Здесь и далее используется AES с длиной ключа 128 бит. В качестве хэш-функции используется SHA-256.

## 1. Реализовать функцию на вашем языке программирования со следующим интерфесом:

byte[] AesBlockEncrypt(byte[] key, byte[] data, bool isFinalBLock, string padding), где

key - байтовое представление ключа блочного шифра

data - блок для шифрования

isFinalBLock - флаг того, что передан последний блок шифруемого открытого текста

padding - вид дополнения, принимает значение MAC\_padding (10..00000) для OMAC, 'PKCS5' для truncated-MAC, 'NONE' для HMAC.

В ходе реализации необходимо пользоваться стандартной или общеизвестной реализацией алгоритма AES. Если библиотека не поддерживает одноблочное шифрование AES, необходимо воспользоваться режимом ECB.

Пример одноблочного шифрования на C#.

namespace AesExample  
{  
 using System;  
 using System.Security.Cryptography;  
 using System.Text;  
  
 internal class Program  
 {  
 private static void Main(string[] args)  
 {  
 // Your Key here, 16 bytes  
 byte[] key =  
 { 0x73, 0x69, 0x78, 0x74, 0x65, 0x65, 0x6e, 0x2d,  
 0x62, 0x79, 0x74, 0x65, 0x2d, 0x6b, 0x65, 0x79 };  
  
 // Plaintext, 16 bytes string, utf-8  
 string stringPt = "sixteen-byte-msg";  
  
 // Raw Plaintext, bytes  
 byte[] pt = Encoding.UTF8.GetBytes(stringPt);  
  
 // Resulted Ciphertext will be here  
 byte[] ct = new byte[16];  
  
 // Create new AES instance  
 using(Aes aes = new AesCryptoServiceProvider())  
 {  
 // Select Encryption mode  
 aes.Mode = CipherMode.ECB;  
  
 // Create encryptor with your key and zero IV  
 using (var aesEncryptor = aes.CreateEncryptor(key, new byte[16]))  
 {  
 // Transform one block  
 aesEncryptor.TransformBlock(pt, 0, 16, ct, 0);  
 }  
  
 // Get hex-string representation of Ciphertext  
 string hex = BitConverter.ToString(ct);  
 Console.WriteLine(hex.Replace("-", ""));  
 }  
 }  
 }  
}

c++

#include <stdio.h>   
#include <openssl/aes.h>   
  
static const unsigned char key[] = {  
 0x73, 0x69, 0x78, 0x74,   
 0x65, 0x65, 0x6e, 0x2d,  
 0x62, 0x79, 0x74, 0x65,  
 0x2d, 0x6b, 0x65, 0x79  
};  
  
int main()  
{  
 unsigned char text[]="sixteen-byte-msg";  
 unsigned char enc\_out[80];  
 unsigned char dec\_out[80];  
  
 AES\_KEY enc\_key;  
  
 AES\_set\_encrypt\_key(key, 128, &enc\_key);  
 AES\_encrypt(text, enc\_out, &enc\_key);   
  
 int i;  
  
 for(i=0;\*(enc\_out+i)!=0x00;i++)  
 {  
 printf("%X ",\*(enc\_out+i));  
 }  
 printf("\n");  
  
 return 0;  
}

python

from Crypto.Cipher import AES  
cipher = AES.new(b'sixteen-byte-key',AES.MODE\_ECB)  
cipher.encrypt(b'sixteen-byte-msg').hex()

## 2. Реализовать OMAC, truncated-MAC (64 бита) с использованием функции AesBlockEncrypt, HMAC на основе SHA-256.

#### 2.0 Реализовать интерфейс void SetKey (byte[] key)

* инициализирует объект MAC ключом key.

#### 2.1 Реализовать интерфейс void MacAddBlock(byte[] dataBlock)

* добавляющий блок данных для вычисления кода аутентичности, с использованием AesBlockEncrypt или SHA-256.

#### 2.2 Реализовать интерфейс byte[] MacFinalize()

* возвращающий результат вычисления кода аутентичности, для всех запрошенных блоков.

#### 2.3 Реализовать интерфейс byte[] ComputeMac(byte[] data)

* вычисляющий код аутентичности для прозвольных данных, используя метод MacAddBlock

#### 2.4 Реализовать интерфейс bool VerifyMac(byte[] data, byte[] tag)

* проверяющий код аутентичности для прозвольных данных, используя метод ComputeMac

#### 2.5 (Опционально) Использовать реализации реализованных алгоритмов в вашем языке программирования для валидации вашей реализации.

## 3. Для каждого алгоритма выработки кода аутентичности вычислить и проверить для произвольного текст длины 2,5 блока.

#### 3.1 Поменять один бит в блоке - убедиться, что проверка кода аутентичности не выполняется, для модифицированных данных.

## 4. Замерить производительность OMAC и HMAC для сообщений, длины 0.1, 1, 10, 1024 KB (не менее 1000 сообщений).

Построить график зависимости среднего времени выполнения от размера сообщений (2 графика).

#### Доп материалы

Ссылка OMAC (CMAC) https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-38b.pdf (пункт 6)