BİL – 470 KRİPTOGRAFİ VE BİLGİSAYAR GÜVENLİĞİ

PROGRAMLAMA PROJESİ RAPORU

ÖĞRENCİ: NEVRA GÜRSES
NUMARA: 161044071

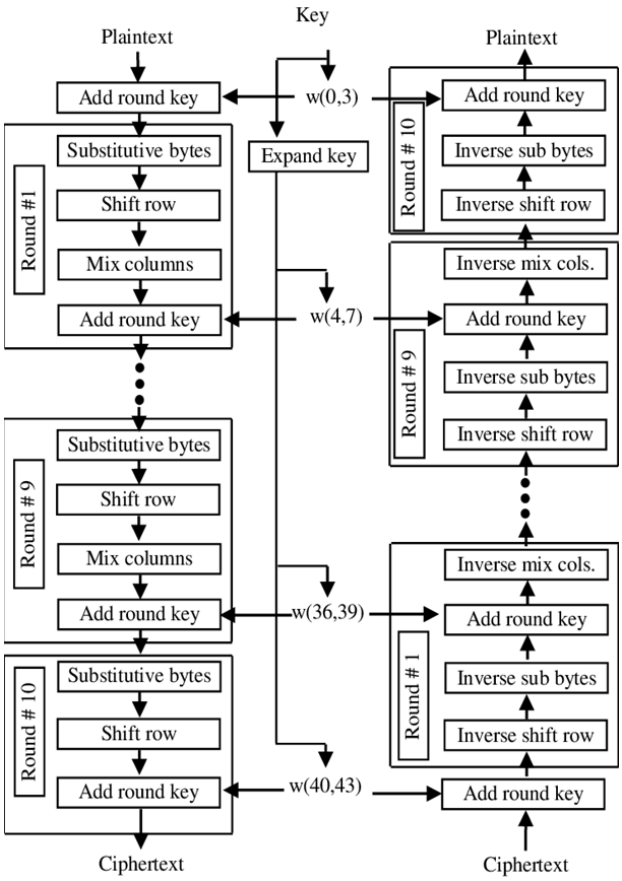
A ve B Kısmı:

Programlama Projesi için istenen AES şifreleme algoritmasının gerçekleştirilmesi ve şifreleme/şifre çözme(deşifreleme) için test verileri ile kullanılması, ayrıca uygulanan simetrik şifreleme algoritması (AES) kullanılarak CBC ve OFB modlarında çalışmayı gerçekleyip testlerinin yazılması Piton (Python) programlama dili kullanılarak yazılmıştır. Aşağıda yapılan işlemler ayrıntılı olarak anlatılmıştır.

AES Şifreleme Algoritması Uygulanması:

AES (Advanced Encryption Standard; Gelişmiş Şifreleme Standardı), elektronik verinin şifrelenmesi için sunulan bir standarttır. AES ile tanımlanan şifreleme algoritması , hem şifreleme hem de şifreli metni çözmede kullanılan anahtarların birbiriyle ilişkili olduğu, simetrik-anahtarlı bir algoritmadır. AES için şifreleme ve şifre çözme anahtarları aynıdır.

Projenin çözümünde AES Rijndael algoritması uygulanmıştır. Bu algoritma için 128,192,256 bitlik anahtarlar ve 128 bit veri kullanılmaktadır. AES Rijndael Algoritması şifreleme ve deşifreleme (şifre çözme) işlemi için uygulanırken aşağıda resmi gösterilen yapı kullanılmıştır:



AES Şifreleme – Şifre Çözme Algoritması Yapısı

Bu algoritmanın uygulanması için AES turlarında gerçekleştirilen işlemlerin yapıldığı Bir sınıf(class) yazıldı. Bu sınıf içerisinde AES bir turu için gerekli olan aşağıdaki yöntemler yapılmıştır:

- Baytları yerine koyma (Her bayt için bir S-Box (S Kutusu) kullanılması)
- Satırları kaydırma (Sütunlar arası permütasyon (yer değiştirme) işlemi yapılması)
- Sütunları karıştırma (Sütunlar arası yerine koyma (substitution) işlemi yapılması)
- Tur anahtarı ekleme (Anahtar ile XOR işlemi yapılması)

Ayrıca şifreleme ve deşifreleme (şifre çözme) yaparken kullanılan anahtar genişletme işlemi de yapılmıştır.

AES uygulanmasında kullanılan S-Box, R-Box, terslenmiş S-Box aşağıda gösterilmiştir:

```
#Class that does AES operations.
class AES(object):
    #Rijndael S-box. This is created with GF(2^8)
    s_box =  [0x63, 0x7c, 0x77, 0x7b, 0xf2, 0x6b, 0x6f, 0xc5, 0x30, 0x01, 0x67,
              0x2b, 0xfe, 0xd7, 0xab, 0x76, 0xca, 0x82, 0xc9, 0x7d, 0xfa, 0x59,
              0x47, 0xf0, 0xad, 0xd4, 0xa2, 0xaf, 0x9c, 0xa4, 0x72, 0xc0, 0xb7,
              0xfd, 0x93, 0x26, 0x36, 0x3f, 0xf7, 0xcc, 0x34, 0xa5, 0xe5, 0xf1,
              0x71, 0xd8, 0x31, 0x15, 0x04, 0xc7, 0x23, 0xc3, 0x18, 0x96, 0x05,
              0x9a, 0x07, 0x12, 0x80, 0xe2, 0xeb, 0x27, 0xb2, 0x75, 0x09, 0x83,
              0x2c, 0x1a, 0x1b, 0x6e, 0x5a, 0xa0, 0x52, 0x3b, 0xd6, 0xb3, 0x29,
              0xe3, 0x2f, 0x84, 0x53, 0xd1, 0x00, 0xed, 0x20, 0xfc, 0xb1, 0x5b,
              0x6a, 0xcb, 0xbe, 0x39, 0x4a, 0x4c, 0x58, 0xcf, 0xd0, 0xef, 0xaa,
              0xfb, 0x43, 0x4d, 0x33, 0x85, 0x45, 0xf9, 0x02, 0x7f, 0x50, 0x3c,
              0x9f, 0xa8, 0x51, 0xa3, 0x40, 0x8f, 0x92, 0x9d, 0x38, 0xf5, 0xbc,
              0xb6, 0xda, 0x21, 0x10, 0xff, 0xf3, 0xd2, 0xcd, 0x0c, 0x13, 0xec,
              0x5f, 0x97, 0x44, 0x17, 0xc4, 0xa7, 0x7e, 0x3d, 0x64, 0x5d, 0x19,
              0x73, 0x60, 0x81, 0x4f, 0xdc, 0x22, 0x2a, 0x90, 0x88, 0x46, 0xee,
              0xb8, 0x14, 0xde, 0x5e, 0x0b, 0xdb, 0xe0, 0x32, 0x3a, 0x0a, 0x49,
              0x06, 0x24, 0x5c, 0xc2, 0xd3, 0xac, 0x62, 0x91, 0x95, 0xe4, 0x79,
              0xe7, 0xc8, 0x37, 0x6d, 0x8d, 0xd5, 0x4e, 0xa9, 0x6c, 0x56, 0xf4,
              0xea, 0x65, 0x7a, 0xae, 0x08, 0xba, 0x78, 0x25, 0x2e, 0x1c, 0xa6,
              0xb4, 0xc6, 0xe8, 0xdd, 0x74, 0x1f, 0x4b, 0xbd, 0x8b, 0x8a, 0x70,
              0x3e, 0xb5, 0x66, 0x48, 0x03, 0xf6, 0x0e, 0x61, 0x35, 0x57, 0xb9,
              0x86, 0xc1, 0x1d, 0x9e, 0xe1, 0xf8, 0x98, 0x11, 0x69, 0xd9, 0x8e,
              0x94, 0x9b, 0x1e, 0x87, 0xe9, 0xce, 0x55, 0x28, 0xdf, 0x8c, 0xa1,
              0x89, 0x0d, 0xbf, 0xe6, 0x42, 0x68, 0x41, 0x99, 0x2d, 0x0f, 0xb0,
              0x54, 0xbb, 0x16]
```

