KARADENİZ TEKNİK ÜNİVERSİTESİ

MÜHENDİSLİK FAKÜLTESİ

AĞ VE VERİ GÜVENLİĞİ
PROJE ÇALIŞMASI

GELENEKSEL SİMETRİK ŞİFRELEME YÖNTEMLERİNİN PYTHON DİLİ İLE GERÇEKLEŞTİRİLMESİ VE GÖRSELLEŞTİRİLMESİ

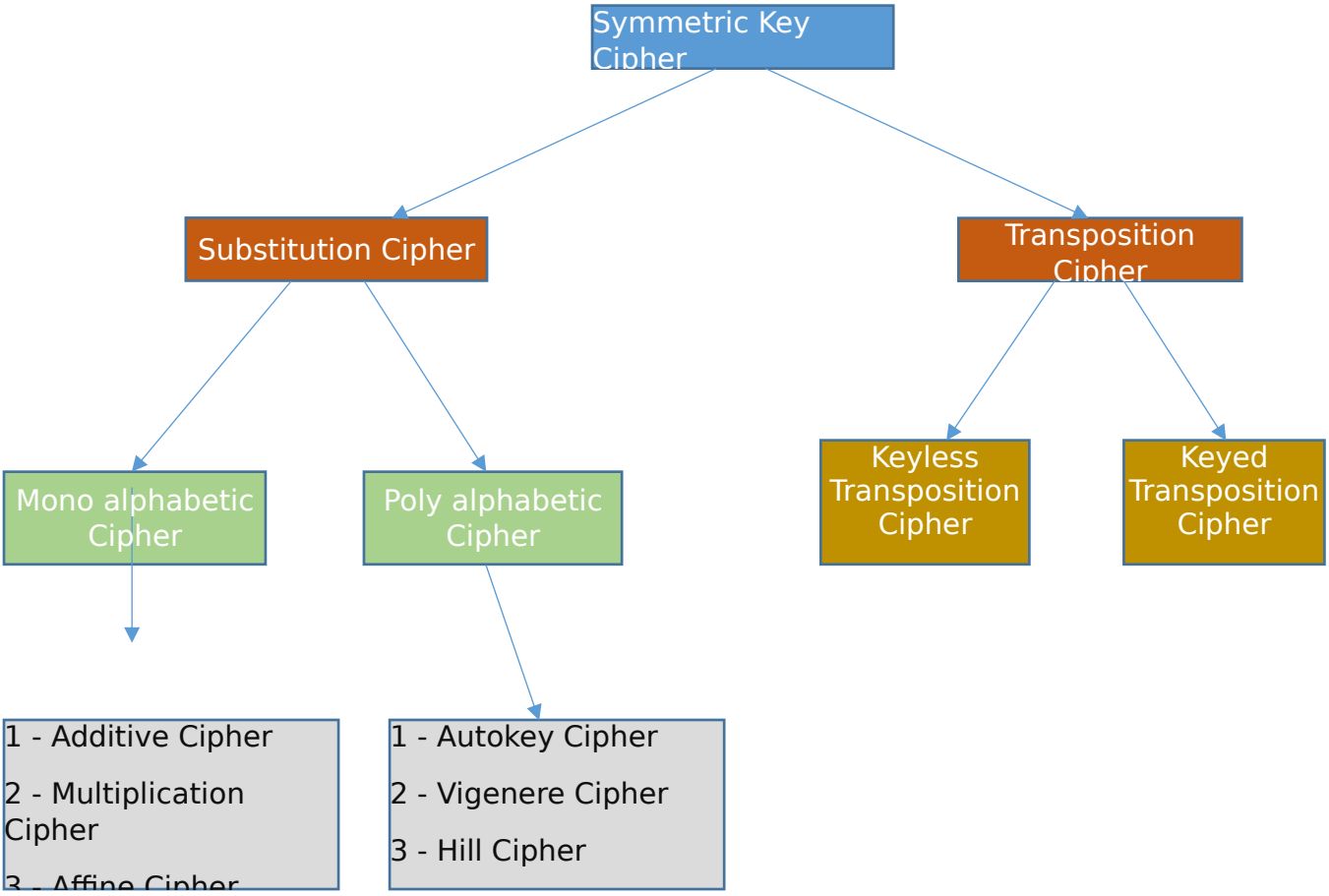
Fethi Erdinç UZUN
243961

Ahmet Faruk YAVUZ
243971

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ
ANABİLİM DALI
Trabzon

1. GİRİŞ

Aşağıdaki tabloda geleneksel simetrik şifreleme yöntemleri verilmiştir.



