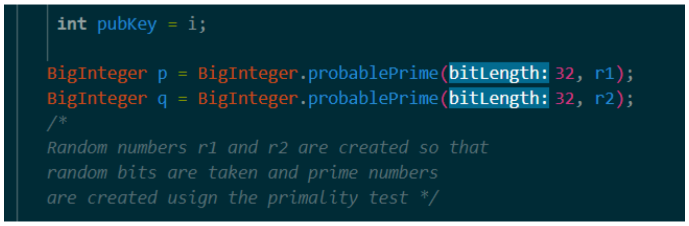
- For randomly generating prime numbers we need to either take random values from nature or use pseudo-random generators. Hence, time in milliseconds at any instant is taken as a factor to generate a random number. The second number is generated by multiplying the first number by a factor.



```java
public static int main(int i) {
    //Lets us create two random integers to get few prime numbers
    Random r1 = new Random(System.currentTimeMillis());
    Random r2 = new Random(System.currentTimeMillis()*10);
```

Fig. 2. Caption

- Then random bits of the selected numbers are taken and numbers are generated using the inbuilt random function call and then the produced numbers are checked for a primality test. Therefore two random primes are achieved.

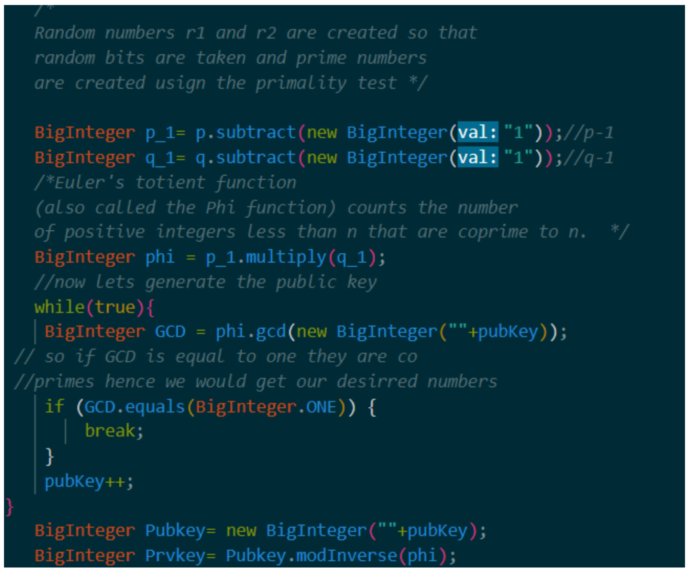


```java
    int pubKey = i;

  BigInteger p = BigInteger.probablePrime(bitLength: 32, r1);
  BigInteger q = BigInteger.probablePrime(bitLength: 32, r2);
  /*
  Random numbers r1 and r2 are created so that
  random bits are taken and prime numbers
  are created usign the primality test */
```

Fig. 3. Caption

- After the generation, the RSA key generation is implemented using inbuilt mathematical functions by calling them. The value for the public key is selected using the input from the user. If the user input satisfies the GCD condition it is used else the value of the input is incremented and the condition will be checked again, now the private key is also calculated.

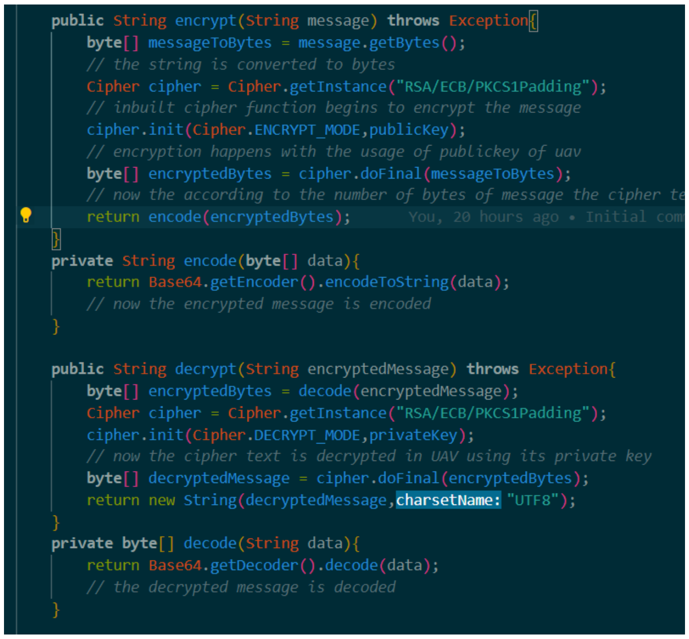


```java
  /*
  Random numbers r1 and r2 are created so that
  random bits are taken and prime numbers
  are created usign the primality test */

  BigInteger p_1= p.subtract(new BigInteger(val: "1"));//p-1
  BigInteger q_1= q.subtract(new BigInteger(val: "1"));//q-1
  /*Euler's totient function
  (also called the Phi function) counts the number
  of positive integers less than n that are coprime to n.   */
  BigInteger phi = p_1.multiply(q_1);
  //now lets generate the public key
  while(true){
   BigInteger GCD = phi.gcd(new BigInteger(""+pubKey));
// so if GCD is equal to one they are co
//primes hence we would get our desirred numbers
   if (GCD.equals(BigInteger.ONE)) {
       break;
   }
   pubKey++;
}
  BigInteger Pubkey= new BigInteger(""+pubKey);
  BigInteger Prvkey= Pubkey.modInverse(phi);
```

