Second homework Joosep Parts 221963IVCM

1 Task

Given the instantiation of Feistel network:

- Block size = 16
- Number of rounds = 6
- Round function: $F(k_i, x) = k_i \oplus x$
- Key generation: $k_{i+1} = k_i \oplus \text{bin}(i+1)$ where $\text{bin}(i+1)$ is the binary representation of $(i+1)$.

To encrypt the message $m = 0110\ 1001\ 1010\ 1111$ using the key $k_0 = 0101\ 1001$, we split the message into two halves, $L_0 = 0110\ 1001$ and $R_0 = 1010\ 1111$.
Using the round function and the key generation function, the encrypted message after 6 rounds of Feistel is calculated as:

$$C = L_6 R_6$$
$$L_i = R_{i-1}$$
$$R_i = L_{i-1} \oplus F(R_{i-1}, k_i)$$
$$k_i = k_{i-1} \oplus \text{bin}(i)$$

After 6 rounds, the encrypted message is:

$$C = 0110\ 1100\ 1010\ 1000$$

```
[1] iteration 0 is: 1010 1111 1001 1111
[1] iteration 1 is: 1001 1111 0110 1000
[1] iteration 2 is: 0110 1000 1010 1101
[1] iteration 3 is: 1010 1101 1001 1100
[1] iteration 4 is: 1001 1100 0110 1100
[1] iteration 5 is: 0110 1100 1010 1000
[1] answer is: 0110 1100 1010 1000
```

Proof can be seen in code: `stackblitz hw21`
Code is also in repository: `here`

2 Task

Using the Vigenère cipher to encrypt the plaintext "THEWANDCHOOSESTHEWIZARD" with key "MAGIC", we get:

$$C = \text{FHKECZDIPQASKAVTECQBMRJ}$$
$$C' = \text{FHKECZDIPQASKAVTERQBMRJ}$$

Diffusion: let's denote the original ciphertext as C and the modified ciphertext (after changing one letter in the plaintext) as C'. The number of different characters between C and C' can be represented as the function $\Delta(C, C')$. Thus, for our given texts:

$$\Delta(C, C') = n$$

where n is the number of letters that changed in the ciphertext due to the modification in the plaintext. The limited value of $n = 1$ suggests limited diffusion in the Vigenère cipher for this example. For diffusion, we would want n to change for more than half of the bits.

$$C = \text{FHKECZDIPQASKAVTECQBMRJ}$$
$$C'' = \text{FHRECZDPPQASRAVTEJQBMRQ}$$

Confusion: let's denote the ciphertext produced with the modified key as C''. The number of different characters between C and C'' can be represented as:

$$\Delta(C, C'') = m$$

where m is the number of letters that changed in the ciphertext due to the modification in the key. Key is now $MANIC$ instead of $MAGIC$. A relatively larger value of $m = 5$ suggests 'better' confusion in the Vigenère cipher for this example, however I would say the statistical properties have not change much regardless of the key modification. For confusion, we would want m to change for more than half of the bits.

```
[2.1] Encrypted message: FHKECZDIPQASKAVTECQBMRJ
[2.2] Letter C changed to R at index 17
[2.2] modifiedPlainText: THEWANDCHOOSESTHELIZARD
[2.2] modifiedEncryptedText: FHKECZDIPQASKAVTERQBMRJ
[2.2] letters changed: 1
[2.3] Letter K changed to R at index 2
[2.3] Letter I changed to P at index 7
[2.3] Letter K changed to R at index 12
[2.3] Letter C changed to J at index 17
[2.3] Letter J changed to Q at index 22
[2.3] Key: MAGIC
[2.3] Encrypted with original key: FHKECZDIPQASKAVTECQBMRJ
[2.3] Modified key: MANIC
[2.3] Encrypted with modified key: FHRECZDPPQASRAVTEJQBMRQ
[2.3] Letters changed after modifying key: 5
```

Proof can be seen in code: `stackblitz hw22`
Code is also in repository: `here`

3 Task

For example, I take random key and two message parts and apply them. For the function 1:

$$\begin{aligned} \text{Given } m &= \text{"THEWAND"}, \\ F'(k,m) &= F(k, \text{"THEWAND"}) \parallel 0^7 \\ &= \text{"FHKECZD"} \parallel \text{"0000000"} \\ &= \text{"FHKECZD0000000"} \end{aligned}$$

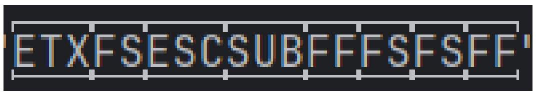
When appending 0's at the end of the function makes it predictable as we can see it's not completely indisdinguishable from totally random function.

