Challenge 2 Writeup

Challenge

We have intercepted a signal from a target ground station. An informant has told us that the ground station is transmitting information to its satellite via CCSDS at 882 bps. Please decode the signal and retrieve the flag.

Solution

Step 1: Preliminary Information Gathering

The following Linux command shows that the file is sampled at 44.1 kHZ:

```
$ file challenge.wav
challenge.wav: RIFF (little-endian) data, WAVE audio, Microsoft PCM, 8 bit,
mono 44100 Hz
```

By using GNU Radio, the contestant can inspect the waterfall (Figure 1b) and constellation (Figure 1c) diagram of the waveform. For initial analysis, the contestant should construct something similar to the flow graph in Figure 1a:

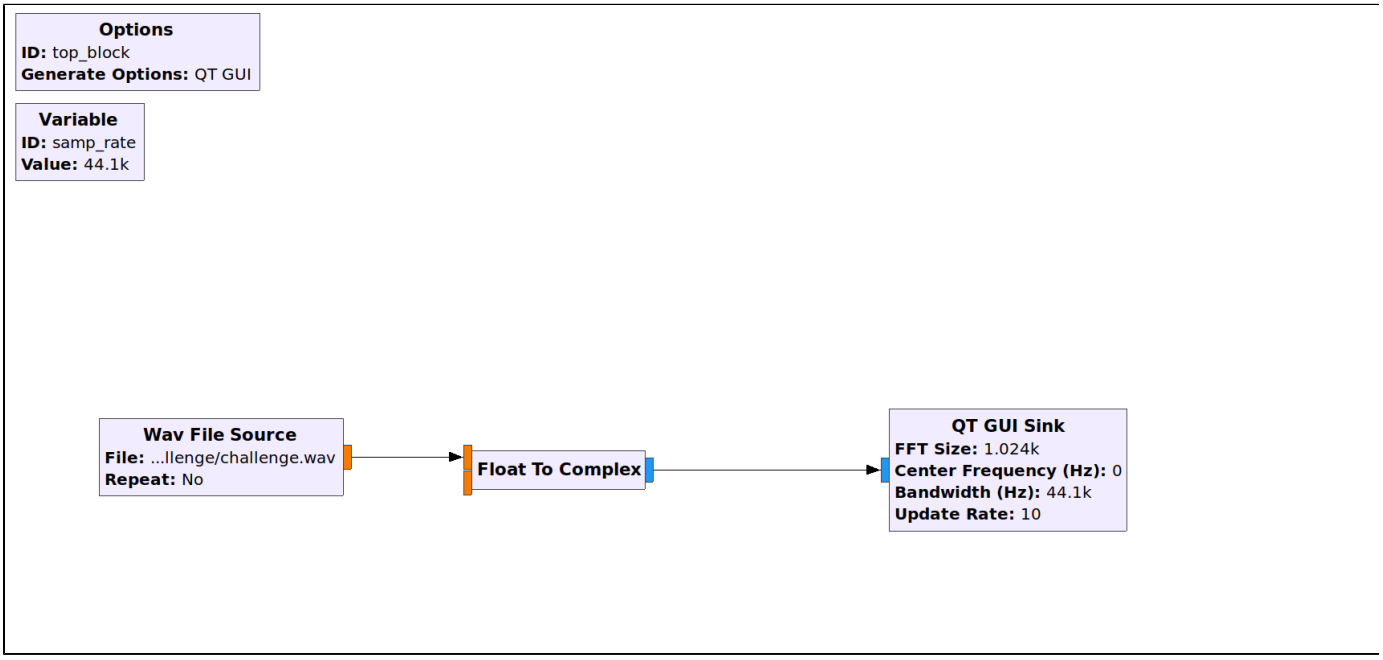


Figure 1a: Flowgraph to inspect the waveform in a waterfall and constellation in GNU Radio

