Secure UAV Communications System Design

Defensive write-up:

1. We decided to use **Asymmetric encryption(RSA)** and **Symmetric encryption(AES256-GCM)** for the following reasons:

   - Firstly to allow only provisioned SEDs to register.
     We generate key pairs (pk_k,sk_k) at the building process and a provisioned list. The registration message will be signed by the SEDs private key (sk_k). SSS upon receiving the message will verify using the corresponding public key(pk_k).
   - For targeted transmissions and broadcast transmissions we used AES256-GCM symmetric encryption to encrypt the message. We decided to use GCM mode to achieve the authenticity and integrity of the messages. To distribute the unique symmetric key we use asymmetric encryption to encrypt the key.

2. **Defensive features with crypto algorithms and sequence number:**
   a. The defensive features of using Asymmetric encryption(RSA) and symmetric encryption(AES256-GCM):
      - Firstly, the use of asymmetric encryption was supposed to prevent unprovisioned SEDs from registering. Second, asymmetric encryption should ensure the secure distribution of symmetric keys between parties for AES256-GCM.
      - The use of AES256-GCM was supposed to ensure the authenticity and integrity of the messages in a timely manner. The use of unique keys for each message was to prevent attack from reusing the previous message or brute force our key.

   b. In addition, to the above measure, we included a 2 bytes sequence number for each message to prevent replay attacks.

3. In our design we distribute the public keys of all SEDs as it would be required to retrieve the symmetric key. We think the key distribution was problematic. If sending SED does not have the receiver's public key,

there will be a public key exchange between the communicating SEDs. This public key exchange is done in plaintext. Our goal was to sign public key request messages with SSS public keys but we could not achieve this for some technical reasons.

We used sender's private key to generate ciphertext for broadcast messages. Receiver will use the corresponding public key to get the original message. We think this does not secure the broadcast transmission as our security was depended on the secrecy of the public key.

4. With knowledge of the system design we would make a more secure design by overcoming our design flaws listed below:

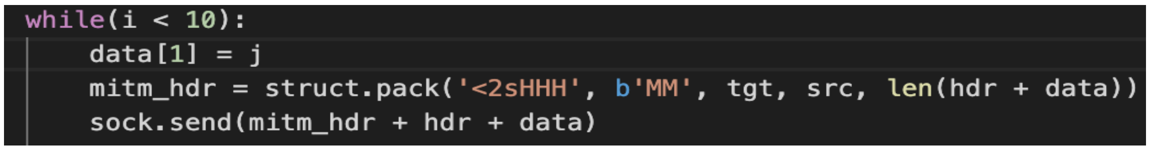
    - The public key exchange between SEDs should be signed and verified by SSS key pairs.
    - Avoid dependency on the secrecy of public keys for broadcast transmission.

Attack write-up:

We actually got only one flag for recovery mode in one sytem design.

Recovery mode flag:

In one design, we noticed that authenticity of the received message is not performed. From this intuition, we thought if we change any byte using man-in-the-middle attack we will be able to disrupt the checksum of the message. Figure 1 represents such a case, where we change the second byte with a random value to get the flag.



```
while(i < 10):
    data[1] = j
    mitm_hdr = struct.pack('<2sHHH', b'MM', tgt, src, len(hdr + data))
    sock.send(mitm_hdr + hdr + data)
```

Figure 1: Example of the attack

Prevent Recovery mode flag:

To prevent this attack an approach to check the integrity of the received message could be adopted. In the design, only decryption of the message is performed but as we can perform a man-in-the-middle attack the receiver will get a malicious plaintext after decryption.