Tablo 1 - Geleneksel Simetrik Şifreleme Yöntemleri

2. PROJENİN AMACI

Projede aşağıdaki geleneksel simetrik şifreleme yöntemleri Python dili ile gerçekleştirilmiş olup grafik arayüzü için Tkinter arayüz tasarım takımı kullanılmıştır.

- Caesar cipher
- Multiplicative cipher
- Affine cipher
- Autokey cipher
- Playfair cipher
- Vigenere cipher
- Hill cipher
- Rail fence cipher

3. PROJENİN ÇALIŞTIRILMASI

Proje eğer Windows işletim sistemi üzerinde çalıştırılacaksa sadece Python'u kurmak yeterlidir. Tkinter hâlihazırda Python'a gömülü gelmektedir. Eğer GNU/Linux üzerinde çalıştırılacaksa Python'u kurduktan sonra Tkinter'in yüklenmesi gerekmektedir. Çünkü GNU/Linux dağıtımları genellikle Tkinter'i ayrı bir paket olarak sunmaktadır. Bu işlemler için Debian tabanlı dağıtımlarda aşağıdaki komut çalıştırılabilir.

```
sudo apt-get install python3 python3-tk
```

Python ve Tkinter'i yükledikten sonra projenin bulunduğu dizindeki *Traditional_Symmetric-Key_Cipher.py* dosyasını aşağıdaki komutla çalıştırmanız yeterlidir.

```
python3 Traditional_Symmetric-Key_Cipher.py
```

Açılan pencerede *Select Cipher Type* Spinner'inden şifreleme yöntemi seçilmektedir. Seçilen şifreleme yöntemine uygun anahtar değerleri *Input 1st Key* ve *Input 2nd Key* entry'lerine girilir. *Encryption* ve *Decryption* radiobutton'larından seçilen işleme göre şifreleme ya da deşifreleme yapılabilmektedir. Encryption işlemi seçili iken Soldaki metin kutusu *Plaintext*; sağdaki metin kutusu *Encrypted Text* isimlerini almaktadır ve *Encode* butonu seçilen şifreleme yöntemine göre şifreleme işlemini gerçekleştirmektedir. Decryption işlemi seçili iken soldaki metin kutusu *Encrypted Text*; sağdaki metin kutusu ise *Decrypted Text* isimlerini almaktadır ve *Decode* butonu seçilen şifreleme yöntemine göre deşifreleme işlemini gerçekleştirmektedir. Kodlanan program dosyadan metin okuyup, metni dosyaya yazabilme işlevlerine sahiptir.

Caesar Cipher

Tarihteki en eski şifreleme tekniği olarak bilinir. Caesar şifrelemesi yerine koyma tekniğini kullanır. Düz metindeki her harf, alfabedeki pozisyonunun bir anahtar değeri ile toplanarak elde edilen değerin alfabedeki karşılığı ile yer değiştirir. Örneğin, “ktuceng” kelimesi 3 anahtar değeri ile şifrelenirse ‘k’ harfinin yeni karşılığı ‘n’ olur. Şifreli metin ise “nwxfhqj” kelimesine karşılık düşer.