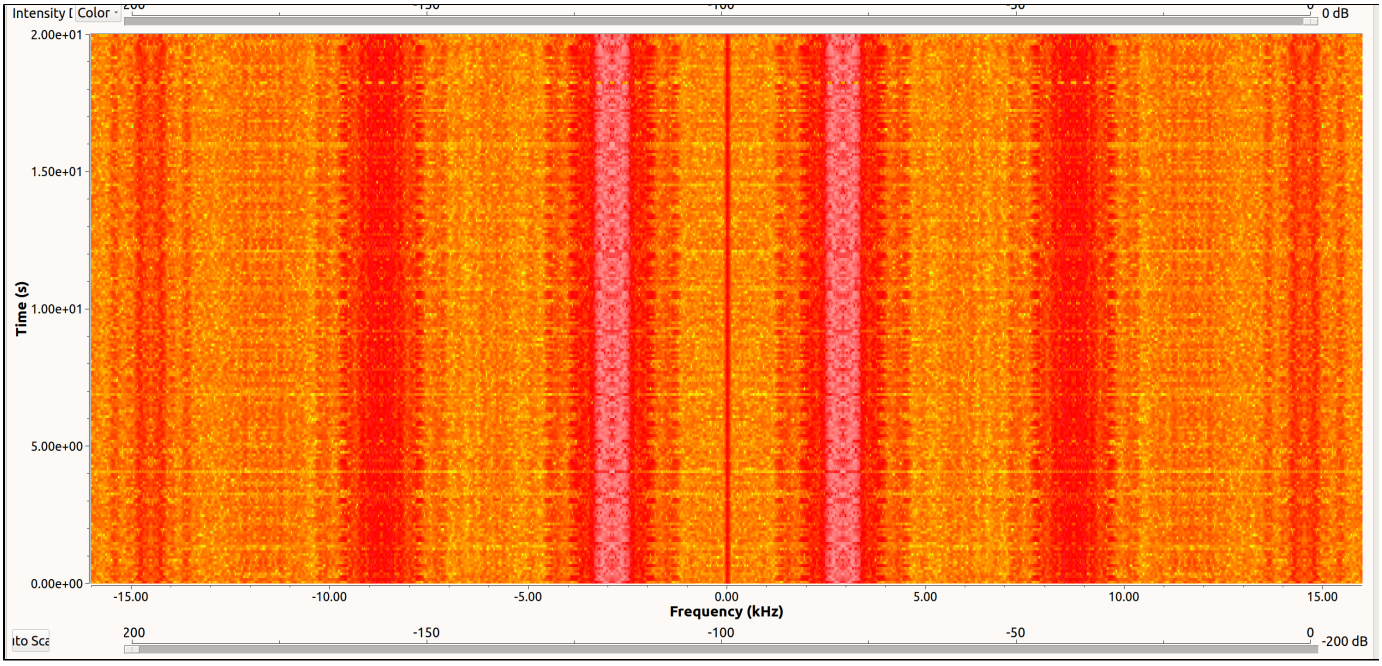


Figure 1b: Waterfall of challenge wav file

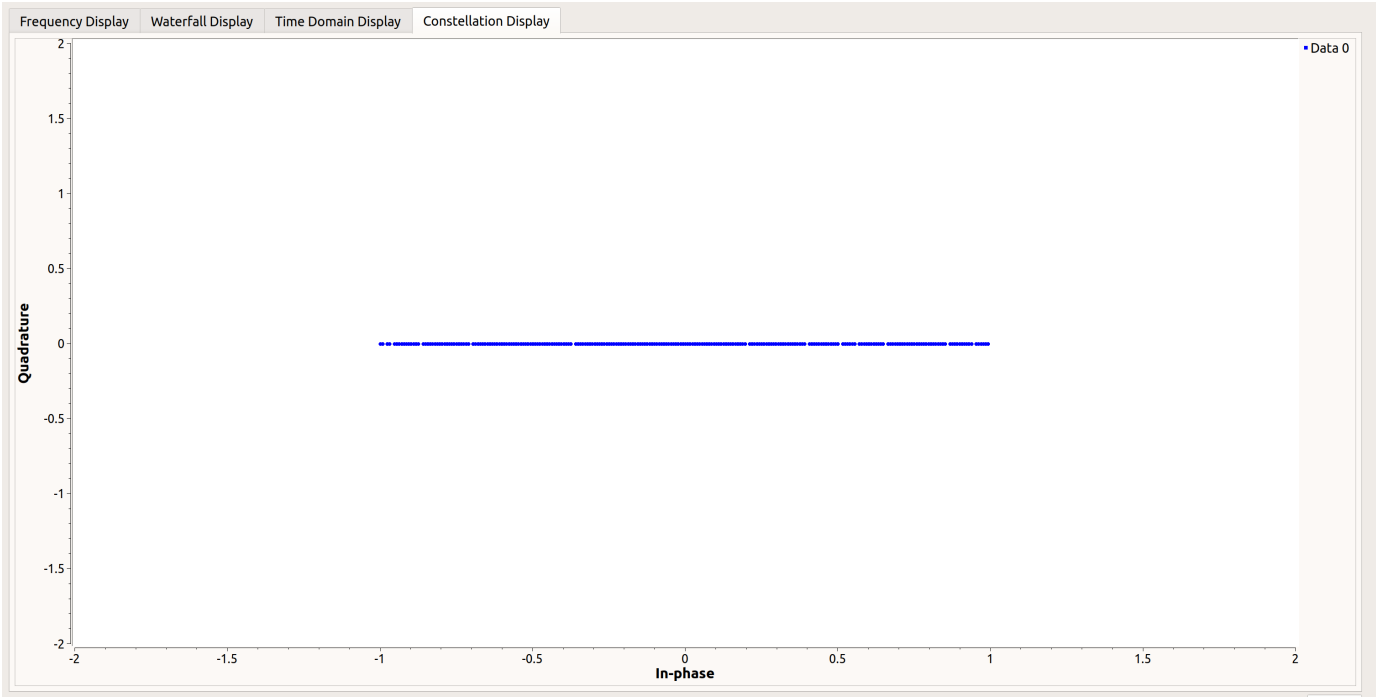


Figure 1c: Constellation of challenge wav file

With this preliminary investigation, the contestant should retrieve the following information:

1. Sampling rate: 44.1 kHz. Should be supported by most internal sound cards of computers.
2. Waterfall: Something interesting is happening at the 4 kHz region. The width of the waterfall at 4 kHz Is around 1.6 kHz, so that should be the Low pass filter cutoff frequency.
3. Constellation: The data is modulated in BPSK format. The line in the constellation diagram shows two distinct phases at 1 and -1 from the base signal.

Step 2: Bitstream Recovery

The first challenge is to bring the signal down to baseband from 4kHz and build a BPSK demodulator. The contestant should build a graph similar to the one in Figure 2. There are different ways to demodulate BPSK signal, the flow graph in Figure 2 is a fairly common one. For those who

don't know how to demodulate signals, they can easily lookup a tutorial to learn demodulation via GNURadio / Matlab. Each GUI box is explained step by step:

1. **Wav File Source**: Direct the challenge.wav in as a floating point source.
2. **Float to complex**: Convert floating point numbers in wav file to complex numbers. A **QT GUI Sink** is attached for debugging purposes.
3. **Multiply**: By multiplying a cosine wave at 4kHz to the input signal source, the signal is brought down to baseband.
4. **Low Pass Filter**: filters out higher frequency noise that are not within 1.6 kHZ bandwidth.
5. **Feed Forward AGC:** Amplifies the signal.
6. **Polyphase Clock Sync**: An algorithm to perform timing recovery. i.e. this block finds the best time to sample a signal in order to reduce inter symbol interference (ISI) and minimize signal-to-noise ratio. (SNR).
7. **Costas Loop**: A carrier recovery module commonly used to synchronize BPSK signals. The variables are kept as default.
8. **CMA Equalizer**: Equalizes I and Q data.
9. **Complex to Real** : Provides the real part of the complex stream
10. **Binary Slicer**: Converts a positive input to 1 and negative input to 0.
11. **Differential decoder**: Each signal bit represents a change in state rather than the signal state itself. Its either present or absent, so the contestant would need to figure this out via trial and error.
12. **FIle sink**: Writes the data stream to a file sink.

