Assignment P3

1 Introduction

In the third assignment, we implement some security features to lownet. Recall that we have used ESP-Now as an open radio channel that anyone can listen to. Hence, **all the communication until now has been insecure**. Anyone can tap in the network, and inject malicious packets.

1.1 High-level functionalities

What is provided: The starting point for this assignment includes the following functionalities and specifications:

1. Specification for the new packet format v2 that encapsulates the original short packets in encrypted form.

Your tasks:

The tasks have been divided into two milestones. The first milestone is a soft target for the first week. The second milestone corresponds to fully functioning submission.

Milestone I:

1. Implement AES encryption to Lownet.
    - Encryption key can be set manually via the serial interface

After Milestone I, the lownet protocol can be considered to be sufficiently secure for experimental setups (right?).

Milestone II:

1. Implement the digital signature verification using the provided sample message.
2. Implement support for the multi-frame signed packets.
3. Implement the new command protocol that obsoletes the inseruce time protocol (so that time update is taken into account only after the signature and sequence numbers have been validated).

In Milestone I you need to integrate RSA and SHA-256 cryptographics algorithms to Lownet.

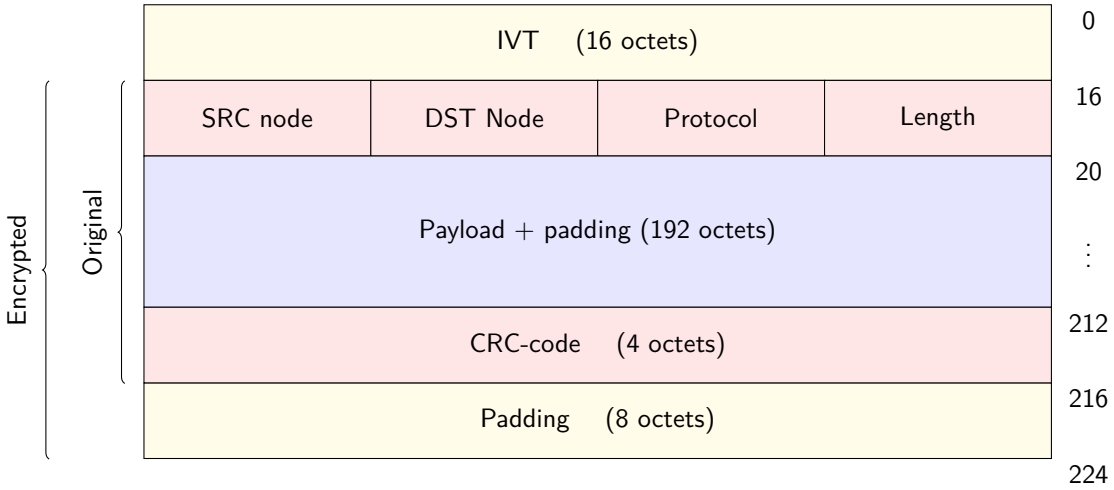
2 Specification

In this section, we specify the packet format and other necessary details for each task.

2.1 Packet format

As said, we use ESP-Now, but equally well the wireless networking could be LoRa, IP, or some mix of them. For now we assume that all devices are in close contact so that radio transmissions reach all nodes. Thus no routing functionality is needed, and the setup is essentially Layer-2 networking.

The packet format v2 is as follows:



Note that:

- Each row is 32 bits or its multiple.
- Yellow blocks, IVT and final padding, encapsulate the original packet v1.
- Both IVT and final padding should be random numbers (for each packet).
- Once the encryption is removed, the v1 packet can be passed to any existing code as it is.
- `Source` and `Destination` node id's are one byte each (`uint8_t`).
  For broadcast messages the destination field is set to `0xFF`.
  Note that you can acquire the node id of your device via the system call `lownet_get_device_id(void)`.
- `Protocol` defines the application. These are defined in the provided include files.
- `Length` corresponds to the length of actual data in the payload.
- `CRC` computation is based on the given 24-bit CRC generator polynomial,

  $$G(x) = x^{24} + x^{10} + x^6 + x^5 + x + 1,$$

  so that the 8 MSBs are zero (32 bits reserved for CRC code in the packet).

2.2 Operation

When encryption is enabled, the lownet device ignores the original (insecure) packets and processes only the encypted packets.

Upon receiving an encrypted packet, the network layer will pass it to `lownet_decrypt()` function, which returns the plaintext packet in the provided buffer.