For the function 2:

$$\begin{aligned} \text{Given } m_1 &= \text{"THEWAND"}, \\ m_2 &= \text{"CHOOSER"}, \\ F'(k, m_1 \parallel m_2) &= F(k, \text{"THEWAND"}) \parallel F(k, \text{"CHOOSER"}) \oplus \text{"0000000"} \\ &= \text{"FHKECZD"} \parallel \text{"LFHBL"} \\ &= \text{"FHKECZDLFHBL"} \end{aligned}$$

For this the XOR with 0's does not change m2, so it's better but the security depends on the security of the function F. So it's not completely secure but better than function from task 1.
For the function 3:

$$\text{Given } m_1 = \text{"THEWAND" and } m_2 = \text{"CHOOSER"},$$
$$F'(k, m_1 \parallel m_2) = F(k, \text{"0THEWAND"}) \oplus F(k, \text{"CHOOSER1"})$$
$$\text{Result} = \text{""}$$

Text is in hex, so it's not readable. 
'ETXFSESCSUBFFFSFSFF'
Given:

$$F'(k, x \| y) = F(k, 0 \| x) \oplus F(k, y \| 1)$$

where $x, y \in \{0,1\}^{n-1}$. Let's consider a counter-example: If the adversary selects $x = 1^{n-1}$ (all 1s) and $y = 0^{n-1}$ (all 0s), then the following occurs:

$$F'(k, x \| y) = F(k, 01^{n-1}) \oplus F(k, 0^{n-1}1)$$

The left component becomes:

$$F(k, 01^{n-1})$$

And the right component becomes:

$$F(k, 0^{n-1}1)$$

Given the bitwise XOR operation, the beginning element will be $1 \oplus 1 = 0$ and the last element will be $1 \oplus 1 = 0$. All other elements for both x and y remain the same, which means they will also be zero. Thus, the function will produce only zeros, making it distinguishable from a random function. Hence, F' does not satisfy the definition of a pseudo-random function when given this adversary's choice.

```
[3.1] Using F’(k, m) = F(k, m)||0^n
[3.1] Message: THEWAND
[3.1] Key: MAGIC
[3.1] Result: FHKECZD0000000
[3.2] Using F’(k, m1||m2) = F(k, m1)||F(k, m2  0^n)
[3.2] Message 1: THEWAND
[3.2] Message 2: CHOOSER
[3.2] Key: MAGIC
[3.2] Result: FHKECZDLFHBL
[3.3] Using F’(k, m1||m2) = F(k, 0||m1)  F(k, m2||1)
[3.3] Message 1: THEWAND
[3.3] Message 2: CHOOSER
[3.3] Key: MAGIC
[3.3] Result:
```

Proof can be seen in code: `stackblitz hw23`
Code is also in repository: `here`

4 Task

4.1: Encryption using Permutation Cipher in OFB Mode

Given:

$$D = 00011$$
$$O = 01110$$
$$G = 00110$$

Thus, the plaintext in binary is:

$$P = 00011\,01110\,00110$$

For the OFB mode with key $K = (4, 1, 3, 5, 2)$:

1. IV $w = 01011 \rightarrow$ Permute to get 10011
2. XOR P_1 with IV to get $C_1 : 00011 \oplus 10011 = 10000$
3. Continue for P_2, P_3 to get C_2, C_3

Resulting in:

$$C = 10000\,10100\,11111$$

4.2: Decryption with Modified Ciphertext

Flipping the 5-th bit of C:

$$C' = 10001\,10100\,11111$$

Decryption yields:

$$P' = 00010\,01110\,00110$$

Differing from P by 1 bit.

4.3: Decryption with Modified IV

Using the modified IV $w' = 11011$, decryption provides:

$$P'' = 01011\,01111\,00100$$

This differs from P by 3 bits.

```
[4] Binary Plaintext D: 00011 O: 01110 G: 00110
[4] After Permutation (using key): 10011
[4] XOR of Plaintext block: 00011 with iv: 10011 and get EncBlock: 10000
[4] After Permutation (using key): 11010
[4] XOR of Plaintext block: 01110 with iv: 11010 and get EncBlock: 10100
[4] After Permutation (using key): 11001
[4] XOR of Plaintext block: 00110 with iv: 11001 and get EncBlock: 11111
[4.1] Encrypted Ciphertext: 10000 10100 11111
[4.2] Modified Ciphertext by flipping 5-th bit: 100011010011111
[4.2] Decrypted Binary for modified Ciphertext: 000100111000110
[4.2] originalBinaryPlaintext: 000110111000110
[4.2] decryptedBinary2: 000100111000110
[4.2] changedBits2: 1
[4.3] Decrypted Binary using modified IV: 010110111100100
[4.3] decryptedBinary3: 010110111100100
[4.3] originalBinaryPlaintext: 000110111000110
[4.3] Number of bits changed in plaintext for: 3
```