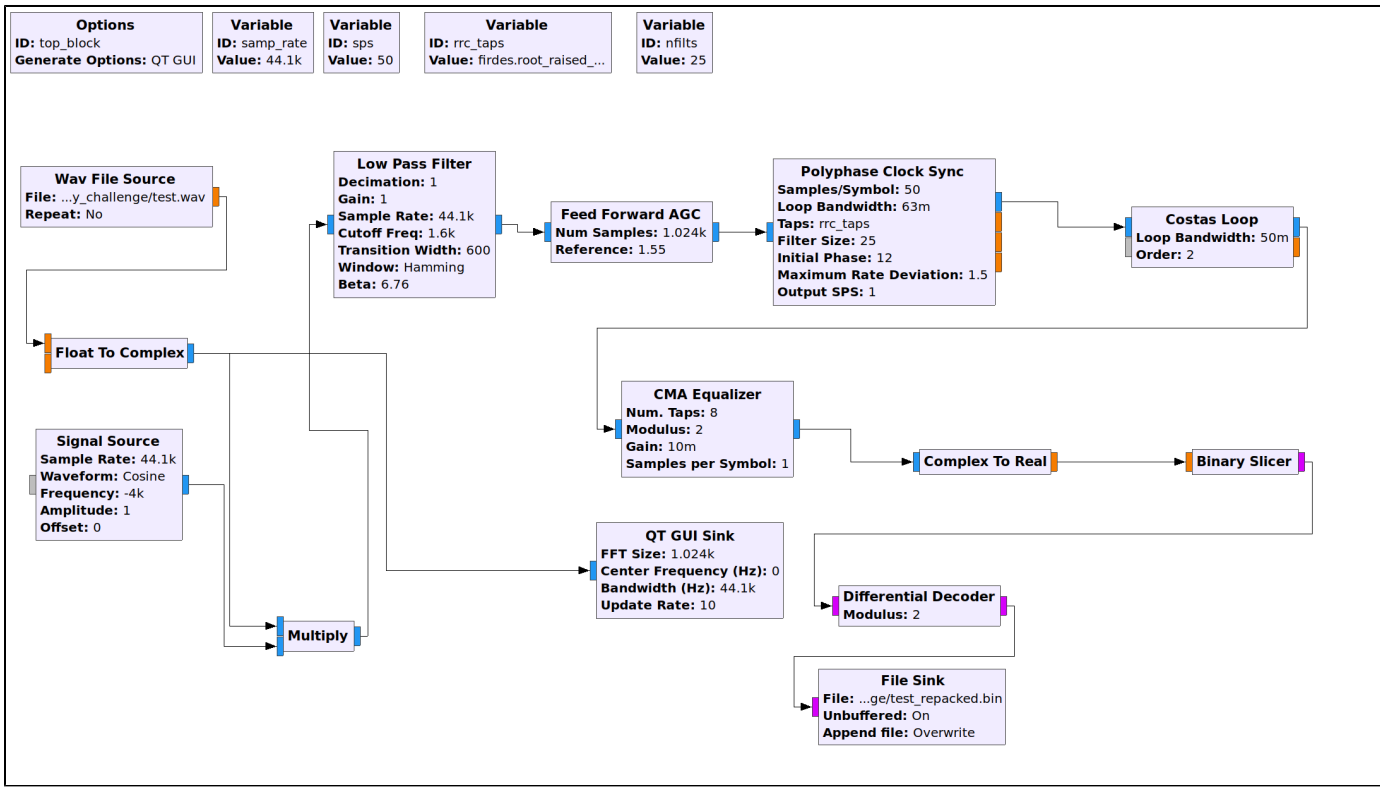


Figure 2: GNURadio Flowgraph to recover the bitstream

Step 3 BitStream Analysis

A screenshot of the output file is shown in Figure 3, which is a stream of 0s and 1s. The LSB of each byte represents a bit, so a byte sequence like '01010001 01010100 01000100 01010001' at offset 0x18c represents a bit stream of '1101 1110 1010 1101', which is the sequence of bytes '0xdead'. The contestant needs to repack the LSB of each byte together to form a cohesive bitstream, retrieving the file shown in Figure 4.

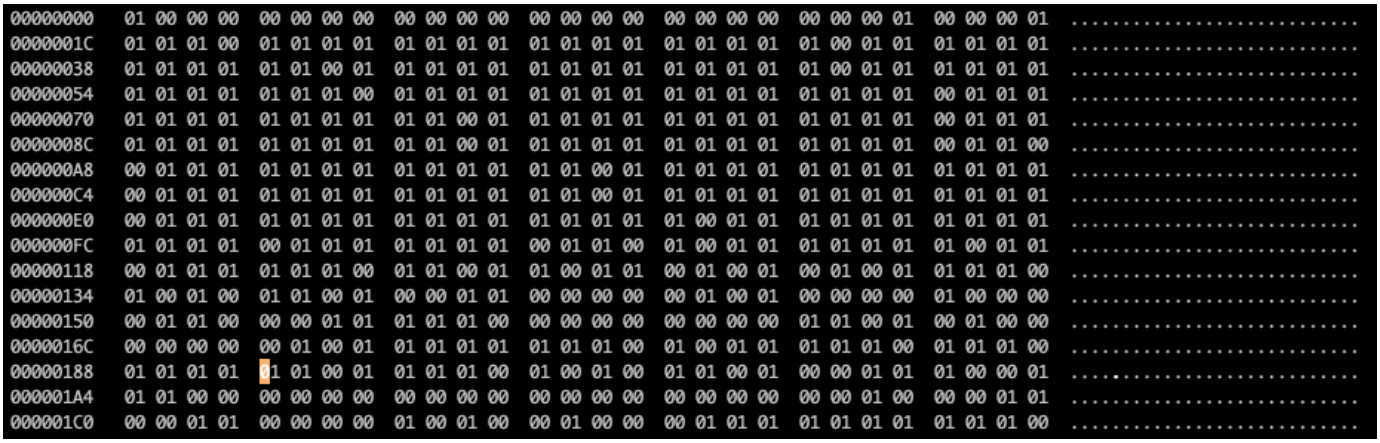


Figure 3: Screenshot of output file from GNURadio flow graph in Figure 2. At offset 0x18c are the bytes 0xdead.