Fig. 4. Caption

- Hence, a pair of keys are generated for the base station. The public key of the base is now taken as the message and it is encrypted using inbuilt key generating and encrypting functions of RSA of the JAVA security library to display the process of encryption of the message at the base station and the decryption at the UAV side. In the end, if the message received contains teh public key of the base station the aim of the project is achieved.

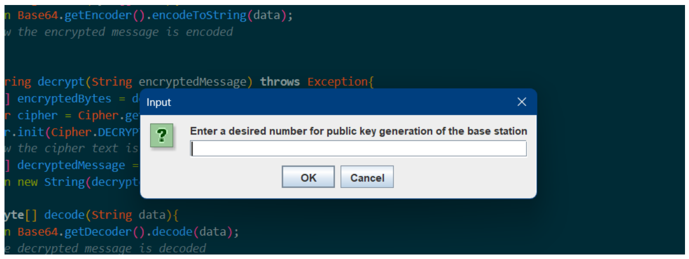


```java
public String encrypt(String message) throws Exception{
    byte[] messageToBytes = message.getBytes();
    // the string is converted to bytes
    Cipher cipher = Cipher.getInstance("RSA/ECB/PKCS1Padding");
    // inbuilt cipher function begins to encrypt the message
    cipher.init(Cipher.ENCRYPT_MODE,publicKey);
    // encryption happens with the usage of publickey of uav
    byte[] encryptedBytes = cipher.doFinal(messageToBytes);
    // now the according to the number of bytes of message the cipher te
    return encode(encryptedBytes);       You, 20 hours ago • Initial com
}
private String encode(byte[] data){
    return Base64.getEncoder().encodeToString(data);
    // now the encrypted message is encoded
}

public String decrypt(String encryptedMessage) throws Exception{
    byte[] encryptedBytes = decode(encryptedMessage);
    Cipher cipher = Cipher.getInstance("RSA/ECB/PKCS1Padding");
    cipher.init(Cipher.DECRYPT_MODE,privateKey);
    // now the cipher text is decrypted in UAV using its private key
    byte[] decryptedMessage = cipher.doFinal(encryptedBytes);
    return new String(decryptedMessage,charsetName: "UTF8");
}
private byte[] decode(String data){
    return Base64.getDecoder().decode(data);
    // the decrypted message is decoded
}
```

Fig. 5. Caption

- A responsive and interactive GUI is encoded and made available for this entire project



```
n Base64.getEncoder().encodeToString(data);
w the encrypted message is encoded


ring decrypt(String encryptedMessage) throws Exception{
] encryptedBytes = d
r cipher = Cipher.ge
r.init(Cipher.DECRYP
w the cipher text is
] decryptedMessage =
n new String(decrypt

yte[] decode(String data){
n Base64.getDecoder().decode(data);
e decrypted message is decoded
```

Input

Enter a desired number for public key generation of the base station

Fig. 6. Caption

III. Inbuilt drone security modules

The drone security module driver is a software API that operates the hardware drone security module, issues commands for functions of Table 1 and receives responses. The software driver and the hardware drone security module communicate using the libusb library [7]. The drone security module software driver is implemented in C, C++, and Java, and runs on operating systems such as Linux, MacOS, Windows, and Android.

Most UAV mission computers run on Linux or Windows platforms, so we can apply the hardware drone security module and software driver to the mission computer as they are. When it comes to flight control computers, UAV manufacturers sometimes use their own flight control computers, most of which actually use the Pixhawk [7] family products. Pixhawk flight controller uses PX4 [8] firmware running on NuttX RTOS, so using the hardware drone security module on Pixhawk requires a software driver running on NuttX

IV. Related works

There are many cryptographic approaches to secure Unmanned Aerial vehicles(UAV's) or aerial drones that include the implementation of well-known protocols like RSA algorithm and Advanced Encryption Standard(AES) algorithms [2]. However, the authors of [2] proved in their work that standard techniques RSA and AES algorithms take a lot of computation time and consume a high amount of energy on small aerial drones.