S-Box

```
#Rijndael inverted S-box
rsbox = [0x52, 0x09, 0x6a, 0xd5, 0x30, 0x36, 0xa5, 0x38, 0xbf, 0x40, 0xa3,
         0x9e, 0x81, 0xf3, 0xd7, 0xfb , 0x7c, 0xe3, 0x39, 0x82, 0x9b, 0x2f,
         0xff, 0x87, 0x34, 0x8e, 0x43, 0x44, 0xc4, 0xde, 0xe9, 0xcb , 0x54,
         0x7b, 0x94, 0x32, 0xa6, 0xc2, 0x23, 0x3d, 0xee, 0x4c, 0x95, 0x0b,
         0x42, 0xfa, 0xc3, 0x4e , 0x08, 0x2e, 0xa1, 0x66, 0x28, 0xd9, 0x24,
         0xb2, 0x76, 0x5b, 0xa2, 0x49, 0x6d, 0x8b, 0xd1, 0x25 , 0x72, 0xf8,
         0xf6, 0x64, 0x86, 0x68, 0x98, 0x16, 0xd4, 0xa4, 0x5c, 0xcc, 0x5d,
         0x65, 0xb6, 0x92 , 0x6c, 0x70, 0x48, 0x50, 0xfd, 0xed, 0xb9, 0xda,
         0x5e, 0x15, 0x46, 0x57, 0xa7, 0x8d, 0x9d, 0x84 , 0x90, 0xd8, 0xab,
         0x00, 0x8c, 0xbc, 0xd3, 0x0a, 0xf7, 0xe4, 0x58, 0x05, 0xb8, 0xb3,
         0x45, 0x06 , 0xd0, 0x2c, 0x1e, 0x8f, 0xca, 0x3f, 0x0f, 0x02, 0xc1,
         0xaf, 0xbd, 0x03, 0x01, 0x13, 0x8a, 0x6b , 0x3a, 0x91, 0x11, 0x41,
         0x4f, 0x67, 0xdc, 0xea, 0x97, 0xf2, 0xcf, 0xce, 0xf0, 0xb4, 0xe6,
         0x73 , 0x96, 0xac, 0x74, 0x22, 0xe7, 0xad, 0x35, 0x85, 0xe2, 0xf9,
         0x37, 0xe8, 0x1c, 0x75, 0xdf, 0x6e , 0x47, 0xf1, 0x1a, 0x71, 0x1d,
         0x29, 0xc5, 0x89, 0x6f, 0xb7, 0x62, 0x0e, 0xaa, 0x18, 0xbe, 0x1b ,
         0xfc, 0x56, 0x3e, 0x4b, 0xc6, 0xd2, 0x79, 0x20, 0x9a, 0xdb, 0xc0,
         0xfe, 0x78, 0xcd, 0x5a, 0xf4 , 0x1f, 0xdd, 0xa8, 0x33, 0x88, 0x07,
         0xc7, 0x31, 0xb1, 0x12, 0x10, 0x59, 0x27, 0x80, 0xec, 0x5f , 0x60,
         0x51, 0x7f, 0xa9, 0x19, 0xb5, 0x4a, 0x0d, 0x2d, 0xe5, 0x7a, 0x9f,
         0x93, 0xc9, 0x9c, 0xef , 0xa0, 0xe0, 0x3b, 0x4d, 0xae, 0x2a, 0xf5,
         0xb0, 0xc8, 0xeb, 0xbb, 0x3c, 0x83, 0x53, 0x99, 0x61 , 0x17, 0x2b,
         0x04, 0x7e, 0xba, 0x77, 0xd6, 0x26, 0xe1, 0x69, 0x14, 0x63, 0x55,
         0x21, 0x0c, 0x7d]
```

Terslenmiş S-Box

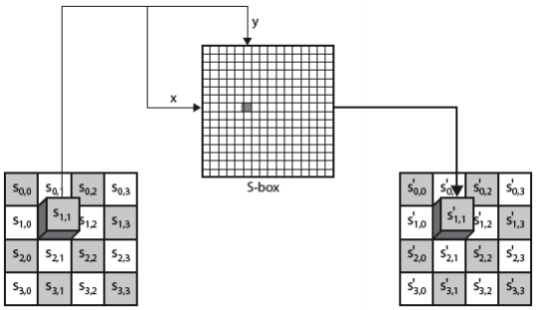
Baytları Yerine Koyma İşlemi:

İlk işlem bayt değiştirmedir. Durum matrisinin her elemanı, değerleri önceden hesaplanarak oluşturulmuş S-kutusundaki değerlerle değiştirilir. Her durum baytı, satır (sol 4 – bit) ve sütun (sağ 4 – bit) ile indekslenmiş bayt ile değiştirilir. GF (2^8) 'de değerlerin tanımlanmış dönüşümü kullanılarak inşa edilmiş S – Box kullanılır. AES uygulanırken yazılan baytları yerine koyma işlemi aşağıda gösterilmiştir:

```
# Substitutiton of all the values from the state with the value in the SBox using the state value as index for the SBox.
def byte_substitution(self,  situation, inverse):
    if inverse:
        getter = self.getting_SBox_Invert
    else:
        getter = self.getting_SBox
    i=0
    while(i<16):
        situation[i] = getter( situation[i])
        i=i+1
    return situation
```