Script to pack the LSB of each byte in recovered bitstream

```
import binascii

final_bin_data = b''
with open("challenge_bitstream_recovered.bin", 'rb') as f:
    binary_data = f.read()
    tmp_byte = 0
    for i in range(len(binary_data)):
        if i % 8 == 0 :
            final_bin_data += '{:02X}'.format(tmp_byte)
            tmp_byte = 0
        if binary_data[i] == b'\x00':
            tmp_byte = tmp_byte << 1
        else:
            tmp_byte = (tmp_byte << 1) + 1

final_bin_data = binascii.unhexlify(final_bin_data)
with open("packed_bistream.bin", 'wb') as f:
    f.write(final_bin_data)
```

```
00000000  00 80 00 01  1E FF FF BF  FD FF FB FF  EF FF F7 FF  DF FF 7F FD  FF F6 7F FD  FF F7 FF DF  ........................
0000001C  FF 7F FF BF  FF 7F 6B FB  7E DB 55 EA  D3 05 08 63  E0 0D 40 5F  EB EE FD EA  D3 9C 00 00  ......k.~.U....c..@_........
00000038  23 30 A4 7F  EB EE FD EA  D7 26 F7 56  E6 42 04 3F  EB EE FD EA  D6 F6 E7 47  26 F6 C2 0F  #0.......&.V.B.?.......G&...
00000054  EB EE FD EA  D7 46 F2 04  D6 16 A6 FF  EB EE FD EA  D7 22 05 46  F6 D0 A4 7F  EB EE FD EA  .....F...........".F........
00000070  D7 26 F7 56  E6 42 04 3F  EB EE FD EA  D6 F6 E7 47  26 F6 C2 0F  EB EE FD EA  D7 46 F2 04  .&.V.B.?.......G&........F..
0000008C  D6 16 A6 FF  EB EE FD EA  D7 22 05 46  F6 D0 A5 4F  EB EE FD EA  D6 16 B6 52  07 96 F7 5F  .........".F...O.......R..._
000000A8  EB EE FD EA  D7 22 07 07  26 F7 46 5F  EB EE FD EA  D6 96 E2 07  06 96 C6 CF  EB EE FD EA  ....."..&.F_................
000000C4  D7 32 06 16  E6 42 07 0F  EB EE FD EA  D7 57 42 07  96 F7 57 2F  EB EE FD EA  D2 06 86 56  .2...B.......WB...W/.......V
000000E0  C6 D6 57 4F  EB EE FD EA  D2 06 F6 E0  A4 77 26 FF  EB EE FD EA  D7 56 E6 42  04 36 F6 EF  ..WO.........w&......V.B.6..
000000FC  EB EE FD EA  D7 47 26 F6  C2 07 46 FF  EB EE FD EA  D2 04 D6 16  A6 F7 22 0F  EB EE FD EA  .....G&...F...........".....
00000118  D5 46 F6 D2  02 87 46 5F  EB EE FD EA  D6 E2 C2 06  E6 96 E6 5F  EB EE FD EA  D2 C2 06 56  .F....F_.........._.......V
00000134  96 76 87 4F  EB EE FD EA  D2 C2 07 36  57 66 56 EF  EB EE FD EA  D2 C2 07 36  97 82 90 AF  .v.O.......6WfV........6....
00000150  EB EE FD EA  D4 36 F6 D6  D6 56 E6 3F  EB EE FD EA  D6 96 E6 72  06 36 F7 5F  EB EE FD EA  .....6...V.?.......r.6._....
0000016C  D6 E7 46 46  F7 76 E2 CF  EB EE FD EA  D2 06 56 E6  76 96 E6 5F  EB EE FD EA  D7 32 06 F6  ..FF.v........V.v._.....2..
00000188  E2 02 86 6F  EB EE FD EA  D6 97 66 52  C2 06 66 FF  EB EE FD EA  D7 57 22 C2  07 46 87 2F  ...o......fR..f......W"..F./
000001A4  EB EE FD EA  D6 56 52 90  A4 36 86 5F  EB EE FD EA  D6 36 B2 06  96 76 E6 9F  EB EE FD EA  .....VR..6._.....6...v......
000001C0  D7 46 96 F6  E2 06 16 EF  EB EE FD EA  D6 42 06 D6  17 92 04 7F  EB EE FD EA  D6 F6 42 77  .F..........B...........Bw
000001DC  32 06 C6 FF  EB EE FD EA  D7 66 52 06  26 52 07 7F  EB EE FD EA  D6 97 46 82  07 96 F7 5F  2........fR.&R........F....._
000001F8  EB EE FD EA  D2 02 87 47  76 F2 C2 0F  EB EE FD EA  D6 F6 E6 52  C2 06 C6 9F  EB EE FD EA  .......Gv..........R........
00000214  D6 67 46 F6  66 62 90 AF  EB EE FD EA  D5 46 86 97  32 06 97 3F  EB EE FD EA  D2 04 77 26  .gF.fb.......F..2..?......w&
00000230  F7 56 E6 4F  EB EE FD EA  D2 04 36 F6  E7 47 26 FF  EB EE FD EA  D6 C2 07 46  F2 04 D6 1F  .V.O......6..G&........F....
0000024C  EB EE FD EA  D6 A6 F7 22  05 46 F6 DF  EB EE FD EA  D0 A5 96 F7  52 77 66 5F  EB EE FD EA  .......".F..........Rwf_....
00000268  D2 07 26 56  16 C6 C7 9F  EB EE FD EA  D2 06 D6 16  46 52 07 4F  EB EE FD EA  D6 86 52 06  ..&V............FR.O......R.
00000284  77 26 16 4F  EB EE FD EA  D6 50 A4 16  E6 42 07 4F  EB EE FD EA  D6 86 52 07  06 17 06 5F  w&.O.....P...B.O......R....._
000002A0  EB EE FD EA  D7 27 32 07  76 16 E7 4F  EB EE FD EA  D2 07 46 F2  06 B6 E6 FF  EB EE FD EA  .....'2.v..O......F.........
000002BC  D7 72 07 76  86 F7 36 5F  EB EE FD EA  D2 07 36 86  97 27 47 3F  EB EE FD EA  D2 07 96 F7  .r.v..6_......6..'G?........
000002D8  52 07 76 5F  EB EE FD EA  D6 17 20 A4  E6 F7 72 0F  EB EE FD EA  D6 97 42 77  32 07 46 9F  R.v_...... ...r.......Bw2.F.
000002F4  EB EE FD EA  D6 D6 52 07  46 F2 06 CF  EB EE FD EA  D6 56 17 66  52 07 46 8F  EB EE FD EA  ......R.F........V.fR.F.....
00000310  D6 52 06 36  17 07 37 5F  EB EE FD EA  D6 C6 52 06  96 62 07 9F  EB EE FD EA  D6 F7 52 06  .R.6..7_......R..b........R.
0000032C  46 17 26 5F  EB EE FD EA  D0 A2 25 46  86 97 32 0F  EB EE FD EA  D6 97 32 04  D6 16 A6 FF  F.&_......%F..2.......2.....
00000348  EB EE FD EA  D7 22 05 46  F6 D2 07 4F  EB EE FD EA  D6 F2 04 77  26 F7 56 EF  EB EE FD EA  ....."..F...O.......w&.V.....
00000364  D6 42 04 36  F6 E7 47 2F  EB EE FD EA  D6 F6 C0 A4  92 76 D2 0F  EB EE FD EA  D7 37 46 57  .B.6..G/.........v.......7FW
00000380  07 06 96 EF  EB EE FD EA  D6 72 07 46  87 26 F7 5F  EB EE FD EA  D6 76 82 07  46 86 52 0F  .........r.F.&._.....v..F.R.
0000039C  EB EE FD EA  D6 46 F6 F7  20 A4 16 EF  EB EE FD EA  D6 42 04 92  76 D2 06 6F  EB EE FD EA  .....F.. ........B..v..o....
```

Figure 4: Screenshot of bytes after concatenation

There are certain patterns in the byte stream, such as '0xdead' and '0xbeef'. Those bytes are in fact the start and tail sequences of CCSDS Telecommand channel and synchronization coding packets (https://public.ccsds.org/Pubs/231x0b3.pdf). **The parity check right now is hardcoded to 0xfe**. The packet structure is as follow:

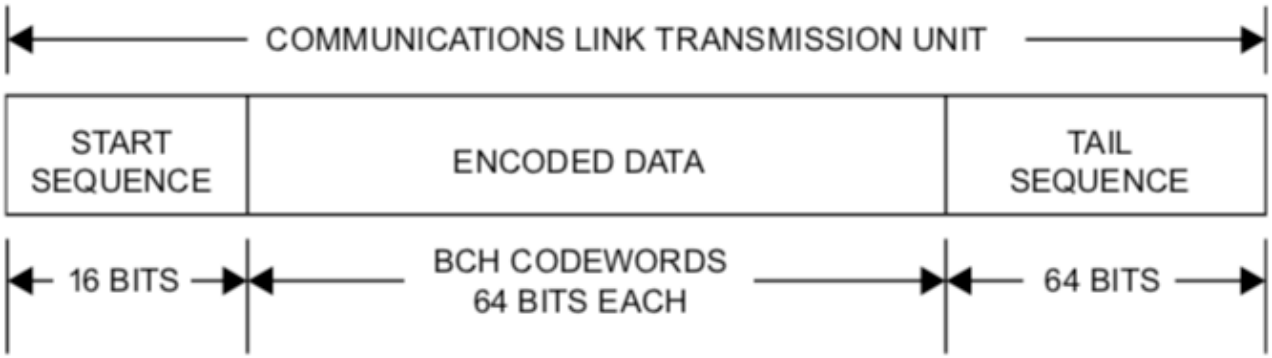


Figure 5a: Packet Structure of each transmission unit.

