# 50.020 Network Security Lab 8: Wireless Security

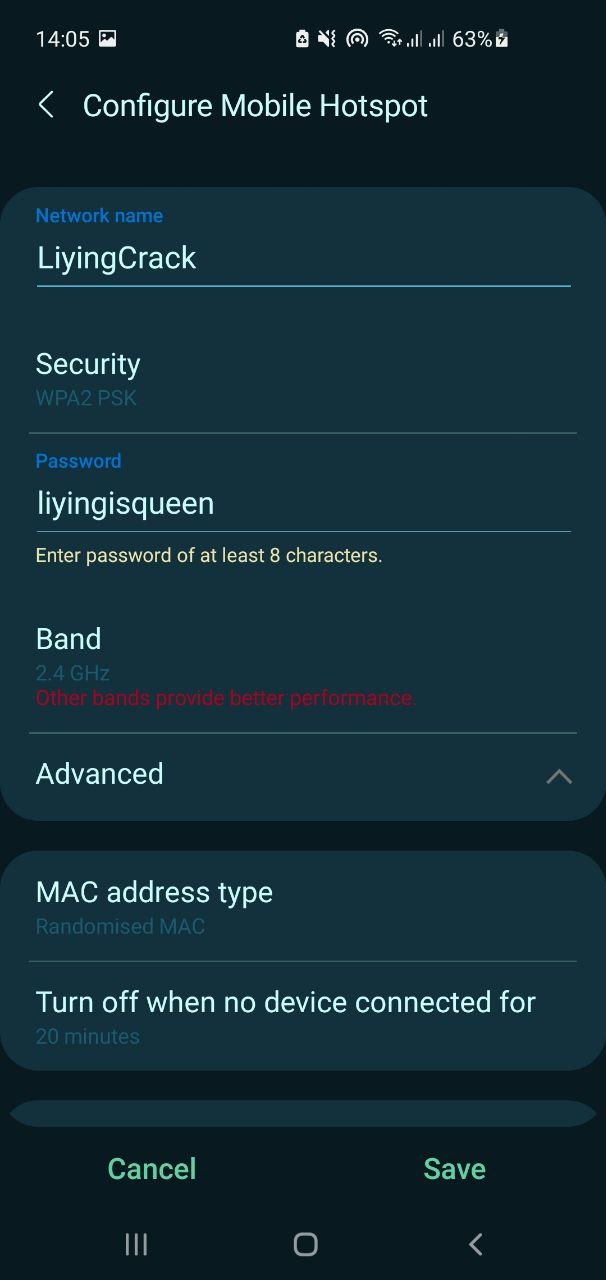
## Task 1: Setup an Access Point

Step 1: Set up an access point

Using my android phone’s mobile hotspot, I have set up the access point.

Step 2: Configure the SSID, username, and password

Unfortunately, I could not find a ‘username’ field in the hotspot configuration. The SSID is set to ‘LiyingCrack’ and the password is set to ‘liyingisqueen’ and as shown:



Step 3: Configure the security protocol to be WPA2

The security protocol is set to be WPA2 and it can be observed in the same screenshot above.

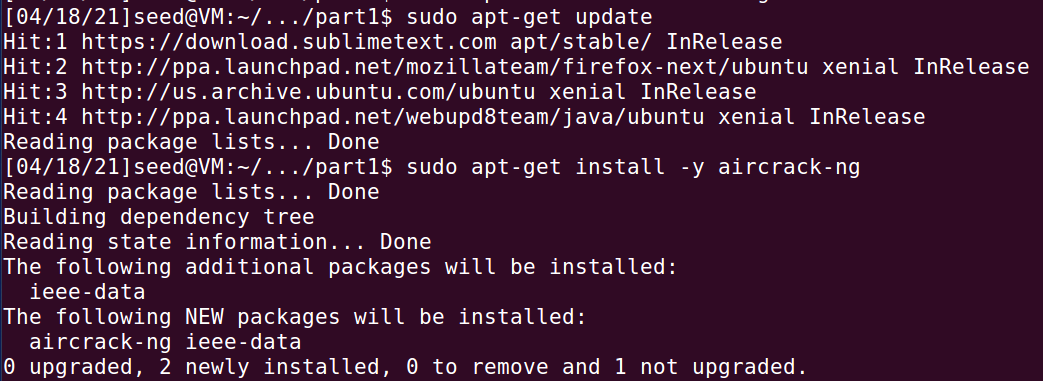
## Task 2: Capturing Wireless Packets

This part is skipped due to hardware constraints.

