Informatyka, studia dzienne semestr III
Podstawy kryptografii 2018/2019
Prowadzący: prof. dr hab. Volodymyr Yemyets piątek, 8:30

Data oddania: _____ Ocena: _____

Bartłomiej Jencz 216783
Sylwester Dąbrowski 216743

Zadanie 2: RSA (Algorytm Rivesta-Shamira-Adlemana)

Streszczenie

Celem zadania jest przygotowanie zestawu narzędzi do generowania kluczy, szyfrowania wiadomości za pomocą klucza i odczytywania szyfrogramów za pomocą klucza w oparciu o algorytm RSA

1. Wstęp i historia.

Historycznie pierwszy, asymetryczny system kryptograficzny opublikowany w 1977 roku przez Rona Rivesta, Adi Shamira i Leona Adlemana. Od pierwszych liter nazwisk twórców tego systemu nazywa się go *kryptosystemem* RSA. Pojawienie się tego szyfru było rozwiązaniem następującego problemu:

Znaleźć system kryptograficzny, w którym klucz szyfrujący jest powszechnie znany (dostępny) i umożliwia zaszyfrowanie wiadomości przez dowolną osobę tak, aby nikt niepowołany (znający także ten klucz) nie mógł odczytać zaszyfrowanej wiadomości, a jednocześnie mógł ją odszyfrować adresat.

Rozwiązanie tego problemu jest możliwe dzięki wprowadzeniu podziału klucza na dwie części: *część publiczną (jawną)*, która służy tylko do szyfrowania i *część prywatną (tajną)* — służącą tylko do deszyfrowania. Taki klucz nazywa się *kluczem asymetrycznym*. Często część publiczną (prywatną) klucza nazywa się krótko *kluczem publicznym (prywatnym)*.

Jest to pierwszy algorytm, który może być stosowany zarówno do szyfrowania jak i do podpisów cyfrowych. Bezpieczeństwo szyfrowania opiera się na trudności faktoryzacji dużych liczb złożonych.

2. Opis algorytmu

Nasz algorytm zaimplementowaliśmy w języku C++ z wykorzystaniem biblioteki GMP. Algorytm możemy podzielić na dwie główne operacje. Generowanie kluczy oraz Szyfrowanie i deszyfrowanie.

Generowanie Kluczy:

W celu wygenerowania pary kluczy (prywatnego i publicznego) należy posłużyć się algorytmem:

- Wybieramy losowo dwie duże liczby pierwsze p i q.
- Obliczamy wartość $n = pq$
- Obliczamy wartość funkcji Eulera dla n: $\varphi(n) = (p-1)(q-1)$
- Wybieramy liczbę e $(1 < e < \varphi(n))$ względnie pierwszą z $\varphi(n)$
- Znajdujemy liczbę d, gdzie jej różnica z odwrotnością modularną liczby e jest podzielna przez $\varphi(n)$: $d \equiv e^{-1} \pmod{\varphi(n)}$ czyli kongruencje

(odwrotność e można łatwo wyznaczyć rozszerzonym algorytmem Euklidesa.)

Ta liczba może być też prościej określona wzorem: $d \cdot e \equiv 1 \pmod{\varphi(n)}$

Kluczem szyfru RSA jest wygenerowana w powyższy sposób trójka liczb *(n, e, d)*, przy czym

- para *(n, e)* jest publiczną częścią klucza.
- liczba *d* jest prywatną częścią klucza.

Liczbę *n* nazywamy *modułem* RSA, liczbę e nazywamy *wyładnikiem szyfrującym*, zaś liczbę *d* – *wykładnikiem deszyfryjącym.*

Złamanie tak ustalonego klucza wymagałoby znalezienia efektywnej metody faktoryzacji dużych liczb – póki co takowa nie istnieje.

Szyfrowanie i deszyfrowanie

Zanim zaszyfrujemy wiadomość, dzielimy ją na bloki o wartości liczbowej nie większej niż n, a następnie każdy z bloków szyfrujemy według poniższego wzoru:

$$c \equiv m^e \ (mod\ n)$$

Zaszyfrowana wiadomość będzie się składać z kolejnych bloków . Tak stworzony szyfrogram przekształcamy na tekst jawny, odszyfrowując kolejne blok według wzoru:

$$m \equiv c^d \ (mod\ n)$$

3. Przykład

Szyfrowanie

e = 7 n = 143	Otrzymaliśmy klucz publiczny (e, n). Przy jego pomocy możemy zakodować liczby od 0 do 142.
c = 7	Załóżmy, iż chcemy przesłać adresatowi zaszyfrowaną liczbę t = 123. W tym celu musimy obliczyć wartość wyrażenia: $c = 123^7 \bmod 143 = 425927596977747 \bmod 143 = 7$ Wynik jest zaszyfrowaną liczbą 123. Przesyłamy go do adresata.

