Java Programming assignment #5: DES (Data Encryption Standard) with Java

due by June 16th, 2019

## Introduction

We developed encoding/decoding classes library. Codec (coder and decoder) is all about data presentation. The final codec library is about cryptography. By changing the bits in the plain text, we can keep confidential data from adversaries.

One important goals of data security is 'confidentiality.' Confidentiality prohibits to access correct data from unauthorized users. To keep confidentiality, one can change the symbols or codeword, so that arbitrary users cannot properly find the meaning in the text.

## Out-fashioned Cipher

One the oldest data encryption codes is Caesar's code, which is known to be used by Julius Caesar. To keep the military confidential information, important documents are encrypted, and only the generals can properly comprehend the delivered ordering. The Caesar's code uses a shift key. The shift key is secretly distributed before. Then, all characters in the plain text are changed, according to shift key. The shift key says how many characters are shifted from the plain text. For example, let's assume that we have plain text, 'Attack on South', and shift key is 3. Then, the first character 'A' is changed into 'D', by shifting three characters from the original character 'A'. Similarly, 't' becomes 'w', 'c' becomes 'f', and so on. The encrypted text is 'Dwwdfm rq Vrxwk.' Thus, encryption hinders a reader from understand the meaning of original text. Note that English alphabet has 26 characters, and after the last character 'z', the encryption uses 'a, b, c, …' so that all the characters can be mapped.

It can be simply decrypted by shifting characters on the opposite direction. At this time, the same shift key is required. If the shift key is three, then we can go back three characters from the alphabet sequence, and we can find the correct character.

There are many attacks on encryption and cryptographic algorithms. There are

some known attacks on the Caesar's code; First, we can guess the shift key value. Once shift key is known, Caesar's code is doomed. Even though the mapping is different, the frequency of each character is the same. Thus, carefully observing the frequency of the characters used in the encrypted text, and general frequency of used characters in English text can be compared. Then, we can guess some possible shift key values. Try one by one, until it makes feasible plain text.

Second, we can try to decode the text using all the possible shift keys. The possible key values are from 0 to 25, i.e. 26 keys are all possible values. Because English alphabets have 26 distinct characters, we can try with all 26 keys. This approach is called as 'brute-force' method.

After all, Caesar's code is not used for modern digital data encryption system. Many mathematical & engineering approaches are presented, so that an attackers are difficult to break the confidentiality. One of modern encryption algorithm is DES (Data Encryption Standard). DES shuffles bits based on confusion and diffusion principles. The algorithm uses 16 different keys for encryption, and bit changes are spread over the text for each encryption rounds.

The DES-Data Encryption Standard- algorithm

The goal of this homework is to implement DES algorithm. DES encryption can be found in many literatures, and I will take single example for concrete description of the algorithm. At first, DES is a block cipher algorithm, encode fixed length bit group (64 bits or 8 bytes) forms a single block. A block is encrypted once, and decrypted once as a unit.

At first, overall DES process is described, and then detailed algorithm will be explained. DES encryption conducts 16 iteration of sub encoding (encryption) processes. Each encryption process is called as round, and different keys are used for different rounds. The input block (64 bits) goes through initial permutation, which is randomly looking relocation of bits, and the first of 16 rounds begins. Each round encodes bits according to Feistel structure, which we will explain later. After the 16 rounds, the output goes through final permutation, which is another relocation of bits.

The Feistel structure divide the input bits into half, and apply encryption operation on the left half using the right half, and the round-specific encryption

key.

The algorithm works with one random seed (DES key), and seven pre-defined tables. The following sections illustrate the DES process.

First, DES generates the encryption keys, or subkeys. The encryption is processed through 16 iteration of encoding rounds. In each round, different sub keys are used, and the sub keys used for 16 rounds are derived from initial DES key. From initial 64 bit DES key, every 8-th bits (8, 16, 24, 32, 40, 48, 56, 64) are ignored, and only 56 bits are used to derive the sub keys. Then, 56 bits are rearranged according to PC-1 permutation table. The permuted 56-bits are divided into half. Next, left half and right half generate the next round's left and right half of sub keys, respectively. At round of 1, 2, 9, 16, left and right keys are left-rotated once, rest of the rounds keys are left-rotated twice.

Then, we have 56-bit values for each round, and we select only 48 of 56 bits, according to PC-2 permutation table. Now, we have 48-bit secret key for each round encryption, that is called Subkey. That's DES key scheduling, the first phase of DES encryption.

Second, DES encodes 64-bit block data. Firstly, the 64-bit data goes through initial permutation, arbitrary bit relocation. According to IP table, bits are rearranged, and 64 bits are prepared for entering Feistel network.

The Feistel network takes two inputs: 64-bit input data and subkey, and single output: 64-bit output data. The Feistel network's input data (64 bit) is divided into two halves (L0, R0). Note that both L0 and R0 are all 32 bits. Here, R0 is not encrypted, and immediately delivered to left half of output, L1.

On the other hand, L0 is encrypted. L0 is encoded with R0 and SubKey0. So, we will call S-box function, f, that generates f(SubKey0, R0). The f function generates 32 bits from 48 bit subkey for round 0, and 32 bit R0. To match the number of bits in data, 32 bit R0 values are expanded according to E table. Then, two 48 bits are XOR'ed. Now, we have 48 bits, and makes 8*6bits pairs. Each 6 bits group is further permuted according to S0-table and carefully select 4 bits. Thus, we have 8*4 = 32 bits, the output of f-function.

The 32-bit output of f-function is XOR'ed with original L0 value, and we get the

final piece, output of right half R1.

To repeat the rounds, almost all idea is the same as round-0, except for SubKey and S0-table. In round 1, we use SubKey1, S1-table, and round N (<=16), we use SubKeyN, and SN-table. They can be found in Wikipedia page. That's Feistel Network processing.

All after Feistel network processing, output is pair of L15, R15. They are merged as single 64 bits. Then, Final Permutation is conducted. The Final permutation rearranges bits according to FP table, or ($IP^{-1}$, inverse IP table). That's final result of DES encryption.

Side note:

DES algorithm is known to be weak against differential cryptoanalysis attack and possibly brute-force attacks; The primary reason is that the key length is too short for the modern computers (too easy to break), and now outdated officially. So, do not use for any professional/commercial use.

Your Goal

Very similarly to what we did in the encoding/decoding, encrypt plain text file using DES algorithm.

Requirements to do

Please stick to JCF (Java Core Framework), and try to practice what you've learned in the class.

Req1. Design and implement DES encoding class:

Req2. Make class hierarchy, along with your encoding library. Design a parent class or interface. Make some classes relevant to the DES encryption class.

Req3. Test your class with the homework description text.

Optional Req4. Make a decoding method that gets you back the original text.

We will build an example Table look up class;

Seehwan Yoo