In this assignment, you will be asked to implement different encryption and decryption algorithms using C, making your own cryptographic library. You should create 2 files: cs457_crypto.h, containing the function declarations and cs457_crypto.c, containing the implementation of the functions.

1. One-time pad

One-time pad is an encryption algorithm that uses a randomly generated key (also referred to as a one-time pad) that is at least the size of the plaintext. During encryption, each bit or character of the plaintext is XOR-ed with the corresponding bit or character of the key.

a. Implement the functions one_time_pad_encr and one_time_pad_decr, that receive as arguments the plaintext/ciphertext as well as the random key and return the result. In order to produce the key, you should use a pseudorandom number generator (such as /dev/urandom). Since /dev/urandom will return a new random value upon each read, you will first need to generate a random secret key and store it in memory in order to successfully decrypt the encrypted message. Assume that the plaintext consists only of letters or numbers.

b. Let us assume that a programmer used the same “one time pad” to encrypt two English words. The result of the encryption of the two words is “e9 3a e9 c5 fc 73 55 d5” and “f4 3a fe c7 e1 68 4a df”. Implement the function word_decryption() that finds which these two words are and prints them. The list of all English words can be found at https://raw.githubusercontent.com/dwyl/english-words/master/words.txt

2. Rail Fence cipher

Rail fence (also called a zigzag cipher) is a form of classical transposition cipher. The plaintext is written downwards diagonally on “rails”, then moving up when the bottom rail is reached, down again when the top rail is reached and so on until the whole plaintext is written out. After that, the individual rows are combined to obtain the ciphertext. The following example shows how the algorithm works with the plaintext “Hello World”. The
produced ciphertext is “Horel ollWd”. Note that the space is written and read normally as the other characters.

```
  H o r l
  e W o l
  l W d
```

a. Implement the functions `rail_fence_encr` and `rail_fence_decr`, which receive as arguments the plaintext/ciphertext and an integer representing the number of rails the algorithm will use. The functions should return the result of the operation.

3. Beaufort cipher

The Beaufort cipher is a substitution cipher that uses a tableau (shown below) and a keyword to encipher the plaintext. The keyword should be repeated in order to make a key that matches the size of the plaintext. The steps for the encryption are the following for every plaintext letter:
- find the plaintext letter in the topmost horizontal row
- travel down the column, until you find the current key letter
- the leftmost letter in the current row is the new ciphertext letter

For example, if the plaintext is “ATTACKATDAWN” and the keyword is “LEMON”, the ciphertext is:

Plaintext: ATTACKATDAWN
Key: LEMONLEMONLE
Ciphertext: LLTOLBETLNPR

The deciphering is performed using the same algorithm.
## a. Beaufort Cipher

Implement the functions `beaufort_encr` and `beaufort_decr`, which receive as arguments the plaintext/ciphertext and the keyword that the algorithm will use. The functions should return the result of the operation. Assume that the plaintext consists only of capital English letters.

| Z   | A   | B   | C   | D   | E   | F   | G   | H   | I   | J   | K   | L   | M   | N   | O   | P   | Q   | R   | S   | T   | U   | V   | W   | X   | Y   | Z   |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| Z   | A   | B   | C   | D   | E   | F   | G   | H   | I   | J   | K   | L   | M   | N   | O   | P   | Q   | R   | S   | T   | U   | V   | W   | X   | Y   | Z   |
| A   | B   | C   | D   | E   | F   | G   | H   | I   | J   | K   | L   | M   | N   | O   | P   | Q   | R   | S   | T   | U   | V   | W   | X   | Y   | Z   |
| B   | C   | D   | E   | F   | G   | H   | I   | J   | K   | L   | M   | N   | O   | P   | Q   | R   | S   | T   | U   | V   | W   | X   | Y   | Z   |
| C   | D   | E   | F   | G   | H   | I   | J   | K   | L   | M   | N   | O   | P   | Q   | R   | S   | T   | U   | V   | W   | X   | Y   | Z   |
| D   | E   | F   | G   | H   | I   | J   | K   | L   | M   | N   | O   | P   | Q   | R   | S   | T   | U   | V   | W   | X   |

4. **Affine Cipher**

Affine cipher is a substitution cipher, where each letter of the alphabet is mapped to its numeric equivalent, encrypted using a mathematical function and converted back to a letter. Each letter is enciphered with the function \((ax + b) \mod m\). Assume that we correspond each letter with its position in the alphabet starting from 0 (A-0, B-1, C-2 etc.).

For example, we want to encrypt “AFFINE CIPHER” with \(a=5\), \(b=8\) and \(m=26\).
<table>
<thead>
<tr>
<th>Plaintext</th>
<th>A</th>
<th>F</th>
<th>F</th>
<th>I</th>
<th>N</th>
<th>E</th>
<th>C</th>
<th>I</th>
<th>P</th>
<th>H</th>
<th>E</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>0</td>
<td>5</td>
<td>5</td>
<td>8</td>
<td>13</td>
<td>4</td>
<td>2</td>
<td>8</td>
<td>15</td>
<td>7</td>
<td>4</td>
<td>17</td>
</tr>
<tr>
<td>(5x+8)mod 26</td>
<td>8</td>
<td>7</td>
<td>7</td>
<td>22</td>
<td>21</td>
<td>2</td>
<td>18</td>
<td>22</td>
<td>5</td>
<td>17</td>
<td>2</td>
<td>15</td>
</tr>
<tr>
<td>Ciphertext</td>
<td>I</td>
<td>H</td>
<td>H</td>
<td>W</td>
<td>V</td>
<td>C</td>
<td>S</td>
<td>W</td>
<td>F</td>
<td>R</td>
<td>C</td>
<td>P</td>
</tr>
</tbody>
</table>

5. Feistel cipher

Feistel cipher model is a structure used to develop many block ciphers such as DES (block ciphers encrypt a fixed size block of the plaintext at a time). The Feistel cipher consists of rounds and a separate key is used on each round. For the i-th round the process is the following:

- Divide the input block into 2 halves, the left one (L_i) and the right one (R_i).
- Compute L_{i+1} = R_i and R_{i+1} = L_i XOR f(R_i, K_i), where f is an encrypting function and K_i is the key of this round.

After n rounds, the final ciphertext is (R_{n+1}, L_{n+1}).

The decryption process is similar, with the difference of using the keys in the reverse order.

a. Implement the functions **feistel_encr** and **feistel_decr**, which receive as arguments the plaintext/ciphertext, the size of the plaintext/ciphertext as well as the keys and return the result of the operation. Assume that n=4, block size = 128, size of R_i = 64, size of L_i = 64 and f(r, k_i) = (r*k_i) mod (2^64). You should generate the keys using a pseudorandom number generator (as with one-time pad algorithm). Each key should be the same size as the right part of the block (in our case 64 bits).
Notes

- You need to submit cs457_crypto.h, cs457_crypto.c, a Makefile that compiles the library, a Readme file explaining your implementation or unimplemented parts and a test file that utilizes the implemented functions.
- If you implement more helper functions or macros, explain their functionality in the Readme file.
- This assignment should be implemented using C on Linux-based machines.
- You can use the course’s mailing list for questions. However, read the previous emails first since your question might have already been answered.
- Do not send private messages with questions to the TAs, since other students might have the same question and everyone deserves the answer.
- Do not send code snippets or pieces of your implementation to the mailing list when asking a question.
- Submitted code will be tested for plagiarism using plagiarism-detection software.