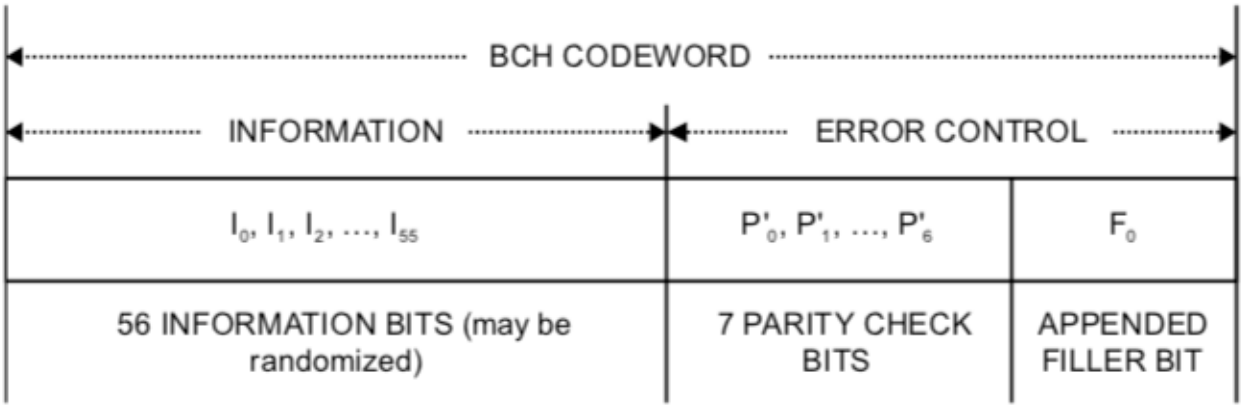


Figure 5b: Packet Structure of Codeword.

When the data is extracted out we get the byte stream shown in Figure 5

Sample script to extract data from channel packets

```
import binascii
import struct

final_bin_data = ''
final_extracted_data = ''
start_seq = 'DEAD'
start_seq_bin = '1101111010101101'
# for the first packet, sometimes the first bit is lost due to
synchronization, use this to identify packets
start_seq_alternative ='0101111010101101'
final_bin_unpacked = ''
with open("challenge_bitstream_recovered.bin", 'rb') as f:
    binary_data = f.read()
    for i in range(len(binary_data)):
        if binary_data[i] == b'\x00':
            final_bin_unpacked += "0"
        else:
            final_bin_unpacked += "1"

# extract data within each physical packet
# each packet is 12 bytes
# byte 0-1: start sequence
# byte 2-8: actual data
# byte 9: parity
# byte 10-11: end sequence
i = 0
while i < len(final_bin_unpacked):
    if final_bin_unpacked[i:i+len(start_seq_bin)] == start_seq_bin or \
            final_bin_unpacked[i:i+len(start_seq_alternative)] ==
start_seq_alternative:
        # start sequence detected, extract 7 data bytes
        extracted_bin_data = final_bin_unpacked[i+len(start_seq_bin):i +
len(start_seq_bin) + 7 * 8]
        # concatenate data bytes to form an actual byte in output binary
file
        for j in range(0, len(extracted_bin_data), 8):
            byte_value = int(extracted_bin_data[j:j+8],2)
            final_extracted_data += "{:02x}".format(byte_value)
        i += 12 * 8
    else:
        i += 1
final_extracted_data = binascii.unhexlify(final_extracted_data)
with open("data_intermediate.bin", 'wb') as f:
    f.write(final_extracted_data)
```