Upon transmission the reverse occurs. The network layer will call `lownet_encrypt()` function which shall carry out the encryption with the current key.

Your task (in Milestone I) is to implement the two above functions.

2.3 AES-256

The encryption algorithm we employ is the standard AES-256 in CBC mode. See the course lecture notes and slides for more information. Each packet will start with a 16-byte long IVT that is used to initilize the encryption/decryption.

The assignment package will contain some test strings (input output pairs).

2.4 Digital Signature: RSA and SHA-256

The protocol for *signed packets* will be described next. We reserve two bits from the protocol field to indicate that the current frame is part of the signed frame:

00	normal frame (no signature)
01	normal frame, signature will follow
10	the first part of the signature
11	the second part of the signature

The signature frames have the following format for the payload:

hash of the public key	32 octets
hash of the message	32 octets
half of the signature	128 octets

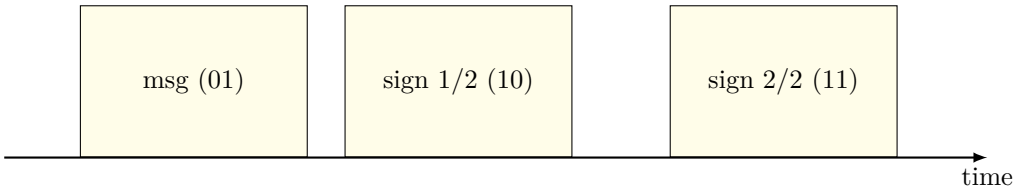
The network layer will buffer the packet that needs a matching signature for up to 10 seconds. If no valid signature is received by then, the packet is discarded.

Received signature frames are associated with the message frame using the SHA-256 hash value, computed for the 200 bytes long message.

Similarly, each node shall keep the 2048-bit public RSA key the master node has published (also in the header files).

```
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAxG9VF9wuocepQnwBkxUb
4YxCo1NJ1MAKAGoaK2csfPABSRkjlESev42rFVzejGtOp2pxKcyihDXVe1BEzD0q
HXxEgtkRyO/bJNhGxoMmWTbikO3BmIMIO9zIk3leaNtyy49U27CKDgUHOPp6zd3c
dgD3nE4fIE7tU3mCJ4xh5xMHeyoqa/MV3EkE9VDV2vCTP3KyKDFObYqig6XWydeQ
CPmSAr0rRYiriguOvQGGxPeaCWPaUAG+t2W7ydpeju+Dkzl6NHm0q9JdLfpg8zje
BgLekdFxyM4jAK2hCX+vswUrYqbm5m9rptxQUuSYpk27Ew7uWRaomAWWeMLIg+zt
rwIDAQAB
-----END PUBLIC KEY-----
```

In principle, the system can support more than one authority, each with their own published key. The public key hash in the signature frames is used to identify the key used for signing each message.



2.5 Time protocol

Time protocol will deprecated due security concerns. It is replaced with the command protocol described next. Your device **should not** accept any time updated via the old protocol!

2.6 Command protocol

The new command protocol has the following features:

1. all packets shall be signed
2. all packets shall have a strictly increasing sequence number

64-bit sequence number	(8 octets)
command	(1 octet)
undefined (reserved)	(3 octets)
(command specific data)	(0 - 180 octets)

A command is valid iff:

i) signature is valid
ii) no valid command with equal or higher sequence number has been received.

Two commands will be implemented at this stage:

- **Time:** (`0x01`)
  The payload shall have the time (`lownet_time_t`, 5 octets).
- **Test:** (`0x02`)
  The payload shall contain arbitrary content that should be copied as it is in the optional payload of the ping message (see the description of Assignment P2).

2.7 Serial console

The textual interface via serial console is expanded include commands to configure encryption.

- Command `/ENCRYPT` (or `/ENC`) sets the encryption key. The 256-bit key is given in hexadecimal format, i.e., 32 bytes (and 64 hex-digits).
  - Should the input be too short, it will be padded with zeroes.
  - If no key is provided, the encryption mode is disabled and the device returns back to the normal version 1 operation.

3 Testing Procedure

The test device at Gróska will talk both v1 and v2 protocols. The encryption key is
(with zero-padding). You can visit Gróska anytime to test and verify your code. Send `report` via the chat application to get an automatic test report!

Test protocol: Your implementation will be tested as follows:

1. Device has to respond to pings in the encrypted channel (AES encryption works).
2. Device shall receive commands (both time and test) with valid and invalid signatures. It shall only act on valid commands.