## Task 3: Capturing the Four-way Handshake

This part is largely skipped due to hardware constraints.

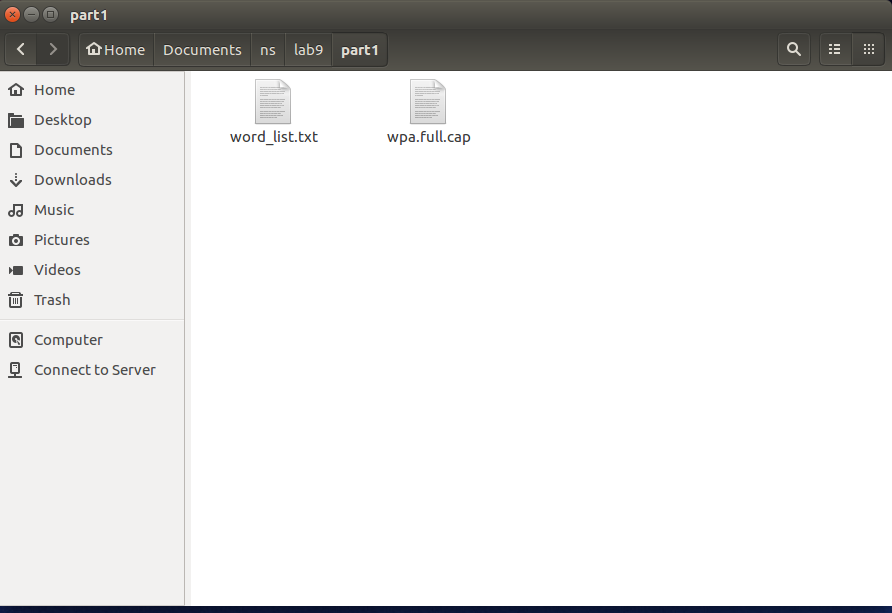
Aircrack-ng is installed on SEED using the following commands:



## Task 4: Cracking WPA2 WiFi Passphrase Using Aircrack-ng

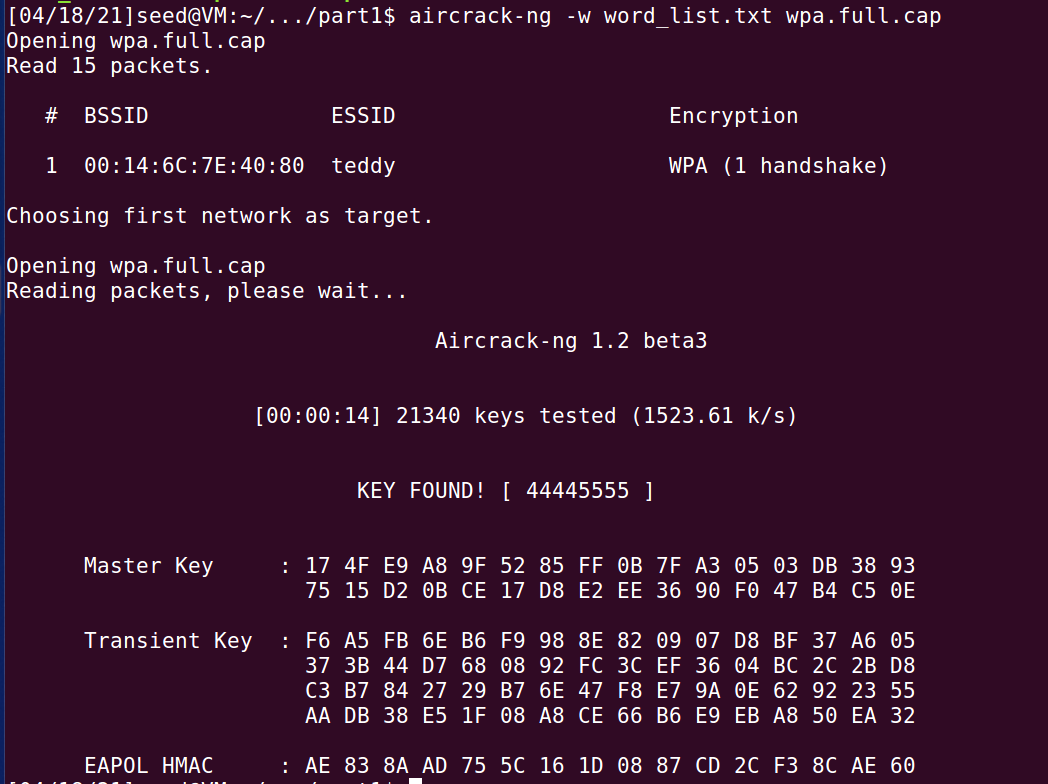
Step 1: Copy the cap/pcap file into the VM