Figure 6. Bytes extracted out from channel packets

```
00000000  30 50 86 3E  00 D4 05 39  C0 00 02 33  0A 47 72 6F  75 6E 64 20  43 6F 6E 74  72 6F 6C 20  0P.>...9...3.Ground Control
0000001C  74 6F 20 4D  61 6A 6F 72  20 54 6F 6D  0A 47 72 6F  75 6E 64 20  43 6F 6E 74  72 6F 6C 20  to Major Tom.Ground Control
00000038  74 6F 20 4D  61 6A 6F 72  20 54 6F 6D  0A 54 61 6B  65 20 79 6F  75 72 20 70  72 6F 74 65  to Major Tom.Take your prote
00000054  69 6E 20 70  69 6C 6C 73  20 61 6E 64  20 70 75 74  20 79 6F 75  72 20 68 65  6C 6D 65 74  in pills and put your helmet
00000070  20 6F 6E 0A  47 72 6F 75  6E 64 20 43  6F 6E 74 72  6F 6C 20 74  6F 20 4D 61  6A 6F 72 20   on.Ground Control to Major
0000008C  54 6F 6D 20  28 74 65 6E  2C 20 6E 69  6E 65 2C 20  65 69 67 68  74 2C 20 73  65 76 65 6E  Tom (ten, nine, eight, seven
000000A8  2C 20 73 69  78 29 0A 43  6F 6D 6D 65  6E 63 69 6E  67 20 63 6F  75 6E 74 64  6F 77 6E 2C  , six).Commencing countdown,
000000C4  20 65 6E 67  69 6E 65 73  20 6F 6E 20  28 66 69 76  65 2C 20 66  6F 75 72 2C  20 74 68 72   engines on (five, four, thr
000000E0  65 65 29 0A  43 68 65 63  6B 20 69 67  6E 69 74 69  6F 6E 20 61  6E 64 20 6D  61 79 20 47  ee).Check ignition and may G
000000FC  6F 64 27 73  20 6C 6F 76  65 20 62 65  20 77 69 74  68 20 79 6F  75 20 28 74  77 6F 2C 20  od's love be with you (two,
00000118  6F 6E 65 2C  20 6C 69 66  74 6F 66 66  29 0A 54 68  69 73 20 69  73 20 47 72  6F 75 6E 64  one, liftoff).This is Ground
00000134  20 43 6F 6E  74 72 6F 6C  20 74 6F 20  4D 61 6A 6F  72 20 54 6F  6D 0A 59 6F  75 27 76 65   Control to Major Tom.You've
00000150  20 72 65 61  6C 6C 79 20  6D 61 64 65  20 74 68 65  20 67 72 61  64 65 0A 41  6E 64 20 74   really made the grade.And t
0000016C  68 65 20 70  61 70 65 72  73 20 77 61  6E 74 20 74  6F 20 6B 6E  6F 77 20 77  68 6F 73 65  he papers want to know whose
00000188  20 73 68 69  72 74 73 20  79 6F 75 20  77 65 61 72  0A 4E 6F 77  20 69 74 27  73 20 74 69   shirts you wear.Now it's ti
000001A4  6D 65 20 74  6F 20 6C 65  61 76 65 20  74 68 65 20  63 61 70 73  75 6C 65 20  69 66 20 79  me to leave the capsule if y
000001C0  6F 75 20 64  61 72 65 0A  22 54 68 69  73 20 69 73  20 4D 61 6A  6F 72 20 54  6F 6D 20 74  ou dare."This is Major Tom t
000001DC  6F 20 47 72  6F 75 6E 64  20 43 6F 6E  74 72 6F 6C  0A 49 27 6D  20 73 74 65  70 70 69 6E  o Ground Control.I'm steppin
000001F8  67 20 74 68  72 6F 75 67  68 20 74 68  65 20 64 6F  6F 72 0A 41  6E 64 20 49  27 6D 20 66  g through the door.And I'm f
00000214  6C 6F 61 74  69 6E 67 20  69 6E 20 61  20 6D 6F 73  74 20 70 65  63 75 6C 69  61 72 20 77  loating in a most peculiar w
00000230  61 79 0A 41  6E 64 20 74  68 65 20 73  74 61 72 55  55 55 55 55  55 30 50 87  FF 16 14 C1  ay.And the starUUUUU0P.....
0000024C  82 4C 81 55  97 32 42 F7  29 82 5D 75  D0 92 7F ED  77 A5 E9 42  3C 0E 10 88  76 7F AA 49  .L.U.2B.).]u....w..B<...v..I
00000268  2B 89 35 4B  06 5C A2 8C  1E 27 80 CC  15 B7 C9 2B  90 D2 86 05  B1 F5 B1 6A  2B 5D D6 15  +.5K.\...'.....+.......j+]..
00000284  EC 3F 54 F2  87 37 D6 63  36 3E C4 5C  1B 5A 2C 0E  09 69 FB EE  C5 87 5F C8  AB 12 7D 03  .?T..7.c6>.\.Z,..i...._...}.
000002A0  56 CC 9D A2  7B A3 8B 95  39 C1 27 40  5C 35 F5 93  22 A8 D7 9E  F0 4C A8 7E  D9 B5 36 5E  V...{...9.'@\5.."....L.~..6^
000002BC  54 5B 31 E0  11 27 4B 7C  7E 72 E6 EE  0A 1E 31 0E  7E 86 68 25  63 9E 5A 83  9D 20 6A 27  T[1..'K|~r....1.~.h%c.Z.. j'
000002D8  9B DA 7C F1  37 B4 F3 2F  C4 95 3F 5B  50 F9 70 DD  AC 94 F3 7E  CA F6 27 26  A0 F0 8F 2C  ..|.7../..?[P.p....~..'&...,
000002F4  26 AD 02 16  BF E1 8C 19  66 33 C2 40  F9 A8 FA 39  E9 98 F8 08  69 E4 0E EB  69 30 CE 23  &.......f3.@...9....i...i0.#
00000310  DF 64 D2 FF  7F A4 7C AB  96 3B 2D 6F  2B A9 5B 79  01 64 50 80  8B A2 DF E7  D9 A6 DC A8  .d....|..;-o+.[y.dP.........
0000032C  80 04 19 FC  FC C1 09 1A  65 9C 13 48  36 0F 79 0E  6D 60 97 61  71 BA 63 CF  65 9A 9B FE  ........e..H6.y.m`.aq.c.e...
00000348  49 EC 9A 83  7D CE BA C6  21 AE C3 C2  B6 E8 68 93  7A 0C DD ED  8B 30 E1 97  D7 6D D9 42  I...}...!.....h.z....0...m.B
00000364  5A 79 1A 84  FE FB AA A9  08 90 BC CD  04 63 A9 70  8E D2 4C DC  7B BD 54 F8  71 17 2E 3B  Zy...........c.p..L.{.T.q..;
```

Figure 7: Output of data_intermediate.bin

At the beginning, we can see the lyrics of Space Oddity, hinting that the contestant is going in the right direction. Let's look at the first few bytes extracted from the byte stream. The first 6 bytes should be representative of the header of a Telecommand datalink packet from the CCSDS standard (Section 4 of https://public.ccsds.org/Pubs/232x0b3.pdf). The analysis of the first 6 bytes is shown below