The authors of [6] suggested a number of strategies and actions that may be used to raise the security of drone communication. Since drones are susceptible to GPS spoofing, anti-spoofing and anti-jamming receivers have been created. A great deal of productive work has been done, and methods for spotting and evading civilian GPS anti-spoofing and anti jamming have been proposed in the literature. These techniques can be divided into two categories: cryptographic (spread spectrum) and non-cryptographic (dual receiver correlation) (antenna array).These techniques, however, were either impossible to adopt or necessitated the purchase of expensive hardware. As a result, they suggested several simplified software-based spoof identification techniques.

The authors of [10] addressed the MAVLink protocol's security flaws and proposed MAVSec, a security-integrated MAVLink mechanism that relies on encryption algorithms to ensure the security of MAVLink messages sent between GCSs and UAVs. They used Ardupilot to test MAVSec and compared the efficiency of various encryption algorithms (such as ChaCha20, RC4, AES-CBC, and AES-CTR) in terms of memory utilization and CPU consumption. The results show that ChaCha20 outperforms and outperforms other encryption algorithms in terms of performance and efficiency. Integrating ChaCha20 into MAVLink will ensure message anonymity while using less memory and CPU, saving memory and energy for resource-constrained drones.

The authors of [1] proposed the system aims to propose authentication methods and ciphering of drone communications to provide mutual authentication between ground control station(GCS) and one drone with multiple flight sessions, propose Hash chacha20 lightweight algorithm to cipher the registers the flight session key in a centralized dataset in the ground station to increase the security of proposed authentication methods, and finally increase securing payload data for MAVLink protocol using HIGHT lightweight algorithm.

The authors of [11] proposed an open-source low energy cryptographic framework tailored for small aerial drones with an in-depth energy consumption analysis. Their framework integrates algorithmic optimizations to improve the performance of standard PKC techniques, all supported with lightweight symmetric ciphers. They have implemented and deployed our optimized framework on Crazyflie 2.0, and compared its performance with the standard techniques.

References

[1] Ismael, Hani M. ”Authentication and encryption drone communication by using HIGHT lightweight algorithm.” Turkish Journal of Computer and Mathematics Education (TURCOMAT) 12, no. 11 (2021): 5891-5908.

[2] Steinmann, Jessica A., Radu F. Babiceanu, and Remzi Seker. ”Uas security: Encryption key negotiation for partitioned data.” In 2016 Integrated Communications Navigation and Surveillance (ICNS), pp. 1E4-1. IEEE, 2016.

[3] Altawy, Riham, and Amr M. Youssef. ”Security, privacy, and safety aspects of civilian drones: A survey.” ACM Transactions on Cyber-Physical Systems 1, no. 2 (2016): 1-25.

[4] Krishna, CG Leela, and Robin R. Murphy. ”A review on cybersecurity vulnerabilities for unmanned aerial vehicles.” In 2017 IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR), pp. 194-199. IEEE, 2017.

[5] Dahiya, Susheela, and Manik Garg. ”Unmanned aerial vehicles: Vulnerability to cyber attacks.” In International Conference on Unmanned Aerial System in Geomatics, pp. 201-211. Springer, Cham, 2019.

[6] Dey, Vishal, Vikramkumar Pudi, Anupam Chattopadhyay, and Yuval Elovici. ”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. IEEE, 2018.

[7] https://libusb.info/

[8] https://pixhawk.org/

[9] https://px4.io/

[10] Allouch, Azza, Omar Cheikhrouhou, Anis Koubâa, Mohamed Khalgui, and Tarek Abbes. ”MAVSec: Securing the MAVLink protocol for ardupilot/PX4 unmanned aerial systems.” In 2019 15th International Wireless Communications and Mobile Computing Conference (IWCMC), pp. 621-628. IEEE, 2019.

[11] Ozmen, Muslum Ozgur, and Attila A. Yavuz. ”Dronecrypt-an efficient cryptographic framework for small aerial drones.” In MILCOM 2018-2018 IEEE Military Communications Conference (MILCOM), pp. 1-6. IEEE, 2018.

You can also check out the code for the entire project on our GitHub:

https://github.com/20bec023IIITDWD/RSA-Algorithm-for-UAV-cryptographic-communication