卫星信号分析与解密 Writeup

介绍

该题目是对一段未知的卫星通信信号进行深入分析，确定其信号特性、调制方式，并尝试解调和解密，最终恢复传输内容。以下将介绍解题的主要步骤和方法。

具体可分为以下步骤：

- 加载并解析 IQ 信号数据
- 时域分析
- 频域分析
- 调制方式识别
- QPSK 解调
- 异或（XOR）解密
- 比特流转换为 ASCII 文本

详细步骤

1. 加载并解析 IQ 信号数据

代码片段：

```
def load_iq_signal_from_iq_file(filename='satellite_signal.iq'):
    data = np.fromfile(filename, dtype=np.float32)
    I = data[0::2]
    Q = data[1::2]
    iq_signal = I + 1j * Q
    return iq_signal
```

2. 时域分析

代码片段：

```
def plot_time_domain(iq_signal):
    plt.figure(figsize=(12, 6))
```

```
    plt.plot(iq_signal.real[:1000], label='I')
    plt.plot(iq_signal.imag[:1000], label='Q')
    plt.legend()
    plt.show()
```

说明：

绘制 IQ 信号前 1000 个采样点的 I 和 Q 分量。

观察信号的幅度和相位变化，初步了解信号特性。

3. 频域分析

代码片段：

```
def plot_frequency_spectrum(iq_signal):
    N = len(iq_signal)
    yf = fft(iq_signal)
    xf = fftfreq(N, 1 / SAMPLE_RATE)
    plt.plot(xf[:N//2], 20 * np.log10(np.abs(yf[:N//2])))
    plt.show()
```

说明：

- 对 IQ 信号进行快速傅里叶变换（FFT），获取频谱信息。
- 绘制频谱图，确定信号的频率成分，估计载波频率和采样率。

4. 调制方式识别

分析：

- 通过频谱图观察，信号主要集中在特定频率，且频谱呈对称性，初步判断为相位调制信号。
- 结合提示，猜测可能采用了 **QPSK（四相移键控）** 调制方式。

5. QPSK 解调

代码片段：

```
def qpsk_demodulate(iq_signal):
    total_symbols = len(iq_signal) // SAMPLES_PER_SYMBOL
    iq_signal = iq_signal[:total_symbols * SAMPLES_PER_SYMBOL]
    iq_signal = iq_signal.reshape((total_symbols, SAMPLES_PER_SYMBOL))
```

```
    symbol_samples = iq_signal[:, SAMPLES_PER_SYMBOL // 2]
    phases = np.angle(symbol_samples)
    bits = []
    for phi in phases:
        phi = phi % (2 * np.pi)
        if 0 <= phi < np.pi/2:
            bits.append('00')
        elif np.pi/2 <= phi < np.pi:
            bits.append('01')
        elif np.pi <= phi < 3*np.pi/2:
            bits.append('11')
        else:
            bits.append('10')
    encrypted_bits = ''.join(bits)
    return encrypted_bits
```

说明:

将信号按照每个符号的采样点数进行分段。

对每个符号，提取中间采样点，计算其相位。

根据相位所属的象限，判决对应的比特序列。

第一区间（0 到 π/2）：'00'

第二区间（π/2 到 π）：'01'

第三区间（π 到 3π/2）：'11'

第四区间（3π/2 到 2π）：'10'

6. 异或（XOR）解密

代码片段：

```
def xor_decrypt(encrypted_bits, key=KEY):
    key_length = len(key)
    key_repeated = (key * ((len(encrypted_bits) // key_length) + 1))[:len(encrypted_bits)]
    decrypted_bits = ''.join(
        str(int(b) ^ int(k)) for b, k in zip(encrypted_bits, key_repeated)
```

```
    )
    return decrypted_bits
```

7. 比特流转换为 ASCII 文本

代码片段：

```
def bits_to_ascii(bit_stream):
    bytes_list = []
    for i in range(0, len(bit_stream), 8):
        byte = bit_stream[i:i+8]
        if len(byte) < 8:
            byte = byte.ljust(8, '0')
        bytes_list.append(int(byte, 2))
    byte_array = bytearray(bytes_list)
    message = byte_array.decode('utf-8', errors='replace')
    return message
```

说明：

将解密后的比特流每 8 位分割为一个字节。

将字节序列转换为字符，假设使用 UTF-8 编码。

处理可能的解码错误，确保得到可读的文本。

最终结果

代码片段：

```
def main():
    iq_signal = load_iq_signal_from_iq_file('satellite_signal.iq')
    # 可选的时域和频域分析
    # plot_time_domain(iq_signal)
    # plot_frequency_spectrum(iq_signal)
    encrypted_bits = qpsk_demodulate(iq_signal)
    decrypted_bits = xor_decrypt(encrypted_bits, KEY)
    message = bits_to_ascii(decrypted_bits)
    if EXPECTED_MESSAGE in message:
```

```
            print(f"成功还原出原始信息: '{message}'")
    else:
            print("未能成功还原出预期的原始信息。")
            print(f"还原的信息内容为: '{message}'")
```

运行结果示例：

成功加载信号文件: satellite_signal.iq

解调后的比特流 (XXXX bits)

解密后的比特流 (XXXX bits)

还原的信息: flag{N91°00'00" E181°00'00"}

成功还原出原始信息: 'flag{N91°00'00" E181°00'00"}'

总结

通过对未知卫星信号的加载、分析、解调和解密，我们成功恢复了原始传输内容。本次解题过程主要涉及以下关键技术：

- **信号加载与预处理**：正确读取 IQ 数据，准备后续分析。
- **时域与频域分析**：了解信号特性，为调制方式识别提供依据。
- **调制方式识别**：通过相位分析，确定使用了 QPSK 调制。
- **信号解调**：采用相位判决方法，将信号转换为比特流。
- **加密解密处理**：使用已知的 XOR 密钥，成功解密比特流。
- **数据还原**：将二进制比特流转换为可读的文本信息。