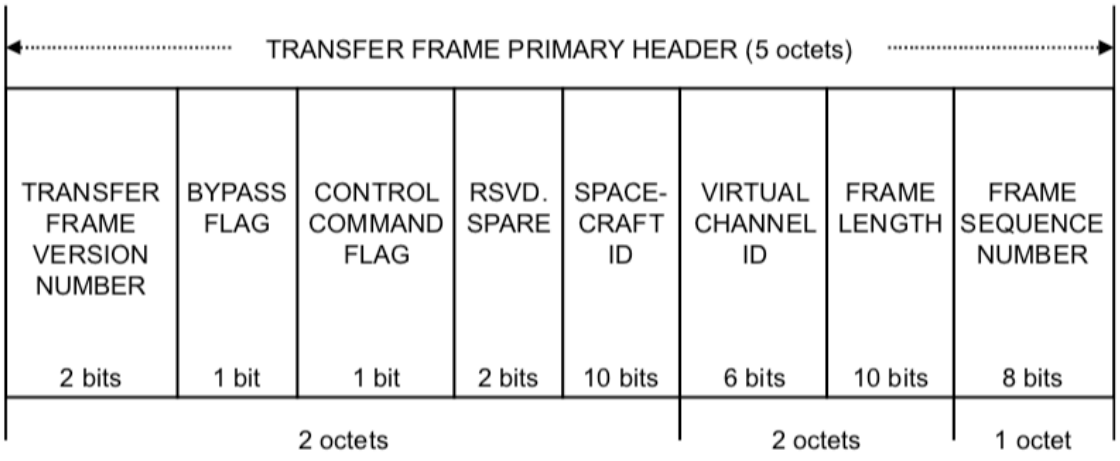


Figure 8a: Frame header of datalink layer

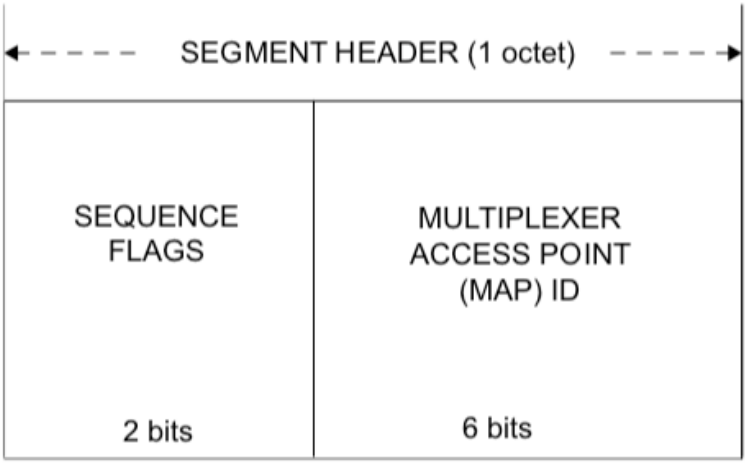


Figure 8b: Segment header of datalink layer

```
The first 6 bytes of data intermediate: 0x30 0x50 0x86 0x3e 0x00 0xd4
(0b00110000 0b01010000 0b10000110 0b00111110 0b00000000 0b11010100)
tx frame version : 0b00
bypass flag: 0b1 (true)
control command flag: 0b1 (Telecommand flag set)
reserved spare: 0b00
spacecraft id:  0b000001010000 (80)
virtual channel ID: 0b10 0001 (33)
frame_length: 0b1000111110 (574 bytes)
frame_seq_num: 0b00000000 (0th frame)
seq_flags: 0b11 (unsegmented)
map_id: 0b010100 (20)
```

The datalink packets are actually out of order in the challenge. With this in mind, the contestant needs to extract each datalink packet out and reorder the datalink packets. The packet data can then be concatenated together

extract and reorder datalink packets

```
# extract data from datalink packet
j = 0
tc_datalink_packet_list = []
while j < len(final_extracted_data):
    if final_extracted_data[j:j+4] == "3050":
        datalink_header = final_extracted_data[j:j+12]
        frame_length = int(datalink_header[5:8], 16) % 1024     # extract
10 bits from the 3rd byte
        frame_seq_num = int(datalink_header[8:10], 16)          # extract
16 bits from the 4th byte
        datalink_packet_data =
final_extracted_data[j+12:j+(frame_length+1)*2]  # the size of packet is
frame_length+1
        j += (frame_length+1)*2
        tc_datalink_packet_list.append((frame_seq_num,
datalink_packet_data, len(datalink_packet_data)))
    else:
        j += 1

# reorder the packets
packet_list = [None] * 27
for p in tc_datalink_packet_list:
    packet_list[p[0]] = p[1]

reordered_data = ''.join(packet_list)

with open("./extracted_datalink_data.bin", 'wb') as f:
    f.write(reordered_data)
```

```
00000000  05 39 C0 00  02 33 0A 47  72 6F 75 6E  64 20 43 6F  6E 74 72 6F  6C 20 74 6F  20 4D 61 6A  .9...3.Ground Control to Maj
0000001C  6F 72 20 54  6F 6D 0A 47  72 6F 75 6E  64 20 43 6F  6E 74 72 6F  6C 20 74 6F  20 4D 61 6A  or Tom.Ground Control to Maj
00000038  6F 72 20 54  6F 6D 0A 54  61 6B 65 20  79 6F 75 72  20 70 72 6F  74 65 69 6E  20 70 69 6C  or Tom.Take your protein pil
00000054  6C 73 20 61  6E 64 20 70  75 74 20 79  6F 75 72 20  68 65 6C 6D  65 74 20 6F  6E 0A 47 72  ls and put your helmet on.Gr
00000070  6F 75 6E 64  20 43 6F 6E  74 72 6F 6C  20 74 6F 20  4D 61 6A 6F  72 20 54 6F  6D 20 28 74  ound Control to Major Tom (t
0000008C  65 6E 2C 20  6E 69 6E 65  2C 20 65 69  67 68 74 2C  20 73 65 76  65 6E 2C 20  73 69 78 29  en, nine, eight, seven, six)
000000A8  0A 43 6F 6D  6D 65 6E 63  69 6E 67 20  63 6F 75 6E  74 64 6F 77  6E 2C 20 65  6E 67 69 6E  .Commencing countdown, engin
000000C4  65 73 20 6F  6E 20 28 66  69 76 65 2C  20 66 6F 75  72 2C 20 74  68 72 65 65  29 0A 43 68  es on (five, four, three).Ch
000000E0  65 63 6B 20  69 67 6E 69  74 69 6F 6E  20 61 6E 64  20 6D 61 79  20 47 6F 64  27 73 20 6C  eck ignition and may God's l
000000FC  6F 76 65 20  62 65 20 77  69 74 68 20  79 6F 75 20  28 74 77 6F  2C 20 6F 6E  65 2C 20 6C  ove be with you (two, one, l
00000118  69 66 74 6F  66 66 29 0A  54 68 69 73  20 69 73 20  47 72 6F 75  6E 64 20 43  6F 6E 74 72  iftoff).This is Ground Contr
00000134  6F 6C 20 74  6F 20 4D 61  6A 6F 72 20  54 6F 6D 0A  59 6F 75 27  76 65 20 72  65 61 6C 6C  ol to Major Tom.You've reall
00000150  79 20 6D 61  64 65 20 74  68 65 20 67  72 61 64 65  0A 41 6E 64  20 74 68 65  20 70 61 70  y made the grade.And the pap
0000016C  65 72 73 20  77 61 6E 74  20 74 6F 20  6B 6E 6F 77  20 77 68 6F  73 65 20 73  68 69 72 74  ers want to know whose shirt
00000188  73 20 79 6F  75 20 77 65  61 72 0A 4E  6F 77 20 69  74 27 73 20  74 69 6D 65  20 74 6F 20  s you wear.Now it's time to
000001A4  6C 65 61 76  65 20 74 68  65 20 63 61  70 73 75 6C  65 20 69 66  20 79 6F 75  20 64 61 72  leave the capsule if you dar
000001C0  65 0A 22 54  68 69 73 20  69 73 20 4D  61 6A 6F 72  20 54 6F 6D  20 74 6F 20  47 72 6F 75  e."This is Major Tom to Grou
000001DC  6E 64 20 43  6F 6E 74 72  6F 6C 0A 49  27 6D 20 73  74 65 70 70  69 6E 67 20  74 68 72 6F  nd Control.I'm stepping thro
000001F8  75 67 68 20  74 68 65 20  64 6F 6F 72  0A 41 6E 64  20 49 27 6D  20 66 6C 6F  61 74 69 6E  ugh the door.And I'm floatin
00000214  67 20 69 6E  20 61 20 6D  6F 73 74 20  70 65 63 75  6C 69 61 72  20 77 61 79  0A 41 6E 64  g in a most peculiar way.And
00000230  20 74 68 65  20 73 74 61  72 05 39 C0  02 62 37 00  00 00 0C 6A  50 20 20 0D  0A 87 0A 00   the star.9..b7....jP  .....
0000024C  00 00 14 66  74 79 70 6A  70 32 20 00  00 00 00 6A  70 32 20 00  00 0C 99 6A  70 32 68 00  ...ftypjp2 ....jp2 ....jp2h.
00000268  00 00 16 69  68 64 72 00  00 01 18 00  00 01 86 00  04 07 07 01  00 00 00 0C  53 63 6F 6C  ...ihdr.................Scol
00000284  72 02 00 00  00 00 0C 48  4C 69 6E 6F  02 10 00 00  6D 6E 74 72  52 47 42 20  58 59 5A 20  r.......HLino....mntrRGB XYZ
000002A0  07 CE 00 02  00 09 00 06  00 31 00 00  61 63 73 70  4D 53 46 54  00 00 00 00  49 45 43 20  .........1..acspMSFT....IEC
000002BC  73 52 47 42  00 00 00 00  00 00 00 00  00 00 00 00  00 00 F6 D6  00 01 00 00  00 00 D3 2D  sRGB.......................-
000002D8  48 50 20 20  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  HP  ........................
000002F4  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 11  63 70 72 74  ........................cprt
00000310  00 00 01 50  00 00 00 33  64 65 73 63  00 00 01 84  00 00 00 6C  77 74 70 74  00 00 01 F0  ...P...3desc.......lwtpt....
0000032C  00 00 00 14  62 6B 70 74  00 00 02 04  00 00 00 14  72 58 59 5A  00 00 02 18  00 00 00 14  ....bkpt........rXYZ........
00000348  67 58 59 5A  00 00 02 2C  00 00 00 14  62 58 59 5A  00 00 02 40  00 00 00 14  64 6D 6E 64  gXYZ...,....bXYZ...@....dmnd
00000364  00 00 02 54  00 00 00 70  64 6D 64 64  00 00 02 C4  00 00 00 88  76 75 65 64  00 00 03 4C  ...T...pdmdd........vued...L
00000380  00 00 00 86  76 69 65 77  00 00 03 D4  00 00 00 24  6C 75 6D 69  00 00 03 F8  00 00 00 14  ....view.......$lumi........
0000039C  6D 65 61 73  00 00 04 0C  00 00 00 24  74 65 63 68  00 00 04 30  00 00 00 0C  72 54 52 43  meas.......$tech...0....rTRC
```

