RSA

Paweł Koch 145330, 16.05.2022

1.RSA

RSA (Rivest–Shamir–Adleman) to kryptosystem z kluczem publicznym, który jest szeroko stosowany do bezpiecznej transmisji danych. Jest też jednym z najstarszych. Skrót „RSA” pochodzi od nazwisk Ron Rivest, Adi Shamir i Leonard Adleman, którzy publicznie opisali algorytm w 1977 roku. Równoważny system został opracowany potajemnie w 1973 roku w GCHQ (brytyjska agencja wywiadowcza sygnałów) przez angielskiego matematyka Clifforda Cocksa . System ten został odtajniony w 1997 roku.

W kryptosystemie z kluczem publicznym klucz szyfrowania jest publiczny i różni się od klucza deszyfrującego, który jest utrzymywany w tajemnicy (prywatny). Użytkownik RSA tworzy i publikuje klucz publiczny na podstawie dwóch dużych liczb pierwszych wraz z wartością pomocniczą. Liczby pierwsze są utrzymywane w tajemnicy. Wiadomości mogą być szyfrowane przez każdego za pomocą klucza publicznego, ale mogą być dekodowane tylko przez kogoś, kto zna liczby pierwsze.

Bezpieczeństwo RSA polega na praktycznej trudności faktoryzacji iloczynu dwóch dużych liczb pierwszych, tzw. „problem faktoringowy”. Złamanie szyfrowania RSA jest znane jako problem RSA. Czy jest to tak trudne, jak problem faktoringu, pozostaje kwestią otwartą. Nie ma opublikowanych metod pokonania systemu, jeśli używany jest wystarczająco duży klucz.

RSA to stosunkowo powolny algorytm. Z tego powodu nie jest powszechnie używany do bezpośredniego szyfrowania danych użytkownika. Częściej RSA jest używany do przesyłania współdzielonych kluczy do kryptografii z kluczem symetrycznym, które są następnie wykorzystywane do masowego szyfrowania-deszyfrowania.

2.Założenia

Liczby p i q nie powinny byc zbyt duże ze względu na wolne działanie algorytmu (jednak teoretycznie im większe tym lepsza jakośc generowanych kluczy a co za tym idzie bezpieczeństwa zaszyfrowanych informacji informacji)

3.Opis metod do wyznaczenia e i d:

Do generowania liczby e użyto najprostszej implementacji wyznaczającej wszystkie liczby względnie pierwsze a następnie wybieraj ąca loswoą z wygenerowanych liczb

```
def co_prime(n):
    coprimes = []
    for i in range(n):
        if gcd(i, n) == 1:
            coprimes.append(i)
    i = randint(0, len(coprimes)-1)
    return coprimes[i]
```

Do generowania liczby d także użyto najprostszego rozwiązania iterującego po wszystkich liczbach aż nie napotka odpowiedniej:

```
def gen_d(e, fi):
    for d in range(fi):
        if (e * d - 1) % fi == 1:
            return d
```

4. Opis realizacji zadania

Do wyznaczania liczby pierwsze o okreslonej liczby cyfr użyto prostego algorytmu wyszukiwania liczb pierwszych:

```
def gen_prime(digits: int):
    lower = 10**(digits-1)
    upper = 10**digits
    primes = []
    for num in range(lower, upper):
        if num > 1:
            for i in range(2, floor(sqrt(num))):
                if (num % i) == 0:
                    break
            else:
                primes.append(num)
    i = randint(0, len(primes)-1)
    return primes[i]
```

Nastepnie w połaczeniu z algorytmami z punktu 3 utworzono generator kluczy:

```python
def rsa():
    p, q = gen_prime(4), gen_prime(4)
    n: int = p * q
    fi = (p-1)*(q-1)
    e = co_prime(fi)
    d = gen_d(e, fi)
    print(f"P: {p} Q: {q}")
    print(f"Fi: {fi}")
    print(f"Public key e: {e} n: {n}")
    print(f"Private key d: {d} n: {n}")

    return ((e, n), (d, n))
```

Zwracane są oba klucze

Nastepnie do szyfrowania użyto nastepujących metod (UWAGA bardzo czasochłonne!!)

```python
def simple_math(m: int, power: int, modulo: int):
    return m**power % modulo


def encrypt(message, key: tuple):
    encrypted = []
    for ch in message:
        encrypted.append(simple_math(ord(ch), key[0], key[1]))
    return encrypted

(function) decrypt: (message: Any, key: tuple) -> list
def decrypt(message, key: tuple):
    decrypted = []
    for ch in message:
        decrypted.append(chr(simple_math(ch, key[0], key[1])))
    return decrypted
```

