الگوریتم رمزنگاری RSA

مقدمه

الگوریتم RSA یکی از اولین الگوریتم‌های رمزنگاری کلید عمومی است که در سال ۱۹۷۷ توسط رون ریوست، ادی شمیر و لئونارد ادلمن ابداع شد. این الگوریتم به نام آن‌ها یعنی RSA نامگذاری شده است. این الگوریتم در زمینه‌های مختلفی مانند SSL/TLS برای رمزنگاری ارتباطات امن، امضاهای دیجیتال و غیره استفاده می‌شود.

مبانی الگوریتم RSA

الگوریتم RSA بر پایه مسئله فاکتورگیری اعداد بزرگ استوار است. الگوریتم RSA از دو کلید عمومی و خصوصی استفاده می‌کند. کلید عمومی برای رمزگذاری و کلید خصوصی برای رمزگشایی استفاده می‌شود. مراحل تولید کلید در RSA به شرح زیر است:

۱- دو عدد اول بزرگ و متفاوت P و Q را انتخاب کنید.

۲ - N = P × Q را محاسبه کنید.

۳- $\phi(N) = (P-1)(Q-1)$ را محاسبه کنید.

۴- یک عدد e انتخاب کنید که $1 < e < \phi(N)$ و $\gcd(e, \phi(N)) = 1$ باشد.

5- d را طوری محاسبه کنید که $d \equiv e^{-1} \pmod{\phi(N)}$

کلید عمومی (e, N) و کلید خصوصی d است.

الگوریتم رابین میلر

این آزمون یکی از پرکاربرد ترین الگوریتم های تصادفی برای تشخیص اول بودن یک عدد صحیح است.

فرض کنیم n عددی فرد(اگه زوج بود نیازی به آزمون نداشت) و بزرگ باشد که میخواهیم اول بودن ان را تشخیص دهیم , ابتدا عدد های d و s باتوجه به فرمول زیر جوری پیدا میکنیم که دی فرد باشد

d * 2^s = n-1

سپس عدد I رو به صورت تصادفی بین 2 و n-2 انتخاب میکنیم.توی این مرحله x رو از طریق فرمول زیر بدست میاوریم

$a^d = x \pmod n$

حالا در این مرحله اگر x برابر با n-1 یا 1 نبود ان را مرکب تلقی میکنیم ولی اگه برابر بود x را حداکثر n-2 بار به توان دو در پیمانه n میکنیم و اگر x برابر با n-1 شد آزمون موفقیت امیز است .

احتمال نادست بودن محاسبات هم در هر بار اجرا ¼ است.

توضیح کد:

```python
def is_prime(n):  1 usage  Mohammadhasan Aladpoosh
    k=20
    if n <= 1:
        return False
    elif n <= 3:
        return True
    elif n % 2 == 0:
        return False

    d = n - 1
    s = 0
    while d % 2 == 0:
        d //= 2
        s += 1

    for _ in range(k):
        a = random.randint( a: 2, n - 2)
        x = pow(a, d, n)
        if x == 1 or x == n - 1:
            continue
        for __ in range(s - 1):
            x = pow(x, 2, n)
            if x == n - 1:
                break
        else:
            return False
    return True
```

این بخش همان تشخیص اول بودن یا نبودن عدد به روش میلر رابین هست که توضیح خاصی نداره فقط درباره پیدا کردن s و d من اول فرض کردم که که d برابر با n-1 و s برابر صفر هست d * 2^s = n-1 درست در بیاد سپس d را با نصف کردن به عدد فرد رسوندم.

```
def generate_large_prime(bit):  3 usages  Mohammadhasan Aladpoosh
    while True:
        n = random.getrandbits(bit)
        if is_prime(n):
            return n
```

این تابع یه عدد رندوم میسازد و سپس با استفاده از تابع قبلی اول بودن اون را تشخیص میدهد و اگر اول بود ان را ریترن میکند.

```
def generate_publickey():  1 usage  Mohammadhasan Aladpoosh
    while True:
        e=generate_large_prime(512)
        if (math.fmod(T,e)!=0 and e<T):
            return e

def generate_privatekey():  1 usage  Mohammadhasan Aladpoosh
    return pow(E,-1,T)
```

توی این قسمت کلید عمومی و خصوصی با توجه به الگوریتم rsa ساخته میشه .

```
def Encryption():  1 usage  Mohammadhasan Aladpoosh
    return pow(Message,E,N)

def Decryption():  1 usage  Mohammadhasan Aladpoosh
    return pow(Cipher,D,N)
```

این دو تابع وظیفه رمز گزاری و رمز گشایی را دارند.

```
P = generate_large_prime(512)
Q= generate_large_prime(512)
global N
N = P * Q
global T
T=(P - 1) * (Q - 1)
global E
global D
E=generate_publickey()
D=generate_privatekey()
global Message
global Cipher
print(f"N: {N}\nT: {T}\nE: {E}\nD: {D}\n")
while(True):
    print("1-Message->Cipher\n2-Cipher->Message")
    Input=int(input())
    if Input == 1:
        Message=int(input("Enter Message :"))
        print(Encryption())
    elif Input == 2:
        Cipher = int(input("Enter Cipher :"))
        print(Decryption())
    else :
        break
```

در این قسمت متغیر های تعریف میشوند و از توابع بالایی استفاده میشود و بخش بصری تحت کنسول برنامه هم نوشته شده است