Bayt Yerine Koyma İşlemi Uygulanması

Baytları yerine koyma işlemi resim ile gösterilirse:



Bayt yerine koyma işlemi

Satırları Kaydırma:

Satır kaydırma işleminde satırlar sırasıyla çevrimsel şekilde kaydırılırlar. Yani ilk satır değiştirilmez, ikinci satır da sola 1 ötelenir, üçüncü satır sola 2 ötelenir ve son satır sola 3 ötelenir. Taşan bölmeler kaydırmanın başına eklenir. Durum sütunlar tarafından işlendiğinden, bu adım sütunlar arasındaki baytların permütasyonunu (yer değiştirme) sağlar. AES uygulanırken yazılan satırları kaydırma işlemi aşağıda gösterilmiştir:

```
#One Shifting row operation.Calling in every iteration shifts the row to the left by one.
def shift_Row(self,  situation, pointer_table, number, inverse):
    i=0;
    while(i<number):
        if inverse:
            situation[pointer_table:pointer_table+4] = \
                    situation[pointer_table+3:pointer_table+4] + \
                    situation[pointer_table:pointer_table+3]
        else:
            situation[pointer_table:pointer_table+4] = \
                    situation[pointer_table+1:pointer_table+4] + \
                    situation[pointer_table:pointer_table+1]
        i=i+1
    return  situation

# Shifting rows with four iteration.
def shift_Rows(self, situation,is_inverse):
    i=0
    while(i<4):
        situation = self.shift_Row( situation, i*4, i, is_inverse)
        i=i+1
    return situation
```

Satır Kaydırma İşlemi Uygulanması

Satır kaydırma işlemi uygulanırken bir döngü içerisinde tek tek tüm satırlar için kaydırma işlemi gerçekleştirilir.

Satır kaydırma işlemi resim ile gösterilirse:

Satır Kaydırma İşlemi

Sütunları Karıştırma:

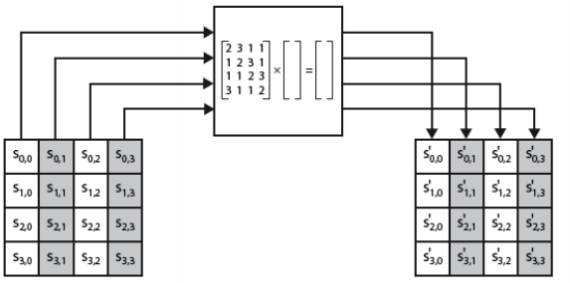
Bu işlemde eski sütunun elemanları kullanılarak yeni sütun elde edilmektedir. Bu yapılırken yeni sütunun elemanları eski sütunun her elemanı hesaba katılarak tek tek hesaplanır. GF(2^8) matris çarpma işlemi uygulanır. AES uygulanırken yazılan sütunları karıştırma işlemi aşağıda gösterilmiştir:

```
# Galois multiplication of 1 column of the 4x4 matrix.
def mixColumn(self, column, isInv):
    if isInv:
        mult = [14, 9, 13, 11]
    else:
        mult = [2, 1, 1, 3]
    keep_column = list(column)
    g = self.galois_multiplication

    column[0] = g(keep_column[0], mult[0]) ^ g(keep_column[3], mult[1]) ^ \
                g(keep_column[2], mult[2]) ^ g(keep_column[1], mult[3])
    column[1] = g(keep_column[1], mult[0]) ^ g(keep_column[0], mult[1]) ^ \
                g(keep_column[3], mult[2]) ^ g(keep_column[2], mult[3])
    column[2] = g(keep_column[2], mult[0]) ^ g(keep_column[1], mult[1]) ^ \
                g(keep_column[0], mult[2]) ^ g(keep_column[3], mult[3])
    column[3] = g(keep_column[3], mult[0]) ^ g(keep_column[2], mult[1]) ^ \
                g(keep_column[1], mult[2]) ^ g(keep_column[0], mult[3])
    return column
# Mix colum operation.Actually this operation is matrix multiplication in GF(2^8)
def mix_Columns(self, state, is_inverse):
    i=0
    #loop over four column.
    while(i<4):
        column = state[i:i+16:4]
        column = self.mixColumn(column, is_inverse)  # applying the mixColumn on one column.
        state[i:i+16:4] = column
        i=i+1
    return state
```

Sütunları Karıştırma

Sütunları Karıştırma yönteminde yardımcı yöntem kullanılmıştır. Yardımcı yöntem 1 sütunu matris ile çarpmaktadır. Döngü içerisinde tüm sütunlar matris ile çarpılır.

Sütun karıştırma işlemi resim ile gösterilirse:



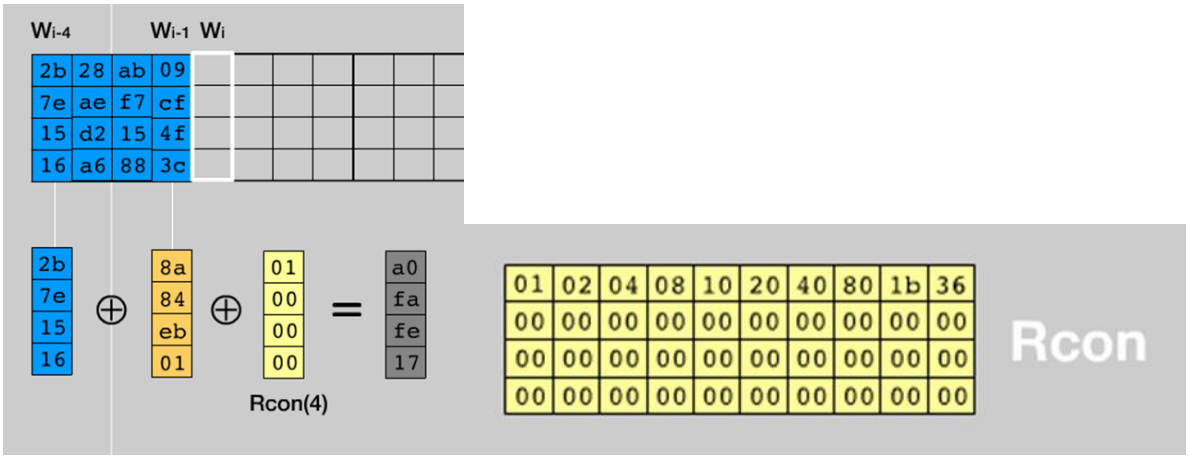
Sütun Karıştırma

Tur Anahtarı Ekleme: Her turda daha önce anlatılan işlemlerle birlikte tur anahtarı oluşturma işlemi yapılmaktadır ve her turda sonuçta oluşan durum ile o tur için hazırlanmış olan yeni anahtar toplama işlemine tabi tutulur. Bahsedilen toplama işlemi XOR işlemi yapılmasıdır. AES uygulanırken yazılan tur anahtarı oluşturma ve tur anahtarı ekleme işlemi aşağıda gösterilmiştir:

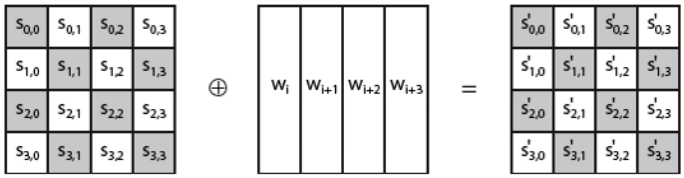
```
#Creation of round key. This key created from given expanded key and position.
def create_round_key(self, expandedKey, roundKey_Pointer):
    round_Key = [0] * 16
    for i in range(4):
        for j in range(4):
            round_Key[j*4+i] = expandedKey[roundKey_Pointer + i*4 + j]
    return round_Key

#For the state, this method XOR's the round key.
def add_round_key(self, state, roundKey):
    i=0;
    while(i<16):
        state[i] ^= roundKey[i]
        i=i+1
    return state
```

Tur anahtarı oluşturma ve tur anahtarı ekleme

Tur anahtarı oluşumu ve eklemesi resim ile gösterilirse:



Anahtar Oluşumu



Tur anahtarı ekleme

Anahtar Genişletme: 128 bit anahtar 4-bit-kelime, 192 bit anahtar 6-bit-kelime, 256 bit anahtar 8-bit-kelime olarak genişletilir. Anahtar döndürme, XOR işlemleri yapılarak uygulanır. AES uygulanırken yazılan anahtar zamanlama, anahtar döndürme ve anahtar genişletme işlemleri aşağıda gösterilmiştir.

```
#key schedule rotate operation of Rijndael.
def rotation(self, word):
    return word[1:] + word[:1]

#Key scheduling  method.
def keyScheduling(self, word, iteration):
    word = self.rotation(word)  # rotation the 32-bit word 8 bits to the left.
    # applying S-Box substitution.
    for i in range(4):
        word[i] = self.getting_SBox(word[i])
    word[0] = word[0] ^ self.getting_Rcon(iteration) # XOR the output of the rcon operation.
    return word
```

Anahtar döndürme ve zamanlama

```python
#key expansion of Rinjdael.
def expandKey(self, key, size, expanded_Key_Size):
    currentSize = 0
    iteration_rron= 1
    expandedKey = [0] * expanded_Key_Size

    # setting the bytes that are 16, 24, 32 of the expanded key to the input key.
    for j in range(size):
        expandedKey[j] = key[j]
    currentSize += size

    while currentSize < expanded_Key_Size:
        temp = expandedKey[currentSize-4:currentSize]     #assigning the previous 4 bytes to the temporary value.

        # Each  bytes applying the key schedule to temp and increment iteration.
        if currentSize % size == 0:
            temp = self.keyScheduling(temp, iteration_rron)
            iteration_rron += 1
        #For 256-bit keys,add an extra Sbox.
        if size == self.size_of_key["SIZE_256"] and ((currentSize % size) == 16):
            for l in range(4): temp[l] = self.getting_SBox(temp[l])

        #XOR temp with the four-byte block 16,24,32 bytes before the new expanded key.
        for m in range(4):
            expandedKey[currentSize] = expandedKey[currentSize - size] ^ \
                    temp[m]
            currentSize += 1

    return expandedKey
```

Anahtar Genişletme

AES için tüm yapılan bu işlemlerin uygulanması sırasıyla aşağıdaki şekilde çağırılmıştır.

```python
# Appling the four operation for AES round
def round_of_AES(self, state, roundKey):
    state = self.byte_substitution(state, False)
    state = self.shift_Rows(state, False)
    state = self.mix_Columns(state, False)
    state = self.add_round_key(state, roundKey)
    return state


# Perform the AES steps.
def AES_operation(self, state, expandedKey, nbrRounds):
    state = self.add_round_key(state, self.create_round_key(expandedKey, 0))
    i = 1
    while i < nbrRounds:
        state = self.round_of_AES(state,self.create_round_key(expandedKey, 16*i))
        i += 1
    state = self.byte_substitution(state, False)
    state = self.shift_Rows(state, False)
    state = self.add_round_key(state,self.create_round_key(expandedKey, 16*nbrRounds))
    return state
```

AES tur işlemlerinin bir yöntem içinde çağırılması.

AES şifreleme yöntemi için ilk önce anahtar genişletilmiş, daha sonra yapılan tüm işlemlerin kullanıldığı yöntem çağırılmıştır ve böylece şifreleme işlemi yapılmıştır. Gösterilecek olunursa:

```python
# Encryption a 128 bit input block with given key size.
def encrypt(self, message_input, key, size):
    numberofRounds = 0
    block = [0] * 16
    output = [0] * 16
    # setting the number of rounds.
    if size == self.size_of_key["SIZE_128"]:
        numberofRounds = 10
    elif size == self.size_of_key["SIZE_192"]:
        numberofRounds = 12
    elif size == self.size_of_key["SIZE_256"]:
        numberofRounds = 14
    else: return None

    expandedKeySize = 16*(numberofRounds+1)  # expanded key size

    # iterate over the columns and rows
    for i in range(4):
        for j in range(4):
            block[(i+(j*4))] = message_input[(i*4)+j]

    expandedKey = self.expandKey(key, size, expandedKeySize)    #expanded key
    block = self.AES_operation(block, expandedKey, numberofRounds)   # encrypt the block using the expandedKey

    # unmapping the block again into the output.
    for k in range(4):
        for l in range(4):
            output[(k*4)+l] = block[(k+(l*4))]

    return output
```