Bylo to jednak rozwizanie zbyt wolne (wymagało zaszyfrowania każdego znaku wiadomości z osobna co było piekielnie czasochłonne). zastozowano zatem rozwiazanie które wyznacza sume wartosci ASCII szyfrowanej wiadomosci (szyfrowanie takie i tak jest bardzo wolne). Znaki zostaja zmaienione na wartosci ASCCI nastepnie zsumowane i zaszyfrowane, przydeszyfrowaniu jest sprawdzane tylko cz odszyfrowana liczba jest równa sumie zaszyfrowanych znaków.

```python
def simple_encrypt(message, key: tuple):
    encrypted = 0
    for ch in message:
        encrypted += ord(ch)
    return simple_math(encrypted, key[0], key[1])


def simple_decrypt(message, key: tuple):
    decrypted = simple_math(message, key[0], key[1])
    return decrypted
```

Wartości uzyskane podczas szyfrowania i deszyfrowania:

```
Public key e: 18897 n: 164869
Private key d: 56401 n: 164869
[20509, 146686, 18583, 34709, 28053, 146686, 115523, 151716, 71757, 151716, 154933, 39195, 18583, 71757, 146686, 18583, 31544, 151716, 347
 7205, 64954, 18583, 13001, 28053, 64954, 39195, 115523, 99567, 28053, 64954, 154933, 28053, 146686, 97626, 18583, 99567, 7205, 146686, 66
, 151716, 34709, 18583, 31544, 146686, 99567, 18583, 115523, 34709, 146686, 159169]
['T', 'a', ' ', 'w', 'i', 'a', 'd', 'o', 'm', 'o', 's', 'c', ' ', 'm', 'a', ' ', 'r', 'o', 'w', 'n', 'e', ' ', 'p', 'i', 'e', 'c', 'd', 'z
'i', 'e', 's', 'i', 'a', 't', ' ', 'z', 'n', 'a', 'k', 'o', 'w', ' ', 'r', 'a', 'z', ' ', 'd', 'w', 'a', '!']
```

5. Odpowiedzi na pytania

Jakie elemnty algorytmu są trudne w realizacji?
Sam algorytm nie należy do najtrudniejszych jednak korzysta z bardzo czasochłonnych obliczeń wymagajacych dużych możliwości zarówno procesora jak i pamięci.
Obliczanie liczb pierwszych jest bardzo czasochłonne, wyznaczanie liczby wzglednie pierwsze też jest czasochlonne w powyższej implementacji i dodatkowo na koniec liczby są podnoszone do ogromnych potęg co jeszcze bardziej spowalnia obliczenia.

Co stanowi o bezpieczeństwie i jakości tego algorytmu szyfrowania?
Bezpieczeństwo algorytmu opiera sia na posiadaniu 2 rodzajów kluczy które są ze sobą połaczone, w przypadku gdy ktoś zaszyfruje waidmośćdo nas naszym kluczem publiczym nikt inny nie jest w stanie tej wiadomosci odszyfrować mimo posiadania naszego klucza publicznego. tylko osoby z naszym kluczem prywatnym bed a w stanie odszyfrować wiadomość co jest głowną zaletą RSA. Sama jakośc agloyrmu jest zalezna od jak wysokich liczb pierwszych użyjemy do szyfrowania, im wieksze i z większym zakresem liczb pomiędzy nimi tym lepiej, takie liczby utworza bardzo trudny do złamania klucz który może posłużyć do szyfrowania o wysokim sopniu bezpieczeństwa.

6.Wnioski:

RSA jest bardzo bezpiecznym algorytmem, bedącym w powszechnym użyciu. Stanowi świetne zabezpieczenie, które bardzo ciężko zaatakowac. Ma jednak swoje wady, jest

bardzo powolny i korzysta z własności liczb pierwszych co sprawia, że gdyby znaleziono na nie wzor matematyczny to zaszyfrowane za pomocą RSA wiadomości mogłyby być narażone na złamanie w bardzo szybkim czasie. Obecnie jednak jest szeroko wykorzystywanym algorytmem szyfrowania na którym opiera się dużo nowożytnej kryptografii.