Programda *caesar_cipher()* fonksiyonu parametre olarak düz metni ve anahtar değerini alıp geriye şifrelenmiş metni döndürmektedir. *decrypt_caesar_cipher()* fonksiyonu parametre olarak verilen şifreli metni ve anahtar değerini alıp geriye düz metni döndürmektedir. Her iki fonksiyon da büyük harf küçük harf duyarlılığına sahiptir. Latin alfabesi haricinde girilen karakterler olduğu gibi çıkış metnine aktarılır.

```
def caesar_cipher(plainText, key):
    i = 0
    encryptedText = ""

    while i < len(plainText):
        char =  ord(plainText[i])
        if char >= 65 and char <= 90:
            char = (char%65 + key)%26 + 65

        elif char >= 97 and char <= 122:
            char = (char%97 + key)%26 + 97

        encryptedText += chr(char)
        i=i+1
    return encryptedText
```

```
def decrypt_caesar_cipher(encryptedText, key):
    i = 0
    decryptedText = ""
    while i < len(encryptedText):
        char = ord(encryptedText[i])
        if char >= 65 and char <= 90:
            char = (char%65 - key)%26 + 65

        elif char >= 97 and char <= 122:
            char = (char%97 - key)%26 + 97

        decryptedText += chr(char)
        i=i+1
    else:
        return decryptedText
```

Multiplicative Cipher

Bu şifreleme yönteminde düz metindeki her harfin alfabedeki pozisyonu ile anahtar değeri çarpılarak edilen sonucun mod 26’daki değeri, alfabedeki karşılığı ile yer değiştirir. Ancak seçilen anahtarın tersinin alınabilir olması için 26 ile aralarında asal özelliği göstermelidir. “ktuceng” kelimesi bu yönteme göre 3 anahtar değeri ile şifrelenirse ‘k’ harfinin yeni karşılığı ‘e’ olur. Şifreli metin ise “efigmns” kelimesine karşılık düşer.

Programda *multiplicative_cipher()* fonksiyonu parametre olarak düz metni ve anahtar değerini alıp geriye şifrelenmiş metni döndürmektedir. Anahtar değerinin 26 ile aralarında asal olup olmadığının kontrolünü *gcd()* fonksiyonu yapmaktadır. Deşifreleme için ise *decrypt_multiplicative_cipher()* fonksiyonu parametre olarak verilen şifreli metni ve anahtar değerini alıp geriye düz metni döndürmektedir. Girilen anahtar değerinin tersini *take_inverse()* fonksiyonu geriye döndürmektedir. Her iki fonksiyon da büyük harf küçük harf duyarlılığına sahiptir. Latin alfabesi haricinde girilen karakterler olduğu gibi çıkış metnine aktarılır.

```
def gcd(x, y):
    if y == 0:
        return x
    else:
        r = x % y
        return gcd(y, r)
```

```
def take_inverse(r1,r2):

    global s1,s2,s,t1,t2,t,q
    s1 = 1
    s2 = 0
    t1 = 0
    t2 = 1
    while not r2 == 0:
        r = r1 % r2
        q = (r1-r)/r2
        s = s1-q*s2
        t = t1-q*t2
        r1 = r2
        r2 = r
        s1 = s2
        s2 = s
        t1 = t2
        t2 = t

    return int(t1%26)
```

```
def multiplicative_cipher(plainText, key):
    i = 0
    encryptedText = ""

    if gcd(key,26) == 1:
        while i < len(plainText):
            char =  ord(plainText[i])
            if char >= 65 and char <= 90:
                char = (char%65 * key)%26 + 65

            elif char >= 97 and char <= 122:
                char = (char%97 * key)%26 + 97

            encryptedText += chr(char)
            i=i+1
        return encryptedText
    else:
         print("Sectiginiz sayi 26 ile aralarinda asal
degil..!")
              return "Error: Sectiginiz sayi 26 ile
aralarinda asal degil..!"
```

```
def
decrypt_multiplicative_cipher(encryptedText,
key):
    i = 0
    decryptedText = ""
    inverseOfKey = take_inverse(26,key)

    while i < len(encryptedText):
        char = ord(encryptedText[i])
        if char >= 65 and char <= 90:
            char = (char%65 * inverseOfKey)%26
+ 65

        elif char >= 97 and char <= 122:
            char = (char%97 * inverseOfKey)%26
+ 97

        decryptedText += chr(char)
        i=i+1
    else:
        return decryptedText
```

Affine Cipher

Bu şifreleme yöntemi Caesar ve Multiplicative şifreleme yöntemlerinin birleşimidir. Birincisi 26 ile aralarında asal olmak üzere iki anahtar değeri kullanır. Düz metne önce Multiplicative sonra Caesar şifreleme uygulanır. Örneğin, “ktuceng” kelimesi bu yönteme göre 17 ve 12 anahtar değerleri ile şifrelenirse şifreli metin “axouczk” kelimesine karşılık düşer.