AES Şifreleme Yöntemi

Deşifreleme (Şifre Çözme) Tüm işlemlerin bir yöntem altında çağırıldığı yöntem aşağıda gösterilmektedir. Şifre çözme işleminde terslenmiş S-Box kullanılmaktadır.

```python
# appling the 4 operations of the inverse round.
def AES_inverse_round(self, state, roundKey):
    state = self.shift_Rows(state, True)
    state = self.byte_substitution(state, True)
    state = self.add_round_key(state, roundKey)
    state = self.mix_Columns(state, True)
    return state

# Perform the inverse AES.
def AES_inverse_operation(self, state, expandedKey, nbrRounds):
    state = self.add_round_key(state,self.create_round_key(expandedKey, 16*nbrRounds))
    i = nbrRounds - 1
    while i > 0:
        state = self.AES_inverse_round(state,self.create_round_key(expandedKey, 16*i))
        i -= 1
    state = self.shift_Rows(state, True)
    state = self.byte_substitution(state, True)
    state = self.add_round_key(state, self.create_round_key(expandedKey, 0))
    return state
```

```python
# Decription a 128 bit input block with given key size.
def decrypt(self, message_input, key, size):
    numberofRounds = 0
    block = [0] * 16
    output = [0] * 16
    # setting the number of rounds
    if size == self.size_of_key["SIZE_128"]:
        numberofRounds = 10
    elif size == self.size_of_key["SIZE_192"]:
        numberofRounds = 12
    elif size == self.size_of_key["SIZE_256"]:
        numberofRounds = 14
    else: return None

    expandedKeySize = 16*( numberofRounds+1)   #expanded key size

    # iteration over the columns and rows.
    for i in range(4):
        for j in range(4):
            block[(i+(j*4))] = message_input[(i*4)+j]

    expandedKey = self.expandKey(key, size, expandedKeySize) #expanded key.
    block = self.AES_inverse_operation(block, expandedKey,  numberofRounds)    # decryption of the block using the expandedKey.
    # unmapping the block again into the output.
    for k in range(4):
        for l in range(4):
            output[(k*4)+l] = block[(k+(l*4))]
    return output
```

Deşifreleme(Şifre Çözme) Yöntemi

B KISMI - Mod Uygulamaları :

Modların uygulanması için bir sınıf yazılmış ve AES şifreleme – deşifreleme (şifre çözme) algoritması kullanılarak mod işlemleri yapılmıştır.

B KISMI – CBC Modu Uygulanması:

Mesaj bloklara bölünür, şifreleme işleminde birbirine bağlanır, önceki her şifre bloğu mevcut düz metin bloğu ile zincirlenir, işlemi başlatmak için İlk Vektör (IV) kullanılır.

$Ci = DES_{K1}(P_i \text{ XOR } C_{i-1})$

$C_{-1} = IV$

Aşağıda uygulanan, CBC şifreleme ve şifre çözme işlemleri gösterilmiştir.

```
if mode == self.modeOfOperation["CBC"]:
    for i in range(16):
        if firstRound:
            text_input[i] =  plaintext[i] ^ initialVector[i]
        else:
            text_input[i] =  plaintext[i] ^ ciphertext[i]
    firstRound = False
    ciphertext = self.aes.encrypt(text_input, key, size)
    for k in range(16):
        cipherOut.append(ciphertext[k])
```

CBC Şifreleme Kısmı

```
elif mode == self.modeOfOperation["CBC"]:
    output = self.aes.decrypt(ciphertext, key, size)
    for i in range(16):
        if firstRound:
            plaintext[i] = initialVector[i] ^ output[i]
        else:
            plaintext[i] = text_input[i] ^ output[i]
    firstRound = False
    if originalsize is not None and originalsize < end:
        for k in range(originalsize-start):
            out.append(chr(plaintext[k]))
    else:
        for k in range(end-start):
            out.append(chr(plaintext[k]))
    text_input = ciphertext
```

CBC Deşifreleme(Şifre Çözme) Kısmı

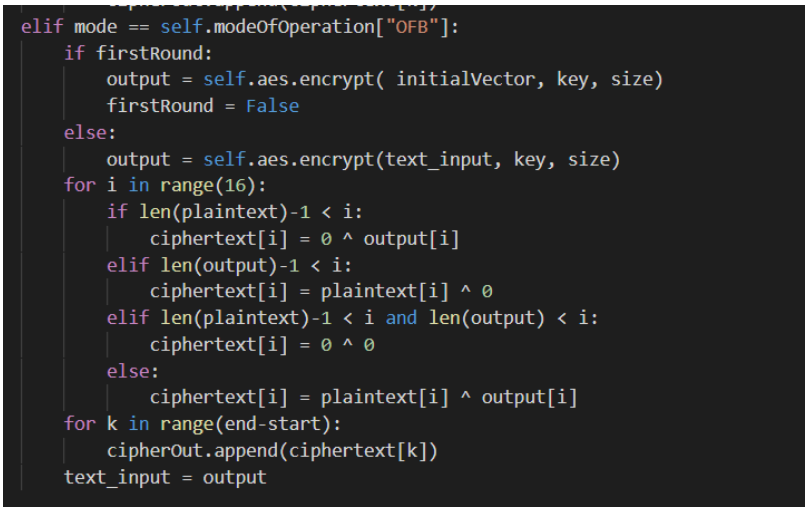
B KISMI – OFB Modu Uygulanması: Mesaj bir bit dizisi olarak değerlendirilir. Mesaja şifre çıkışı eklenir. Çıktı daha sonra geri beslenir. (Geri bildirim mesajdan bağımsızdır.)

$C_i = P_i$ XOR O_i

$O_i = DES_{K1}(O_{i-1})$

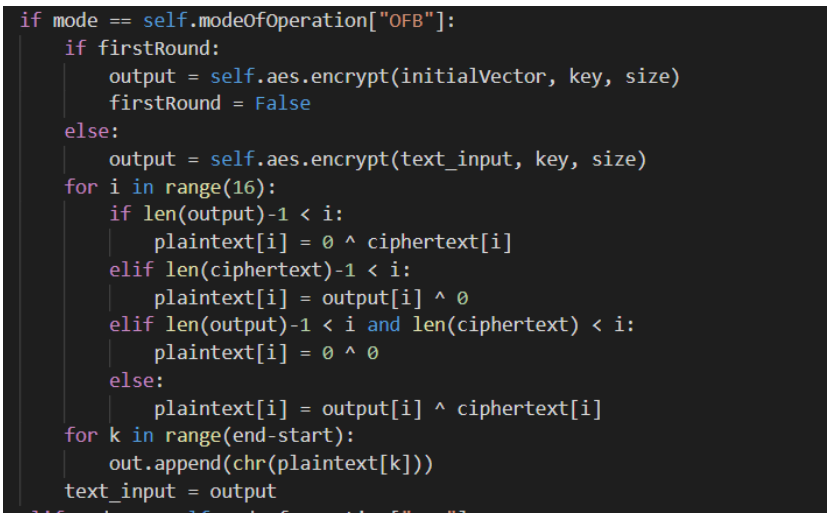
$O_{-1} = IV$

Aşağıda uygulanan, OFB şifreleme ve şifre çözme işlemleri gösterilmiştir.



```
elif mode == self.modeOfOperation["OFB"]:
    if firstRound:
        output = self.aes.encrypt( initialVector, key, size)
        firstRound = False
    else:
        output = self.aes.encrypt(text_input, key, size)
    for i in range(16):
        if len(plaintext)-1 < i:
            ciphertext[i] = 0 ^ output[i]
        elif len(output)-1 < i:
            ciphertext[i] = plaintext[i] ^ 0
        elif len(plaintext)-1 < i and len(output) < i:
            ciphertext[i] = 0 ^ 0
        else:
            ciphertext[i] = plaintext[i] ^ output[i]
    for k in range(end-start):
        cipherOut.append(ciphertext[k])
    text_input = output
```