The provided wpa.full.cap file word\_list.txt word list is transferred to the VM:



Step 2: Use aircrack-ng to crack the passphrase

To crack the passphrase using the word list provided, the following command is run:



The password set for the WiFi is shown to be ‘**44445555**’.

**Q: What is the difference between Monitor Mode and Promiscuous Mode?**

Monitor mode: Sniffing the packets in the air without connecting (associating) with any access point.

Promiscuous mode: Sniffing the packets after connecting to an access point. This is possible because the wireless-enabled devices send the data in the air but only "mark" them to be processed by the intended receiver. They cannot send the packets and make sure they only reach a specific device, unlike with switched LANs.

**Q: If the WiFi traffic is on-going, how to crack the WiFi password?**

Force users to reconnect to the network so as to capture the 4-way handshakes during their reconnect. This can be done by using airodump-ng to monitor the target AP in the background, and then using aireplay-ng to deauthenticate the users. Reference: <https://welkin.dev/2019/03/10/DOS-A-Router/>

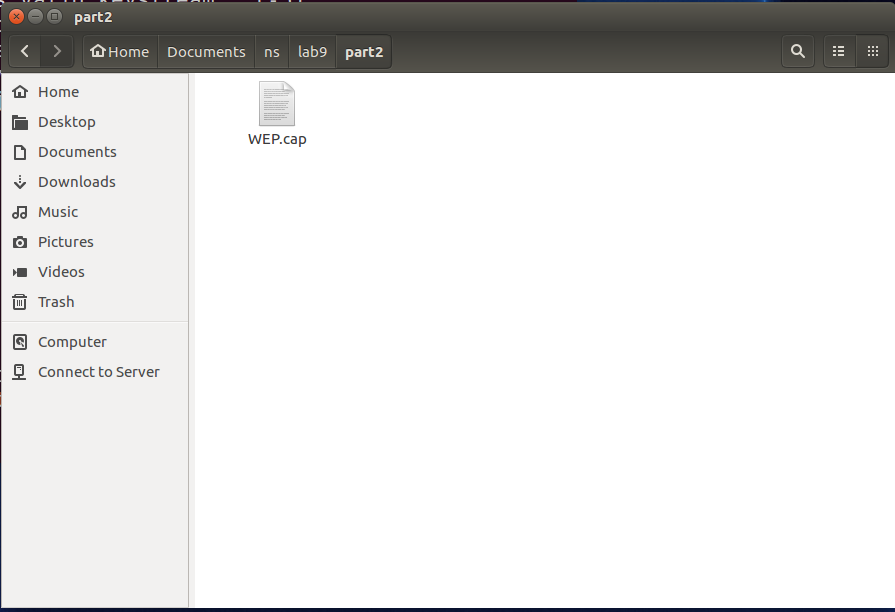
## Task 5: Cracking the WEP Password

Step 1: Install Aircrack-ng

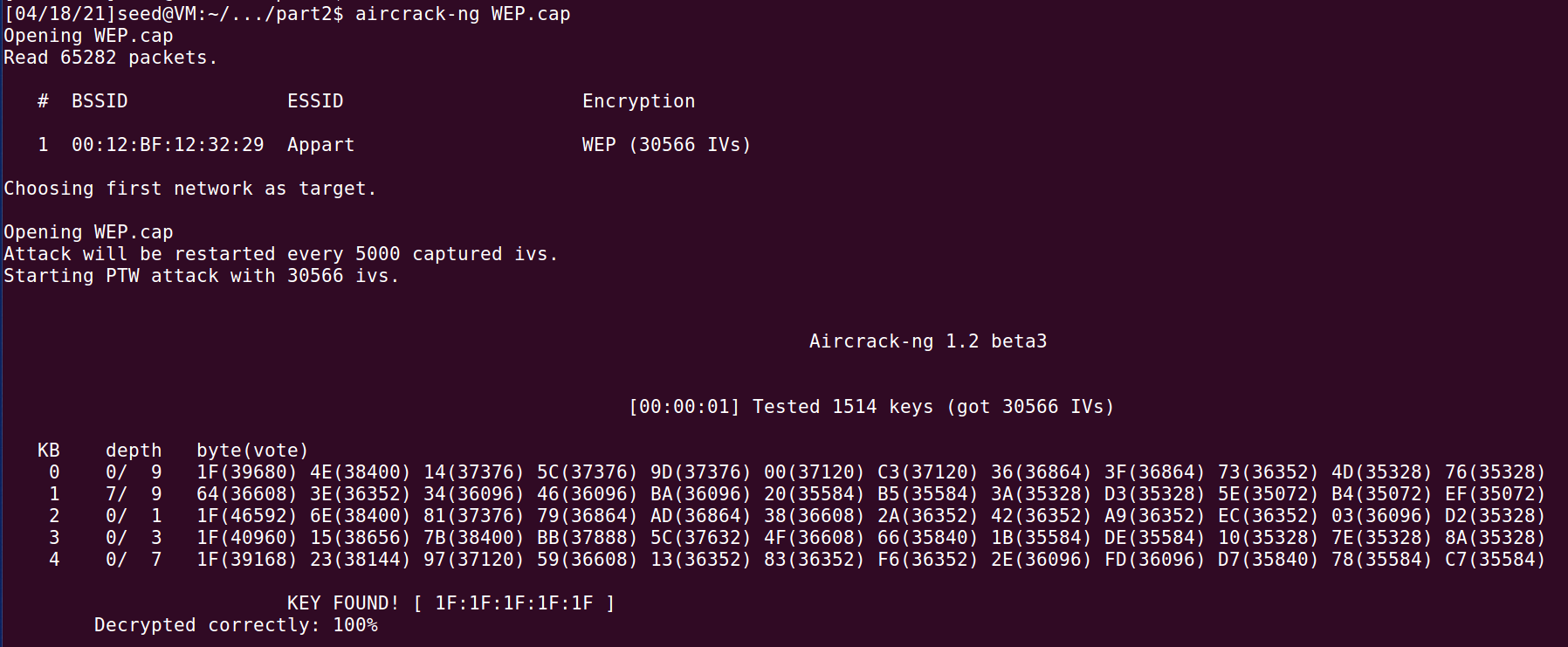
Done in the previous part.

Step 2: Cracking the WEP.cap

The provided WEP.cap file is transferred to the VM:



To crack the WEP protocol, the following command is run:



The cracked password/key is ‘**1F:1F:1F:1F:1F**’.