Deszyfrowanie

d = 103 n = 143 c = 7	Otrzymaliśmy zakodowaną wiadomość o wartości 7. Jesteśmy w posiadaniu klucza prywatnego, który służy do rozszyfrowywania wiadomości zakodowanych kluczem publicznym.
t = 123	Wykonujemy następujące operacje: $t = 7^{103} \bmod 143$ Potęga jest zbyt duża, aby można ją było w normalny sposób obliczyć . Jednakże nas nie interesuje wartość liczbowa potęgi, a jedynie reszta z dzielenia jej przez 143. *Skorzystamy więc z szybkiego potęgowania modularnego (dyskretnego)* $7^{103} \bmod 143 = 7^{64+32+4+2+1} \bmod 143 = (7^{64} \bmod 143) \times (7^{32} \bmod 143) \times (7^4 \bmod 143) \times (7^2 \bmod 143) \times 7 \bmod 143$ $7^1 \bmod 143 = 7$ $7^2 \bmod 143 = (7^1 \bmod 143)^2 \bmod 143 = 49 \bmod 143 = 49$ $7^4 \bmod 143 = (7^2 \bmod 143)^2 \bmod 143 = 49^2 \bmod 143 = 113$ $7^8 \bmod 143 = (7^4 \bmod 143)^2 \bmod 143 = 113^2 \bmod 143 = 42$ $7^{16} \bmod 143 = (7^8 \bmod 143)^2 \bmod 143 = 42^2 \bmod 143 = 48$

	$7^{32} \bmod 143 = (7^{16} \bmod 143)^2 \bmod 143 = 48^2 \bmod 143 = 16$ $7^{64} \bmod 143 = (7^{32} \bmod 143)^2 \bmod 143 = 16^2 \bmod 143 = 113$ Do wyliczenia potęgi bierzemy tylko te reszty, które występują w sumie potęg 2: $t = 7^{103} \bmod 143 = 113 \times 16 \times 113 \times 49 \times 7 \bmod 143 = 123$

4. Dyskusja

Największą trudnością tego algorytmu jest wygenerowanie odpowiednich liczb pierwszych które muszą być długie (min. 1024 bit teraz za bezpieczne uznaje się minimum 2048) oraz znalezienie liczby e względnie pierwszej z $\varphi(n)$. Sama operacja szyfrowania i deszyfrowania nie jest skomplikowana ponieważ jest to zwykłe potęgowanie i dzielenie z resztą. Żeby móc bez problemowo działać na dużych liczbach postanowiliśmy skorzystać z biblioteki open-source GMP (*general multi-precision*). Ponadto żeby ułatwić znalezienie klucza zaczęliśmy od WYBRANIA liczby e tak żeby była ona liczbą pierwszą co zapewnia że będzie względnie pierwsza względem $\varphi(n)$. Warunek zapiszemy wzorem:

$$NWD(e, p-1) = NWD(e, q-1) = 1 \quad \text{(Na mocy małego twierdzenia Fermata)}$$

Takie rozwiązanie pozwala szybciej znaleźć klucz spełniający warunki algorytmu RSA.

Bezpieczeństwo algorytmu

Bezpieczeństwo szyfru RSA opiera się na trudności rozkładu dużej liczby n na czynniki. Nie jest znany żaden efektywny algorytm rozkładu dowolnej takiej liczby działający w czasie wielomianowym. Oczywiście, dla pewnych klas liczb naturalnych istnieją algorytmy umożliwiające ich rozkład na czynniki w krótkim czasie. Dlatego wymaga się, żeby liczby pierwsze p, q wykorzystywane do konstrukcji klucza spełniały pewne postulaty zabezpieczające przed efektywnym stosowaniem znanych algorytmów rozkładu liczb naturalnych na czynniki pierwsze.

Szyfr RSA można zaatakować również poprzez próbę wyznaczenia m z zależności

$$c = m^e \; MOD \; n,$$

ale okazuje się, że jest to zadanie równie trudne, jak rozkład na czynniki liczby n

5. Podsumowanie

Według nas powinno się jeszcze poprawić dobór rozmiaru bloku który będzie mieścił wiadomość. Aktualnie jest on ustawiony na stałe. (Są to bloki rozmiaru 2).

Wszystkie dane zapisujemy do plików. Więc osoba która będzie miała ten sam program będzie mogła z ich pomocą odszyfrować wiadomość.

Rozwiązywanie tego zadania pomogło nam rozwinąć nasze umiejętności które wymagają skorzystania z biblioteki działającej na dużych liczbach. Liczymy również na to, że napisany przez nas algorytm będzie wykorzystywany do innych zadań które wykonujemy na uczelni.

Bibliografia

[1] *Matematyka Dyskretna (FTIMS, Inf. I st. 1 sem.) Wykład 9*
[2] https://eduinf.waw.pl/inf/alg/001_search/0067.php