Proof can be seen in code: `stackblitz hw24`
Code is also in repository: `here`

5 Task

article tikz

6 Task

$$\begin{tikzcd}
\text{IV: } r_0 \arrow[r] & E(k, m_1) \arrow[r, "\oplus"] & c_1 \arrow[dl] \\
m_1 \arrow[ur] & E(k, m_2) \arrow[r, "\oplus"] & c_2 \arrow[dl] \\
m_2 \arrow[ur] & E(k, m_n) \arrow[r, "\oplus"] & c_n \\
m_n \arrow[ur] & &
\end{tikzcd}$$

Decryption Process

XOR is its own inverse, so the decryption process is almost identical to the encryption process but in reverse order. To decrypt you would do the following:

$$\begin{tikzcd}
\text{IV: } r_0 \arrow[r] & D(k, c_1) \arrow[r, "\oplus"] & m_1 \arrow[dl] \\
c_1 \arrow[ur] & D(k, c_2) \arrow[r, "\oplus"] & m_2 \arrow[dl] \\
c_2 \arrow[ur] & D(k, c_n) \arrow[r, "\oplus"] & m_n \\
c_n \arrow[ur] & &
\end{tikzcd}$$

7 Task

6.1: Output each integer five times

Given that one output operation takes a constant amount of time (let's call this constant c), for n integers, outputting each 5 times will take $5nc$ time.

$$\text{Time for 1} : 5nc = c \times O(n) = O(5 \times n)$$

6.2: Apply bubble sort

In bubble sort, we repeatedly traverse the array and swap adjacent elements if they are in the wrong order. The worst-case occurs when the array is reverse sorted, which requires $\frac{n(n-1)}{2}$ comparisons and swaps. Let's take a simple reversed array of size 4 for illustration: $\{4, 3, 2, 1\}$

- **1st pass**: Compare 4 and 3, 3 and 2, 2 and 1. 3 swaps. Array becomes $\{3, 2, 1, 4\}$

- **2nd pass**: Compare 3 and 2, 2 and 1. 2 swaps. Array becomes $\{2,1,3,4\}$
- **3rd pass**: Compare 2 and 1. 1 swap. Array becomes $\{1,2,3,4\}$

$$\text{Time for 2}: \frac{n(n-1)}{2} = O(n^2)$$

6.3: Find the greatest common divisor of the 2nd and 4th elements of the array

Accessing specific positions within an array, such as the 2nd and 4th elements, is a direct operation with constant time complexity, denoted as $O(1)$. For the computation of the GCD, the Euclidean algorithm is typically employed. The worst-case time complexity of the Euclidean algorithm is bounded by the number of digits in the smaller number. Let's denote the time complexity of the algorithm by $T(a,b)$, where a and b are the two numbers with $a > b$.

$$T(a,b) = 1 + T(b, a \mod b) \tag{1}$$

The equation above illustrates the recursive nature of the Euclidean algorithm. The worst-case scenario occurs when every iteration reduces the number by a constant factor. This corresponds to the Fibonacci sequence, as demonstrated by `Lamé's theorem`. Given Lamé's theorem, the number of steps required is bounded by the index of the Fibonacci number that is closest to b. If this index is h, then:

$$F_h \approx \phi^h \Rightarrow h \approx \log_\phi b \tag{2}$$

Where ϕ is the golden ratio. Therefore, the time complexity can be expressed in terms of the logarithm:

$$\text{Time for Step 3}: O(\log b) \tag{3}$$

In essence, the $O(\log b)$ complexity arises because the number of divisions required by the Euclidean algorithm is at most proportional to the number of digits in b, which scales logarithmically with b.

8 Task Bonus

OTP (One Time Pad) encryption, when used correctly, provides perfect secrecy. However, when applied in CBC mode, its properties change. In CBC mode, each block of plaintext is XORed with the previous ciphertext block before being encrypted with the OTP. The first block of plaintext is XORed with an initialization vector (IV). Thus, if an adversary knows one plaintext block and its corresponding ciphertext block, he can infer information about the next plaintext block. Code encrypts both m_0 and m_1 in CBC mode using OTP. Then, the adversary attacks by computing the encryption of m_0 and comparing it with the received ciphertext to infer which message was encrypted first.

```
[BONUS] m0 string: Hello
[BONUS] m1 string: World
[BONUS] key: supersecretkey
[BONUS] iv: 12345
[BONUS] otpCbcEncrypt with: Hello+World supersecretkey 12345
[BONUS] CipherText:"/=(.(>#)
[BONUS] cipherM0:"/=(
[BONUS] cipherText.slice(0, iv.length):"/=(
[BONUS] m0(1st block) has been encrypted.
```

Proof can be seen in code: `stackblitz bonus`
Code is also in repository: `here`