CPA encryption and decryption in OFB mode

Sarthak Mahajan, 2018111005

March 7, 2022

1 Code structure:

- This folder contains cpa.py and run.py code files
- cpa.py contains the code to CPA encrypt and decrypt messages.
- run.py is used to take inputs and return outputs of the CPA encryptor/decryptor.

2 Instructions to run the code:

- Run the following command in the folder where this document is contained: python3 run.py
- There will be prompts asking you to input the relevant information serially, and will return the output.

3 Explanation of Functions in run.py:

3.1 run():

- It asks for inputs of the choice of encryption/decryption, key, key length in binary, message or ciphertext(depending on encryption/decryption choice)(all in decimal format)
- Calculates p: the order of the group for DLP and g: the generator of this group.
- prints the encrypted message or decrypted cipher depending on the choice.

4 Explanation of Functions in cpa.py:

4.1 OFB_CPA_enc(m, k, key_len,p,g):

- p,g are in decimal format, they are needed to be passed to the dlp function. p is the largest prime possible in n bits, where n= length of the key, g is a

random number in the range(1,p-1); note that this is fixed for the program in terms of p(so that same seed gives same value for different iterations of the program), but it could be any random number in the given range.

- all inputs are in decimal format, m represents the message to be encoded, k is the key, key_len is the length of the key in binary format.
- It returns a CPA secure ciphertext of the message m in binary format
- It uses the OFB mode of operation where each block is a PRF for encryption
- First we convert m into binary format, and cut it to chunks of size key_len, if the last chunk's size is lesser the key's size we pad it with zeros on the right side
- iv is an initialization vector that is set to the number of padded zeros in the last chunk of the message
- We implement the OFB by iterating over message chunks in a for loop, calculating PRF, taking XOR and appending to final ciphertext: cipher
- Uses pad,PRF functions

4.2 OFB_CPA_dec(c, k, key_len,p,g):

- p,g are in decimal format, they are needed to be passed to the dlp function. p is the largest prime possible in n bits, where n= length of the key, g is a random number in the range(1,p-1); note that this is fixed for the program in terms of p(so that same seed gives same value for different iterations of the program), but it could be any random number in the given range.
- c is the ciphertext to be decrypted in binary format, k is the key in decimal format, key_len is the length of the key in binary format.
- It returns the decrypted ciphertext in binary format
- It uses the OFB mode for decryption
- Its symmetrical, to encryption (as its OFB mode), with cipher and message replaced, but the iv is the same as in encryption: the 1st chunk of the ciphertext c
- After calculating the OFB output using for loop, we take out the last iv bits, as its the number of padded zeros during encryption.
- Uses pad,PRF functions