**Project 1**

CS-4780

Howard Nguyen, Andres Vicente, Antonio Vazquez Bravo

**Running the SDES (SDES.SDES)**

The two main and important functions are the encrypt and decrypt functions which have their own string parameter overload to cipher strings, by default the functions take in a short type value. After importing the SDES class, statically call the encrypt or decrypt functions while passing either a short-type or string-type value, and a KeyGeneration instance as parameters. The functions will return the desired short or string output depending on which is used.

**Steps**

1. Import SDES and KeyGeneration
2. Create instance of KeyGeneration by passing in a 10-bit short-type value
3. Call encrypt/decrypt from SDES and pass in a plain string or short-type value, and the KeyGeneration instance
4. Functions should return a processed short or string value depending on input

**Generating a Key (SDES.KeyGeneration)**

To generate keys used in this project, an instance of the KeyGeneration class is needed to be passed into the ciphering function. When calling the class’s constructor a 10-bit short-type value is required and passed in, which on construction will generate the 2 subkeys used for the SDES. Getter functions for both subkeys (k1, k2) are provided.

**Running Triple SDES (SDES.TripleSDES)**

Triple SDES operates the same as SDES, however two KeyGeneration instances for each raw key are required as parameters. See SDES for more details.

**Dependencies**

* **PermTable:** a class that provides static functions of permutation table and S-box results. Used by SDES class.
* **CASCII (Part 3 only):** The provided class to convert from ascii to compact ascii. Only used to do part 3 of the project.

**Part 1:**

**Raw key Plain Cipher**

0000000000 00000000 01001000

1111111111 11111111 11010110

0000011111 00000000 00011100

0000011111 11111111 11000000

0000000000 01001000 00000000

1111111111 11010110 11111111

0000011111 01001101 00000000

0000011111 00011011 11111111

**Part 2:**

**Raw key 2 Raw Key 2 Plain Cipher**

0000000000 0000000000 00000000 01001000

1000101110 0110101110 11010111 10101100

1000101110 0110101110 10101010 01110101

1111111111 1111111111 10101010 01111101

1000101110 0110101110 01101110 11100110

1011101111 0110101110 10010111 01010000

0000000000 0000000000 11001110 10000000

1111111111 1111111111 00001110 10010010

**Part 3:**

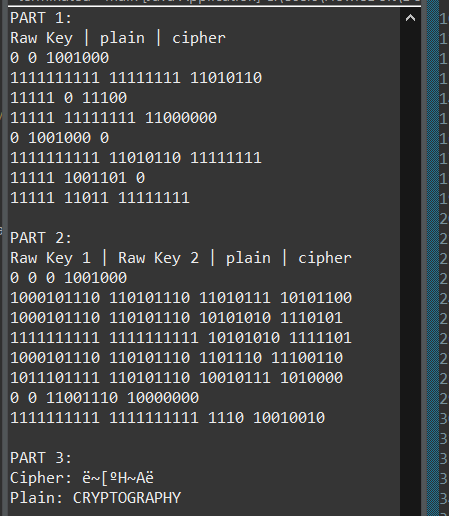
1. Plain: CRYPTOGRAPHY

Cipher: ë~[ºH~Aë

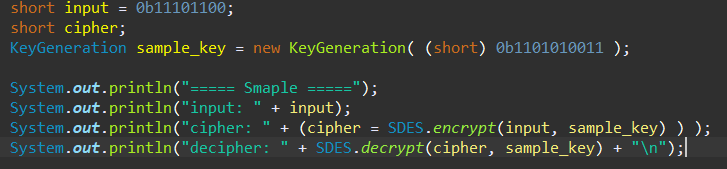
2.3.) An attempt was made with parts 3.2, and 3.3. The method used was a brute-force approach where the decryption function would produce all possible results using all possible keys from 0 to 1111111111. The results can be found in out.txt and out2.txt for all combinations, however tangible text can’t be found leading to some improper use of CASCII conversions.

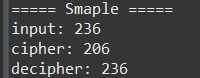
**Screenshots**

**Output of all 3 parts (excluding 3.2 3.3)**



**Sample encrypt and decrypt**

****

****