Diffie-Hellman Anahtar Değişimi Ugulaması

https://github.com/programmer-666/Cryptography/blob/main/Asymetric/Diffie-Helman.py

Python3

Diffie-Hellman Anahtar Değişimi Nedir?

Anahtarların taraflar arasında değiştirilmesi (elde edilmesi) için geliştirilmiş bir yöntemdir. Bu yöntem sayesinde karşılıklı iki taraf güvensiz iletişim ortamında ortak gizli anahtar elde edebilir. Elde edilen anahtar ise daha sonraki işlemlerde kullanılır.

Çalışması

Taraflarımız Alfa ve Beta olsun. Yöntemin kullanılabilmesi için iki tarafında a ve b tam sayılarını bilmelidir. Bunun dışında yine iki tarafın kendisine ait gizli bir anahtara ihtiyacı var. Anahtar değişimi $g^{ab} = g^{ba}$ matematiksel formülüne dayanır. Buna göre sırasıyla şu adımlar izlenerek anahtar dağıtılabilir:

1. g ve p biliniyorlar, g = 3, p = 17
2. Alfa: x = 15
3. Alfa: g^x mod p = 6
4. Alfa: 6 -> Beta
5. Beta: x = 13
6. Beta: g^x mod p = 12
7. Beta: 12 -> Alfa

8 numaralı adımda ise taraflar değişim denklemlerini çözerek aynı değeri elde eder.
Alfa için, **12^15 mod 17 = 10.** Beta için, **6^13 mod 17 = 10.**

Uygulama

Bu yöntemin uygulaması Python üzerinde yapılacaktır. Basit bir sınıf ve dönüştürme işlemleriyle gayet hızlı bir şekilde bitirilebilir.

Modüller

- Random

Hazırlık

Tarafları ifade eden bir sınıf yapısı altında anahtarların oluşturulması, tanımlanması ve işlenmesi yapılmaktadır. Anlatımdan farklı olarak bu uygulamada decimal değil binary üzerinden işlemler gerçekleştirilecektir. Bu işlemler ise mantıksal üs ve mod almadır. Hazırlanan uygulama 1024 bit uzunluğunda değerler ile çalışabilecek bir durumda olacaktır.

Modül

Yazdığımız kod modüler bir yapıda olacaktır. Yani ihtiyaca göre tekrar düzenlenip kullanılabilir, geliştirilebilir.

Byte'lar

Bu uygulama anlatımdaki adımların birebir aynısı olacak. Farklı olarak onluk sayı sistemindeki değerler yerine ikilik sayı sistemindeki değerler ile işlemler yapılacak. Bu da işlemlerin bit düzeyinde gerçekleştirileceği anlamına gelir. Mantıksal üs almak için XOR, mod almak için (AND - 1) ifadeleri kullanılmaktadır.

Kodlar

```
# suhaarslan.com
from random import randbytes
class Client:
    def keyControl(self, x):
        # takes first 32 bytes
        return x[:self.__byte]
    # Control Funtions
    def __init__(self, a, b):
        # a, b 32 bytes public key / a, b string
        self.__byte = 128
        self.public_key_1 = bytes(self.keyControl(a.encode()))
        self.public_key_2 = bytes(self.keyControl(b.encode()))
        self.__createPrivateKey()
    def __createPrivateKey(self):
        # private key 32 bytes
        self.__private_key = randbytes(self.__byte)
    def getPrivateKey(self):
        return self.__private_key
    # Key Functions
    def Make(self, onc = 0):
        if onc == 0:
            self.local = hex((int(self.public_key_1.hex(), 16)^int(self.__private_key.hex(),
16))&int(self.public_key_2.hex(), 16)-1)
            return self.local
        else:
            self.l_onc = hex((int(onc, 16)^int(self.__private_key.hex(), 16))&int(self.public_key_2.hex(), 16)-1)
            return self.l_onc
    # Calcs
```

Fonksiyonlar Hakkında

keyKontrol: Bu fonksiyon dışarıdan anahtarların girişinde kullanılmaktadır. Belirlenen uzunluğun dışında giriş olduğunda kısaltır.
__init__: Bilinen a ve b anahtarları alır ve gizli anahtarı oluşturur.
__CreatePrivateKey: Belirlenen boyutta rastgele byte'lar oluşturur. Gizli anahtarı yaratır.
getPrivateKey: Gizli anahtarı döndürür.

Make

Bu fonksiyon sayesinde denklem hesaplamaları yapılır. Bir adet parametre alır, bu parametre **Oncoming** yani gelen anlamındadır. Dışarıdan gelen (karşı taraftan) değeri alır. Ama bu değer yoksa karşı tarafa gönderilecek değerin hesabı yapılır.

```
k1 = """Ua){jk2#N^=yShan.]}:+#'TZL6s!F!WG8A=&-ML{gJ(B>5$xC=X/]H'[6gyNn6*B`4:UB,~)et[">$9:d#9F6nQjcp,!
pm5FPP(=VGTXe6U=Ypta&JjrRfE}"/j~g"/"""
k2 = """rt$}Lu9Gdsu:^&>8[2>waMC}g+q[=g~KJ=ymp5"`=:&M-XUDQ&SB3Yc_B-V/5b@_kt(:
[=r`98C(r2rE@wA#c_T8k+D>EMqrG5$₩_xUaDx)Tr4_J"b{vud+X<9'N<:sB"""
alpha = Client(k1, k2)
alphaCalc = alpha.Make()
beta = Client(k1, k2)
betaCalc = beta.Make()
print(alpha.Make(onc=betaCalc))
print(beta.Make(onc=alphaCalc))
```