OFB Şifreleme Kısmı



```
if mode == self.modeOfOperation["OFB"]:
    if firstRound:
        output = self.aes.encrypt(initialVector, key, size)
        firstRound = False
    else:
        output = self.aes.encrypt(text_input, key, size)
    for i in range(16):
        if len(output)-1 < i:
            plaintext[i] = 0 ^ ciphertext[i]
        elif len(ciphertext)-1 < i:
            plaintext[i] = output[i] ^ 0
        elif len(output)-1 < i and len(ciphertext) < i:
            plaintext[i] = 0 ^ 0
        else:
            plaintext[i] = output[i] ^ ciphertext[i]
    for k in range(end-start):
        out.append(chr(plaintext[k]))
    text_input = output
```

OFB Şifre Çözme Kısmı

A ve B Kısmı İçin Test Verileri ile Test Çıktıları:

```
C:\Users\Nevra Gürses\Desktop>python2 AES.py

~~~~~~~~~~~~~~~~~~~~~~~~~~
Sifrelenecek Metin:  Bu AES ve CBC modu ile sifrelenecek metindir.
Mod:  CBC
Anahtar Boyutu: 16 bayt
Rastgele Anahtar: [188, 193, 17, 248, 189, 187, 136, 236, 64, 210, 153, 120, 232, 104, 67, 124]
Sifrelenmis Metin:  [196, 150, 101, 31, 184, 53, 140, 192, 125, 129, 33, 218, 170, 227, 139, 4, 79, 111, 162, 102, 212, 243, 65, 93, 40, 237, 22, 8, 89, 52, 112, 12, 212, 1
10, 30, 158, 246, 238, 3, 34, 23, 240, 19, 31, 211, 183, 110, 212, 31, 217, 88, 69, 211, 12, 47, 150, 74, 59, 169, 212, 59, 113, 1, 214]
Sifresi Cozulmus(desifrelenmis) Metin:  Bu AES ve CBC modu ile sifrelenecek metindir.
~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~
Sifrelenecek Metin:  Bu AES ve OFB modu ile sifrelenecek metindir.
Mod:  OFB
Anahtar Boyutu: 16 bayt
Rastgele Anahtar: [188, 193, 17, 248, 189, 187, 136, 236, 64, 210, 153, 120, 232, 104, 67, 124]
Sifrelenmis Metin:  [39, 36, 182, 82, 244, 84, 242, 22, 74, 90, 118, 124, 9, 208, 215, 76, 112, 74, 210, 53, 86, 55, 245, 195, 6, 40, 72, 127, 203, 111, 174, 204, 182, 188,
 186, 249, 197, 176, 128, 56, 196, 38, 25, 159, 216, 22, 255, 8, 24, 187, 72, 240, 52, 237, 212, 125, 167, 53, 22, 209, 96, 219, 22, 206]
Sifresi Cozulmus(desifrelenmis) Metin:  Bu AES ve OFB modu ile sifrelenecek metindir.
~~~~~~~~~~~~~~~~~~~~~~~~~~
```

16 Bayt (128 Bit) Anahtar ile CBC ve OFB Modları Şifreleme – Şifre Çözme

```
~~~~~~~~~~~~~~~~~~~~~~~~~~
Sifrelenecek Metin:  Bu AES ve CBC modu ile sifrelenecek metindir.
Mod:  CBC
Anahtar Boyutu: 24 bayt
Rastgele Anahtar: [99, 150, 182, 7, 107, 241, 141, 61, 192, 188, 199, 191, 82, 43, 106, 206, 89, 67, 139, 154, 116, 154, 223, 221]
Sifrelenmis Metin:  [244, 5, 136, 128, 122, 184, 151, 35, 164, 189, 130, 167, 67, 99, 142, 91, 88, 65, 12, 243, 85, 169, 193, 12, 63, 230, 161, 253, 174, 247, 252, 16, 138,
 130, 8, 31, 158, 196, 37, 74, 112, 253, 42, 226, 164, 55, 99, 111, 101, 91, 172, 107, 47, 137, 101, 64, 107, 212, 242, 81, 183, 159, 123, 37]
Sifresi Cozulmus(desifrelenmis) Metin:  Bu AES ve CBC modu ile sifrelenecek metindir.
~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~
Sifrelenecek Metin:  Bu AES ve OFB modu ile sifrelenecek metindir.
Mod:  OFB
Anahtar Boyutu: 24 bayt
Rastgele Anahtar: [99, 150, 182, 7, 107, 241, 141, 61, 192, 188, 199, 191, 82, 43, 106, 206, 89, 67, 139, 154, 116, 154, 223, 221]
Sifrelenmis Metin:  [61, 99, 218, 74, 106, 57, 174, 33, 76, 132, 76, 72, 155, 177, 67, 153, 239, 54, 173, 133, 212, 160, 43, 221, 252, 22, 19, 32, 179, 138, 210, 232, 58, 9
5, 116, 68, 185, 38, 85, 124, 163, 208, 242, 63, 63, 120, 175, 8, 104, 24, 110, 53, 95, 210, 205, 157, 58, 244, 58, 185, 210, 163, 104, 88]
Sifresi Cozulmus(desifrelenmis) Metin:  Bu AES ve OFB modu ile sifrelenecek metindir.
~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~
```

24 Bayt (192 Bit) Anahtar ile CBC ve OFB Modları Şifreleme – Şifre Çözme