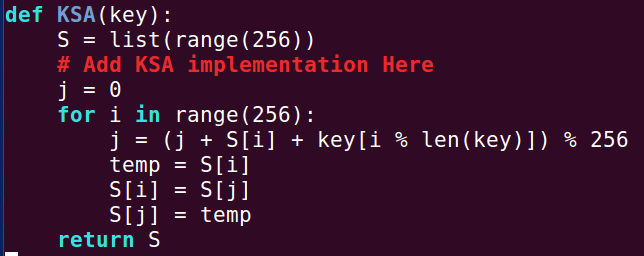
## Task 5: Cracking the WEP Packet

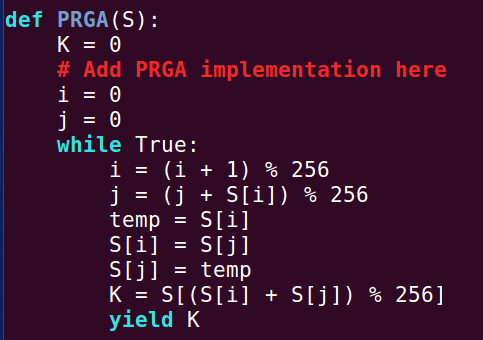
Step 1: Recall the WEP encryption process

Yup.

Step 2: Implement the RC4 Algorithm

A full implementation of the code can be found in rc4.py.

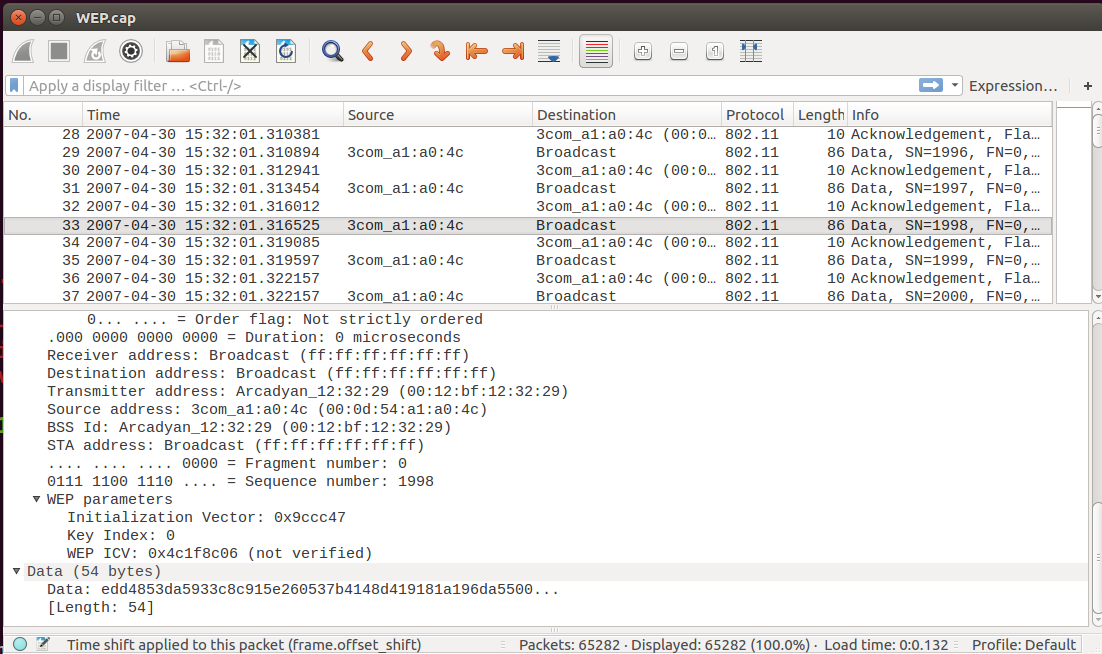
The pseudocode given in the handout is used to fill in the code for KSA:  


The pseudocode given in the handout is used to fill in the code for PRGA:  


Step 3: Verify Your Results

The test cases given are used to verify that the RC4 algorithm is correct. Each key and ciphertext pair (edit line 41 for key and line 42 for ciphertext) give the correct plaintext (printed on line 52) stated.