Programda *affine_cipher()* fonksiyonu parametre olarak düz metni ve iki anahtar değerini alıp geriye şifrelenmiş metni döndürmektedir. İlk anahtar değerinin 26 ile aralarında asal olup olmadığının kontrolünü *gcd()* fonksiyonu yapmaktadır. Deşifreleme için ise *decrypt_affine_cipher()* fonksiyonu parametre olarak verilen şifreli metni ve anahtar değerlerini alıp geriye düz metni döndürmektedir. Her iki fonksiyon da büyük harf küçük harf duyarlılığına sahiptir. Latin alfabesi haricinde girilen karakterler olduğu gibi çıkış metnine aktarılır.

```
def affine_cipher(plainText, key1, key2):
    i=0
    encryptedText = ""

    if gcd(key1,26) == 1:
        while i < len(plainText):
            char =  ord(plainText[i])
            if char >= 65 and char <= 90:
                char = (char%65 * key1)%26
                char = (char + key2)%26 + 65

            elif char >= 97 and char <= 122:
                char = (char%97 * key1)%26
                char = (char + key2)%26 + 97

            encryptedText += chr(char)
            i=i+1
        return encryptedText
    else:
        print("Sectiginiz Key1 değeri 26 ile aralarinda asal degil..!")
        return "Error: Sectiginiz Key1 değeri 26 ile aralarinda asal degil..!"
```

```
def decrypt_affine_cipher(encryptedText, key1, key2):
    i = 0
    decryptedText = ""
    inverseOfKey = take_inverse(26,key1)

    while i < len(encryptedText):
        char = ord(encryptedText[i])
        if char >= 65 and char <= 90:
            char = (char%65 - key2)%26
            char = (char * inverseOfKey)%26 + 65

        elif char >= 97 and char <= 122:
            char = (char%97 - key2)%26
            char = (char * inverseOfKey)%26 + 97

        decryptedText += chr(char)
        i=i+1
    else:
        return decryptedText
```

Autokey Cipher

Autokey şifreleme yönteminde seçilen integer değerli anahtar değerinin peşine düz metin eklenir. Düz metin ile oluşturulan anahtar metin toplanır.

```
  h e l l o w o r l d
+ 13 h e l l o w o r l
```

Deşifreleme de ise şifrelenmiş metnin ilk harfinin indeks değerinden anahtar değeri çıkarılır. Elde edilen değer şifreli metnin bir sonraki indeks değerinden çıkarılarak işleme devam edilir. Çıkarma işleminin sonucundaki değerler düz metni verir.

autokey_cipher() fonksiyonu parametre olarak aldığı düz metni ve anahtar değerini kullanarak şifreli metni geriye döndürür. *decrypt_autokey_cipher()* fonksiyonu ise parametre olarak aldığı şifreli metni ve anahtar değerini kullanarak düz metni geriye döndürür.

Playfair Cipher

Playfair şifrelemesi I. Dünya Savaşında İngiliz ordusu tarafından kullanılmıştır. Şifreleme için kullanılan anahtar 25 karakterden oluşan 5x5 bir matristir. ‘i’ ve ‘j’ harfleri aynı karakter olarak kullanılır.

Şifrelemeden önce düz metinde art arda iki harf varsa bu iki harf arasına doldurma karakteri eklenir. Doldurma yapıldıktan sonra düz metnin uzunluğu tek ise ekstra bir doldurma karakteri daha eklenir. Şifreleme için aşağıdaki adımlar izlenir.