```
~~~~~~~~~~~~~~~~~~~~~~~~~~
Sifrelenecek Metin:  Bu AES ve CBC modu ile sifrelenecek metindir.
Mod:  CBC
Anahtar Boyutu: 32 bayt
Rastgele Anahtar: [234, 57, 237, 55, 16, 124, 41, 97, 160, 67, 236, 13, 106, 69, 94, 220, 159, 42, 160, 49, 214, 98, 252, 253, 236, 198, 153, 166, 32, 242, 42, 217]
Sifrelenmis Metin:  [170, 254, 123, 78, 91, 126, 31, 66, 227, 227, 85, 81, 196, 143, 237, 233, 54, 110, 161, 143, 33, 39, 188, 101, 46, 125, 169, 84, 243, 17, 217, 118, 221
, 133, 202, 92, 162, 125, 225, 142, 217, 91, 193, 29, 24, 216, 254, 225, 56, 30, 53, 215, 24, 70, 78, 226, 232, 43, 65, 196, 214, 101, 100, 152]
Sifresi Cozulmus(desifrelenmis) Metin:  Bu AES ve CBC modu ile sifrelenecek metindir.
~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~
Sifrelenecek Metin:  Bu AES ve OFB modu ile sifrelenecek metindir.
Mod:  OFB
Anahtar Boyutu: 32 bayt
Rastgele Anahtar: [234, 57, 237, 55, 16, 124, 41, 97, 160, 67, 236, 13, 106, 69, 94, 220, 159, 42, 160, 49, 214, 98, 252, 253, 236, 198, 153, 166, 32, 242, 42, 217]
Sifrelenmis Metin:  [142, 45, 184, 49, 10, 251, 69, 242, 160, 45, 159, 198, 106, 115, 109, 210, 228, 185, 67, 97, 145, 240, 110, 103, 72, 193, 86, 30, 128, 247, 49, 152, 20
9, 242, 34, 103, 103, 247, 247, 125, 130, 227, 109, 62, 53, 64, 188, 129, 235, 178, 253, 247, 73, 60, 200, 178, 137, 223, 112, 161, 188, 53, 155, 133]
Sifresi Cozulmus(desifrelenmis) Metin:  Bu AES ve OFB modu ile sifrelenecek metindir.
~~~~~~~~~~~~~~~~~~~~~~~~~~

C:\Users\Nevra Gürses\Desktop>
```

32 Bayt (256 Bit) Anahtar ile CBC ve OFB Modları Şifreleme – Şifre Çözme

C ve D Kısmı:

Herhangi bir doküman üzerinde değişiklik yapılıp yapılmadığını ve yapanın kimliğini anlamak için, özütünü alacak ve sadece işlem yapan kişinin bildiği bir anahtar ile AES şifreleme yöntemi kullanarak şifrelenip dosya sonuna eklenmesi ve dosyanın bütünlüğünün değişip değişmediğinin kontrolü için bir önceki aşamada yapılan işlemleri yaparak üretilen özüt değerin ilk üretilen özüt değer ile karşılaştırılması işlemi Piton(Python) programlama dili kullanılarak yapılmıştır. Aşağıda yapılan işlemler daha ayrıntılı şekilde anlatılmıştır.

Özüt Alma:

Özüt alma işlemi mesaj bloklarının XOR işlemine tabii tutulmasına dayanmaktadır. Herhangi bir boyut mesajına uygulanabilir. Sabit uzunlukta çıktı üretir. Projenin uygulanmasında kullanılmış olan özüt alma işlemi şu şekilde uygulanmıştır:

- Dosyadan tüm içerik okunmuştur.
- Bu içerik 32 karakter ve katı olacak şekilde genişletme işlemine tabii tutulmuştur. Bu genişletme işlemi a'dan z'ye ASCII karakterleri eklenerek yapılmıştır.
- Genişletilmiş içerik 32 karakterlik bloklara bölünmüştür. Bloklar yinelemeli olarak birbirleri ile XOR işlemine tabii tutulmuştur.
- Sonuçta 32 karakterlik özüt oluşmaktadır. Mesaj boyutu ne olursa olsun özüt uzunluğu uygulanan algoritmama göre 32 karakter uzunluğundadır.

Aşağıda bahsedilen özüt alma işlemi için yapılan içerik genişletme ve XOR işlemleri gösterilmektedir:

```
16  #for creating hash as 32 character, XOR operation is doing with expanded input.
17  def xorOperation(extended):
18      hashedMessage=[0]*32
19      if(len(extended)!=32):
20          #XOR operation with 2 loop.
21          for i in range(len(extended)/32):
22              for j in range(32):
23                  if(i==0):
24                      hashedMessage[j]=ord(extended[i*32+j])^ord(extended[(i+1)*32+j])
25                  elif(i>=2):
26                      hashedMessage[j]=hashedMessage[j]^ord(extended[(i*32)+j])
27      #If length is 32, this is hash. Convert characters to integer values.
28      elif(len(extended)==32):
29          for i in range(32):
30              hashedMessage[i] = ord(extended[i])
31      return hashedMessage
32
33  #Hash function. Before, expands the input. After, makes XOR operation.
34  def hash(intent):
35      expanded= expanding_with_32(intent)
36      hashedOutput = xorOperation(expanded)
37      return hashedOutput
38
39  #Encryption of hash with AES.
40  def cipherwithHash(hashedOutput,key):
41      cipherText = [str(i) for i in hashedOutput]
42      cipherText = ' '.join(cipherText )
43      cipher = encryptOperation(cipherText,"OFB",key)
44      return cipher
45
```

İçerik genişletme ve XOR işlemleri kullanılarak yapılan özüt alma işlemi

Bulunan özüt B kısmında kullanılan AES şifreleme ve OFB modu ile şifrelenmiş ve şifrelenme sonucu elde edilen şifre dosya sonuna yazılmıştır. Aşağıda bahsedilen bu işlemler gösterilmektedir.

```
39  #Encryption of hash with AES.
40  def cipherwithHash(hashedOutput,key):
41      cipherText = [str(i) for i in hashedOutput]
42      cipherText = ' '.join(cipherText )
43      cipher = encryptOperation(cipherText,"OFB",key)
44      return cipher
45
46
47  #Write cipher to the end of the file.
48  def writeCipher_to_File(file,hashedCipher):
49      fileName=open(file,"a")
50      cipherText = [str(i) for i in hashedCipher]
51      cipherText = ' '.join(cipherText)
52      fileName.write('\n')
53      fileName.write(cipherText)
54      fileName.close()
```

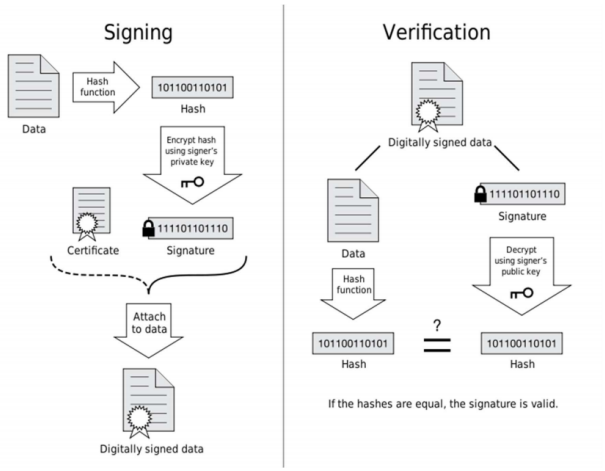
Bulunan özütün AES ile şifrelenmesi ve dosya sonuna yazılması

D – KISMI: Dosyanın Değişip Değişmediği Kontrolü:

Dosyanın değişip değişmediğinin kontrolü için dosyadan son satır yani şifre alınmaktadır. Bu alınan şifre AES deşifreleme(şifre çözme) yöntemi kullanılarak özüt elde edilmiştir.

Aynı zamanda dosya içerisinde şifrenin tutulduğu satır hariç tüm içeriğin tekrar özütü alınmıştır. Elde edilen bu iki özüt karşılaştırılmıştır. Eğer bu iki özüt aynıysa dosya değişmemiştir. Aşağıda bahsedilen bu işlem gösterilmektedir:

Yapılan kontrol işlemi, aşağıdaki resimdeki doğrulama kısmını oluşturmaktadır.



