Alona Trubchaninova 150502

Algorytm RSA

1. Założenia

W praktyce używane są bardzo duże liczby pierwsze w algorytmie RSA, zwykle składające się z setek bitów. Wielkość tych liczb wpływa na bezpieczeństwo algorytmu, ponieważ trudność w faktoryzacji rośnie wraz z rozmiarem liczb pierwszych.

Ogólna zasada mówi, że dla bezpiecznego stosowania algorytmu RSA, długość klucza (czyli liczby bitów w liczbie n) powinna wynosić maksymalnie 2048 bitów.

2. Generowanie klucza
    a. Wybieramy dwie dwucyfrowe liczby pierwsze p i q:
    b. Obliczamy $n = p \cdot q$
    c. Obliczamy $\varphi = (p - 1)(q - 1)$:
    d. Generujemy e jako liczbę względnie pierwszą z φ
    e. Generujemy d w taki sposób, aby spełniona była zależność $e \cdot d \equiv 1 \pmod{\varphi}$

3. Przygotowanie wiadomości

Przygotowujemy wiadomość o długości 50 znaków:

- Wiadomość: "Hello! This is a sample message for RSA encryption. 1234567890 qwerty"

Wywołujemy program “key_generator.py” do generacji kluczy:

```
p, q = int(input("p: ")), int(input("q: "))         # 1)podajemy liczbę p i q
if not (is_prime(p) and is_prime(q)):
    print("p i q muszą być liczbami pierwszymi")

n = p*q                         #2) obliczmy n
phi = (p-1)*(q-1)               #3) obliczmy phi

e = 2
while math.gcd(e, phi) != 1:    #4) szukamy liczbę pierwsą dla której najwjększy wspólny dzielnik z phi wynosi 1
    e += 1                      # to i będzie kluczem publicznym

d = find_d(e, phi)              #5) wyliczamy d, to i będzie kluczem prywatnym
```

Przykładowo możemy podać jako p i q liczby 37 i 41, wtedy przy obliczeniach otrzymijemy:

- n = p*q = 1517
- phi = (p-1)(q-1) = 1440
- e = 7
- d = 823

4. Szyfrowanie wiadomości

Szyfrujemy wiadomość kluczem publicznym:

```
for letter in content:
    encryptedContent += str(ord(letter) ** key % n) + " "
```

- c ≡ m^e (mod n)
- Przykład: m = 8, c = 8^17 (mod 1517) = 1234

Moja zaszyfrowana wiadomość będzie wygłądała tak:

“1130 1137 1032 1032 962 1228 870 1194 1446 783 326 870 783 326 870 1346 870 326 1346 612 334 1032 1137 870 612 1137 326 326 1346 156 1137 870 1279 962 337 870 1435 534 1094 870 1137 1368 546 337 528 334 721 783 962 1368 1496 870 453 32 1207 35 678 1051 864 690 133 714 870 720 1287 1137 337 721 528 1453 ”

5. Odszyfrowanie wiadomości

Odszyfrowujemy wiadomość kluczem prywatnym:

```
for number in numbers:
    message += chr(int(number) ** key % n)
```

- m ≡ c^d (mod n)
- Przykład: c = 1234, m = 1234^641 (mod 1517) = 8

I na końcu

6. Porównanie wiadomości

Pierwotna wiadomość: "Hello! This is a sample message for RSA encryption. 1234567890 qwerty"

Wiadomość odszyfrowana: "Hello! This is a sample message for RSA encryption. 1234567890 qwerty"

Obie wiadomości są identyczne, co świadczy o poprawności algorytmu.

7. Uwagi dotyczące bezpieczeństwa

Algorytm RSA opiera się na trudności rozkładu dużych liczby n na czynniki pierwsze. Bezpieczeństwo algorytmu zależy od wielkości używanych liczb pierwszych. Dla praktycznych zastosowań, liczby te powinny być znacznie większe.

Trudności w realizacji algorytmu:

- Wybór bezpiecznych dużych liczb pierwszych.
- Sprawdzanie względnej pierwszości przy generowaniu e.
- Optymalizacja obliczeń modulo dużych liczb.

Bezpieczeństwo i jakość algorytmu szyfrowania:

- Bezpieczeństwo wynika z trudności faktoryzacji dużych liczb pierwszych.
- Jakość algorytmu zależy od prawidłowego wyboru kluczy i optymalizacji obliczeń.

Wnioski

Algorytm RSA jest potężnym narzędziem do szyfrowania danych, zwłaszcza w komunikatach online. Klucze powinny być starannie generowane, a liczby pierwsze używane do tego celu powinny być wystarczająco duże, aby utrudnić atakom. Praktyczne zastosowanie algorytmu wymaga także efektywnych metod szybkiego potęgowania dla dużych liczb. Bezpieczeństwo algorytmu jest ściśle związane z jakością i bezpieczeństwem używanych kluczy.