Figure 9: Bytestream in extracted_datalink_data.bin

The last layer of encapsulation is the space packet protocol (https://public.ccsds.org/Pubs/133x0b1c2.pdf). The first two bytes of the header uniquely identifies the stream of bytes, which in this case is 0x05c9. The least significant 14 bits of the second 2 bytes is the sequence counter, and the last 2 bytes is the datalength. The contestant can then reconstruct individual space packets and store them in individual files

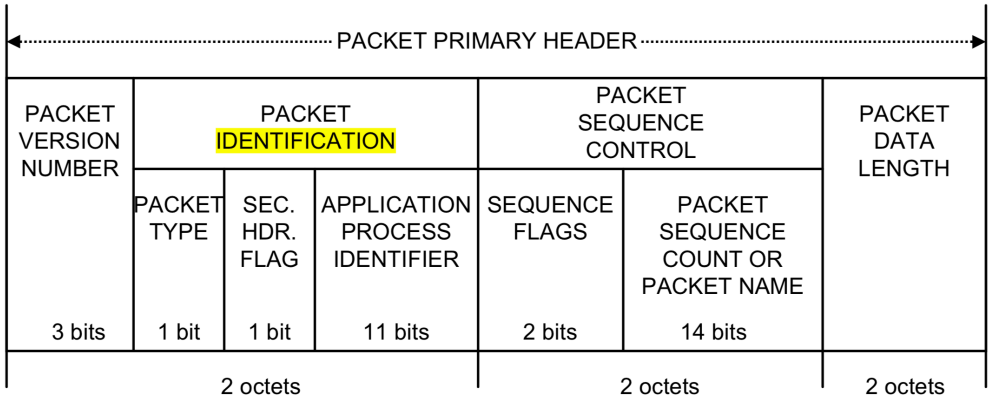


Figure 4-2: Packet Primary Header

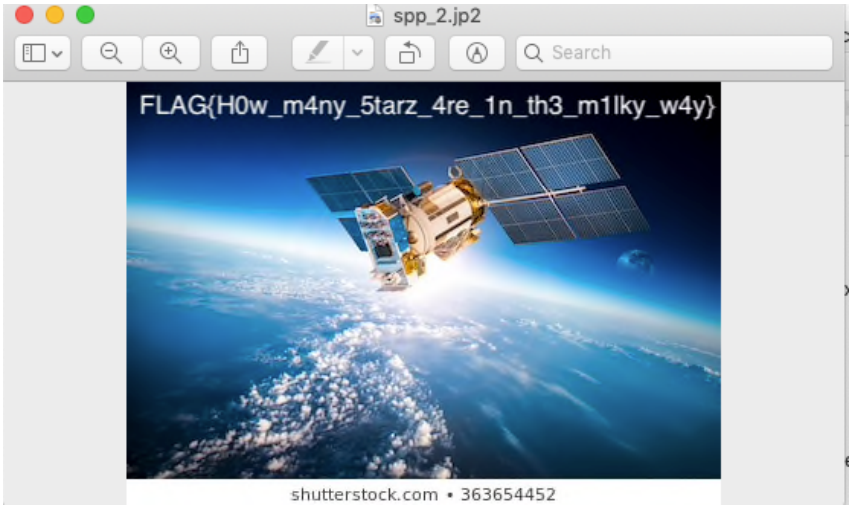
Figure 10: Space packet header structure

Reconstruct space packets

```
# retrieve the original space packets
k = 0
reordered_data = binascii.unhexlify(reordered_data)
space_packet_prefix = "spp_{}.bin"
stream_id = b'\x05\x39'
while k < len(reordered_data):
    if reordered_data[k:k+2] == stream_id:
        spp_packet_data_length =
int(binascii.hexlify(reordered_data[k+4:k+6]), 16)
        space_packet_index =
int(binascii.hexlify(reordered_data[k+2:k+4]),16) % (2 ** 14)
        with open(space_packet_prefix.format(space_packet_index), 'wb') as
f:
            f.write(reordered_data[k+6:k+spp_packet_data_length+6])
        k += spp_packet_data_length + 6
    else:
        k += 1
```

In the sample code, each space packet is saved as an individual file. When the contestant runs file on the extracted binaries, one of them would be a picture:

```
$ file spp*
spp_0.bin: ASCII text
spp_1.bin: ASCII text, with overstriking
spp_2.bin: JPEG 2000 Part 1 (JP2)
```



The flag is in the picture.