```python
#Function for controlling whether file contents changed or not.
#For controlling; decrtyption operation does for cipher in file, and also getting hash again file content.
#if this 2 result are equal, file does not changed, else file was changed.
def controlChanging(file,key):
    fileName=open(file,"r")
    Lines = fileName.readlines()
    cipherLine = Lines[-1]

    #convert integer array to string.
    cipherList = cipherLine.split(" ")
    cipherMessage= ""
    for i in range(0,len(cipherList)):
        cipherMessage+= chr(int(cipherList[i]))

    decryipted = decryptOperation(key,cipherMessage,"OFB") #decryption operation is doing.
   #getting file contents.
    Lines =Lines[:-1]
    clearText = ""
    count = 0
    newLastLine=Lines[-1]
    newLastLine = Lines[-1]
    for line in Lines:
        clearText += line.strip()
        if(line!= newLastLine):
            clearText += '\n'
        count+=1

    hashedOutput= hash(clearText) #hashing file contents.
    hashed = [str(i) for i in hashedOutput]
    hashed  = ' '.join(hashed)
    fileName.close()
    size = len(hashedOutput)
    #Controlling whether 2 hash equal or not.
    for i in range(size):
        if  decryipted[i]!=hashed[i]:
            return True
    return False
```

Dosya içeriğinin değişip değişmediğinin kontrolü

C ve D Kısmı İçin Test Verileri ile Test Çıktıları:

Dosya:

C: > Users > Nevra Gürses > Desktop > testFile.txt

```
merhabalar bu bir test dosyasıdır. Dosya üzerinde özüt alma ve şifreleme işlemi yapılacaktır.
daha sonra dosyanın değişip değişmediği kontrol edilecektir.
```

Özüt Alma İşlemi ve Şifreleme İşleminden Sonra Dosyaya Yazılması, Ardından Değişmeyen Dosya Kontrolü:

```
C:\Users\Nevra Gürses\Desktop>python2 hashing.py

~~~~~~TEST - 1:~~~~~~~~

Anahtar:  [1, 77, 201, 121, 154, 242, 126, 95, 115, 52, 127, 120, 172, 67, 246, 94, 203, 153, 223, 74, 41, 165, 209, 117, 193, 118, 78, 105, 195, 34, 127, 228]

Ozut Metin:  [231, 118, 93, 219, 237, 195, 96, 10, 225, 224, 6, 66, 44, 88, 74, 3, 190, 92, 237, 8, 1, 161, 214, 8, 13, 13, 186, 207, 78, 188, 105, 229]

Sifrelenmis Metin:  [244, 106, 86, 186, 52, 184, 132, 1, 85, 34, 222, 222, 190, 227, 171, 48, 116, 64, 231, 244, 231, 92, 248, 60, 82, 52, 96, 43, 78, 42, 131, 36, 226, 157
, 50, 209, 34, 55, 76, 102, 186, 46, 17, 209, 127, 117, 115, 58, 4, 70, 14, 36, 96, 213, 87, 199, 19, 59, 204, 85, 13, 122, 124, 191, 237, 54, 247, 47, 202, 58, 143, 46, 11
5, 238, 224, 185, 206, 54, 224, 28, 4, 120, 245, 226, 163, 83, 199, 221, 134, 226, 184, 6, 169, 86, 68, 118, 144, 193, 39, 27, 39, 26, 212, 164, 92, 4, 77, 146, 211, 222, 1
81, 176, 59, 146, 107, 245, 233, 241, 237, 211, 208, 196, 6, 23, 115, 121, 163, 74]

Dosya Kontrolu:  False
Sonuc: false, dosya degistirilmedi.
```

Özüt Alma İşlemi ve Şifreleme İşleminden Sonra Dosyaya Yazılması, Ardından Değişen Dosya Kontrolü:

```
~~~~~~TEST - 2:~~~~~~~~

Anahtar:  [27, 174, 115, 45, 0, 164, 200, 34, 29, 102, 219, 184, 68, 122, 38, 29, 183, 25, 35, 54, 105, 168, 84, 81, 227, 40, 41, 163, 242, 247, 221, 207]

Ozut Metin:  [250, 118, 93, 196, 235, 209, 105, 0, 252, 135, 64, 15, 54, 70, 69, 9, 172, 64, 253, 7, 13, 179, 216, 3, 18, 20, 184, 195, 76, 169, 111, 240]

Sifrelenmis Metin:  [204, 23, 181, 222, 16, 160, 36, 241, 157, 202, 150, 11, 222, 39, 113, 41, 19, 89, 42, 23, 171, 18, 249, 8, 158, 239, 135, 4, 166, 54, 49, 175, 207, 93,
 251, 36, 58, 229, 90, 116, 8, 29, 176, 119, 224, 222, 229, 133, 246, 134, 92, 190, 42, 191, 231, 115, 109, 111, 66, 42, 241, 250, 94, 123, 150, 106, 83, 36, 248, 35, 26, 2
48, 216, 121, 77, 237, 165, 125, 219, 30, 78, 207, 53, 178, 32, 135, 248, 18, 158, 165, 146, 112, 139, 56, 74, 146, 202, 236, 111, 21, 182, 45, 57, 145, 239, 110, 166, 51,
231, 45, 107, 137, 192, 59, 42, 119, 8, 200, 202, 211, 56, 27, 4, 25, 254, 226, 99, 199]

Dosya Kontrolu:  True
Sonuc: true, dosya degistirildi.

C:\Users\Nevra Gürses\Desktop>
```