- Düz metin ikişerli gruplara ayrılır.
- Bu gruplardaki harfler anahtar matrisinde aynı satırda iseler bu harflerin bir sağındaki harfler alınır.
- Eğer aynı sütunda iseler bu harflerin bir altlarındaki harfler alınır.
- Eğer farklı satır ve sütunda iseler birincisinin satırının ikincisinin sütunun kesiştiği yer ile ikincinin satırı ile birincinin sütununun kesiştiği yerdeki harf alınır.

Örneğin, “ktuceng” kelimesi “LGDBAQMHECURNİFXVSOKZYWTP” anahtar matrisine göre şifrelenirse “opfqhig” metni elde edilir.

Vigenere Cipher

Fransız matematikçi Blaise de Vigenere tarafından 16. yüzyılda tasarlanmıştır. Anahtar olarak bir metin kullanılır. Düz metin boyunca anahtar tekrarlı olarak düz metinle toplanır.

```
def vigerene_cipher(plainText, key):
    i = 0
    encryptedText = ""

    key = key.lower()
    plainText = plainText.lower()
    plainText = plainText.replace('i', 'i')

    while i < len(plainText):

        char = ord(plainText[i])
        charOfKey = ord(key[i%(len(key))])

            if char >= 97 and char <= 122 and
charOfKey >= 97 and charOfKey <= 122:
            encryptedText += chr((char + charOfKey -
2*97)%26 +97)
            i=i+1
        else:
            encryptedText += chr(char)
            i=i+1
    return encryptedText
```

```
def  decrypt_vigerene_cipher(encryptedText,
key):
    i=0
    plainText = ""

    while i < len(encryptedText):

        char = ord(encryptedText[i])
        charOfKey = ord(key[i%(len(key))])

          if char >= 97 and char <= 122 and
charOfKey >= 97 and charOfKey <= 122:
            plainText += chr((char - charOfKey)
%26 + 97)
            i=i+1
        else:
            plainText += chr(char)
            i=i+1
    return plainText
```

Örneğin, “agveriguvenligisavunmasipazartesigunu” cümlesi “ktuceng” anahtar metnine göre şifrelenirse “kzpgvvmeoyppvmsluxyasklcremgbmyumtaxn” şifreli metni elde edilir.

Hill Cipher

Hill polialfabetik şifrelemesi Lester S. Hill tarafından keşfedilmiştir. Hill şifrelemesinde düz metin eşit büyüklükte bloklara ayrılır. Anahtar kare matristir. Bloklar m uzunluğunda ise anahtar mxm boyutundadır. “linuxcandirgerisiheyecandirr” metni,

[09, 07, 11, 13]
[04, 07, 05, 06]
[02, 21, 14, 09]
[03, 23, 21, 08]

4x4’lük matrisine göre şifrelenirse “jijaugqzhavcsbvgynhkfdpmotsm” şifreli metni elde edilir. *hill_cipher()* fonksiyonu parametre olarak verilen düz metni tersi alınabilir bir anahtar matrisi ile çarparak şifreli metni geri döndürür. *decrypt_hill_cipher()* fonksiyonu parametre olarak verilen şifreli metni anahtar matrisinin tersi ile çarpılarak düz metin elde edilir.

Rail Fence Cipher

Anahtarsız şifreleme tekniklerine güzel bir örnektir. Şifreli metinin ilk yarısı düz metnin çift indeksli harfleri diğer yarısı ise tek indeksli harflerinden oluşur. Deşifreleme de ise şifreli metnin ikiye bölünür. İkinci kısmın ilk harfi, birinci kısmın ilk harfinin peşine; ikinci kısmın ikinci harfi birinci kısmın ikinci harfinin peşine eklenir ve işleme böyle devam edilerek düz

metin elde edilir. Örneğin, “foreverlinux” metnini Rail fence ile şifrelenmesi ile “frvriuoeelnx” şifreli metni elde edilir.

```
f       r       v       r       i       u
   o       e       e       l       n       x
```