# Laboratorium ochrony danych

**Ćwiczenie nr 7 (dzienne 6; zaoczne 5)**

# Temat ćwiczenia: Szyfr Rivesta-Szamira-Adlemana (RSA)

*Cel dydaktyczny:* Poznanie systemu kryptograficznego z kluczem publicznym na przykładzie algorytmu Rivesta-Szamira-Adlemana.

## Wprowadzenie teoretyczne

Algorytm kryptograficzny RSA jest algorytmem wykładniczym z kluczem publicznym. Bezpieczeństwo szyfru RSA opiera się na trudności rozkładu dużej liczby na czynniki pierwsze $p$ i $q$. Tekst szyfrowany przedstawia się w postaci liczb w zakresie od 0 do $n$-1.

W celu wygenerowania kluczy kryptograficznych odbiorca kryptogramu losuje dwie liczby pierwsze $p$ i $q$ oraz liczbę $e$. Iloczyn $n = pq$ oraz $e$ stanowią publiczny klucz szyfrujący. Liczba $e$ musi być względnie pierwsza z funkcją Eulera $\phi(n)$. Spełnione są więc zależności:

$$n = pq,$$
$$\Phi(n) = (p-1)(q-1),$$
$$NWP(e, \Phi(n)) = 1.$$

Elementy szyfru $C_i$ odpowiadające wiadomości $M_i \in [0, n-1]$ otrzymujemy z zależności:

$$C_i = M_i^{\,e} \pmod n .$$

Tajny klucz deszyfrujący zawiera $n$ i liczbę $d$ spełniającą zależność:

$$de \pmod{\Phi(n)} = 1.$$

Liczba $d$ jest odwrotnością multyplikatywną liczby $e$ obliczaną mod $\phi(n)$. Odwrotność istnieje jeśli liczby $e$ i $\phi(n)$ są względnie pierwsze. Możliwe jest również wylosowanie liczby $d$ i obliczenie jej odwrotności $e$.

Wiadomość $M_i$ z kryptogramu $C_i$ możemy obliczyć z zależności:

$$M_i = C_i^{\,d} \pmod n.$$

Funkcje szyfrująca i deszyfrująca są wzajemnie odwrotne. Poprawność algorytmu można pokazać podstawiając $C_i$ do ostatniego równania i przyjmując $ed = t\Phi(n) + 1$ oraz $M_i^{\,\Phi(n)} \pmod n = 1$, gdzie $t$ jest pewną liczbą całkowitą:

$$M_i = M_i^{\,ed} \pmod n = M_i .$$

Przy wyprowadzeniu wzoru stosuje się Eulerowskie uogólnienie twierdzenia Fermata. Arytmetyka wykładników jest arytmetyką modulo $\Phi(n)$. Nie znając czynników *p* i *q* liczby *n*, wyznaczenie wartości $d$ klucza deszyfrującego jest bardzo trudne, co gwarantuje bezpieczeństwo szyfru.

Na bezpieczeństwo szyfru wpływa dobór liczb pierwszych *p* i *q*. Powinny to być liczby o długości powyżej stu cyfr. W celu lepszego zabezpieczenia przed rozkładem *n* na czynniki pierwsze zaleca się aby:

- liczby *p* i *q* różniły się o kilka cyfr,
- zarówno *p*-1 jak i *q*-1 miały duże czynniki pierwsze,
- wartość *NWP*(*p*-1,*q*-1) była niewielka.

W realizacji praktycznej algorytmu występuje problem wyboru algorytmu generowania liczb pierwszych i realizacji funkcji szyfrującej. W załączonym programie w celu wygenerowania liczb pierwszych stosuje się generator losowy liczb nieparzystych a następnie sprawdza się czy wylosowana liczba jest liczbą pierwszą, stosując twierdzenie Eulera. Liczba pierwsza spełnia wtedy zależność:

$$x^{p-1}(\text{mod } p)=1.$$

Funkcję szyfrującą:

$$C = M^{e}(\text{mod } n)$$

realizuje następujący algorytm numeryczny:

- liczbę *e* przedstawiamy w postaci binarnej $e = e_k, e_{k-1},..., e_1, e_0$,
- podstawiamy *C*=1,
- powtarzamy w pętli od *i*=*k* do 0 następujące operacje:
    1. $C = C^2(\text{mod } n)$,
    2. jeżeli $e_i = 1$ to $C = C * M(\text{mod } n)$.

Można również analizować bity od końca. Wówczas algorytm ma postać:

- liczbę *e* przedstawiamy w postaci binarnej $e = e_k, e_{k-1},..., e_1, e_0$,
- podstawiamy *C*=1,
- powtarzamy w pętli od *i*=0 do *k* następujące operacje:
    1. jeżeli $e_i = 1$ to $C = C * M(\text{mod } n)$.
    2. $M = M^2(\text{mod } n)$,

W 1994 roku szyfr RSA został złamany. Odczytano zdanie zaszyfrowane przez twórców RSA siedemnaście lat wcześniej. Pracowało nad tym 600 osób na pięciu kontynentach, przez osiem miesięcy, za pomocą sieci Internet.

## Opis oprogramowania

Przygotowanym programem ćwiczenia jest program RSA_KEY.PAS w Pascalu lub RSA_KEY.CPP w C++. Program ten generuje klucze kryptograficzne dla algorytmu RSA i zmiennych typu longint. Zawiera on następujące funkcje:

- FUNCTION POTĘGA_MOD - funkcja obliczająca wartość $x^e \pmod{n}$,
- FUNCTION LICZBA_PIERWSZA - funkcja sprawdzająca czy wylosowana liczba jest liczbą pierwsza, a jeśli nie jest wyznaczająca liczbę pierwszą większą od liczby wylosowanej.
- FUNCTION MULTI - INV - funkcja obliczająca odwrotność multiplikatywną liczby x według zależności multi_inv*x(mod n)=1.
- FUNCTION GCD - funkcja obliczająca największy wspólny podzielnik gcd(x,y) za pomocą algorytmu Euclidesa.

## Przebieg ćwiczenia

Przed ćwiczeniem należy zapoznać się z opisem algorytmu i oprogramowaniem ćwiczenia. Uruchomić i przeanalizować pracę programu RSA_KEY.PAS lub RSA_KEY.CPP. Wygenerować kilka zestawów kluczy kryptograficznych.

1. Wykorzystując funkcje z programu RSA_KEY napisać i uruchomić własny program, który będzie wykonywał następujące czynności:
   - generował klucze kryptograficzne,
   - szyfrował jeden znak w kodzie ASCII i drukował wynik szyfrowania,
   - deszyfrował utworzony kryptogram i drukował wynik deszyfrowania.

   W wyniku szyfrowania znaków kodu ASCII można otrzymać liczby złożone z kilku bajtów (w zależności od rozmiaru typu całkowitego). W niniejszym ćwiczeniu generowane wartości parametru n oraz kryptogramów C nie przekraczają 65535, czyli mieszczą się na 16 bitach. Przy drukowaniu kryptogramów w postaci znakowej należy wówczas używać jednego znaku dla jednego bajtu (tj. dwóch znaków dla wartości całkowitych C mieszczących się na 16 bitach).
2. Zastosować opracowany program do szyfrowania i deszyfrowania plików dyskowych.