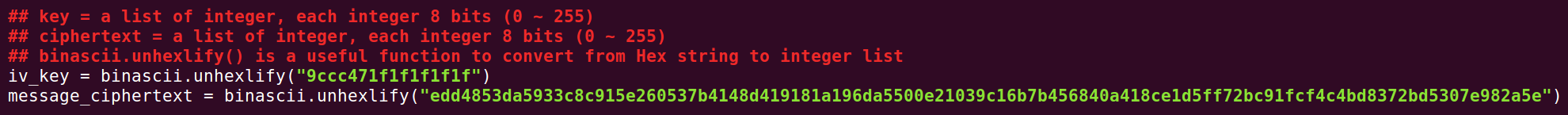
For the CRC, the sequence number chosen is SN1998 (my birth year!). The information is as shown on Wireshark:



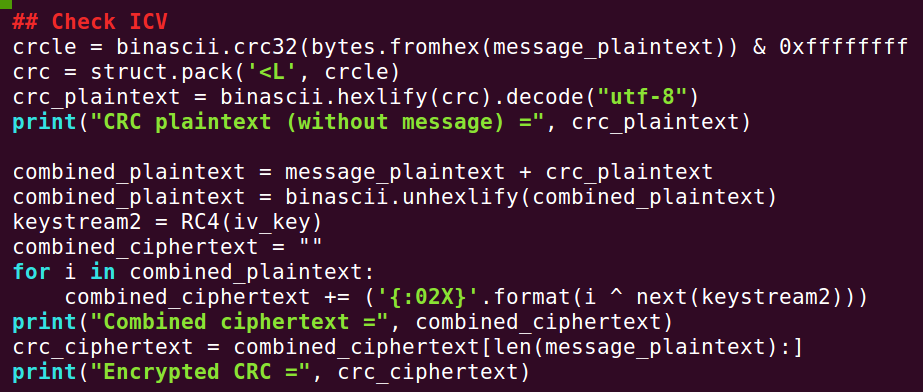
The information is recorded down properly using (right-click) > Copy > …as a Hex Stream:

* IV: **9ccc47**
* Data (encrypted message): **edd4853da5933c8c915e260537b4148d419181a196da5500e21039c16b7b456840a418ce1d5ff72bc91fcf4c4bd8372bd5307e982a5e**
* Encrypted ICV: **4c1f8c06**

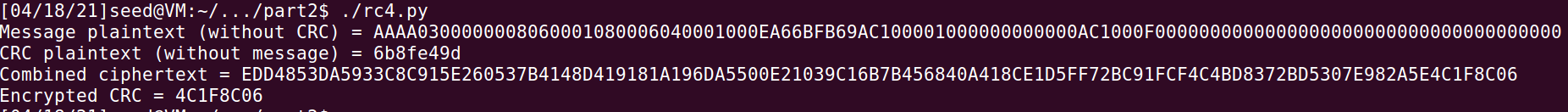
The IV is concatenated with the key (**1f1f1f1f1f**) to give the combined iv\_key (**9ccc471f1f1f1f1f**) and the message\_ciphertext is replaced with the data as shown:



The following code to compute the encrypted ICV is written as such:



The following output is shown:



To explain the output:

1. The message plaintext is fed into RC4 like before, and we get the following message\_plaintext: **AAAA0300000008060001080006040001000EA66BFB69AC100001000000000000AC1000F0000000000000000000000000000000000000**
2. The CRC plaintext is computed from the message plaintext, giving **6b8fe49d**
3. Concatenating the message plaintext and CRC plaintext, we get the combined plaintext. Feeding the combined plaintext into RC4, we get the combined ciphertext: **EDD4853DA5933C8C915E260537B4148D419181A196DA5500E21039C16B7B456840A418CE1D5FF72BC91FCF4C4BD8372BD5307E982A5E4C1F8C06**
4. Finally, we remove the message ciphertext from the front of the combined ciphertext to get the CRC ciphertext at the end: **4C1F8C06**

This final combined CRC ciphertext is compared with the encrypted ICV shown on Wireshark (**4c1f8c06**). Since they match, our RC4 algorithm is confirmed to be correct.

The corresponding cracked payload and ICV are as stated:

* Payload: **AAAA0300000008060001080006040001000EA66BFB69AC100001000000000000AC1000F0000000000000000000000000000000000000**
* ICV: **6b8fe49d**