Lista Nr 04, ćwiczenia

(wyk. 3 XII 09 r.)

Kryptografia RSA[1] [2]:

KROK I – tworzenie kluczy:
(**a**) Wybieramy dwie duże liczby pierwsze p i q.
(**b**) Obliczamy $n = p \cdot q$ i $\varphi(n) = (p-1) \cdot (q-1)$.
(**c**) Wybieramy niedużą liczbę nieparzystą e względnie pierwszą z $\varphi(n)$.
(**d**) Obliczamy liczbę $d < n$ taką, że

$$e \cdot d \equiv 1 \bmod \varphi(n).$$

Taka liczba d istnieje i jest jedyna !
(**e**) Publikujem parę (e, n) jako klucz publiczny.
(**f**) Trzymamy w sekrecie parę (d, n) jako klucz sekretny.

KROK II – kodowanie i dekodowanie:

(**a**) Naszą dziedziną jest $\mathbb{D} = \{0, 1, \ldots, n-1\}$.
(**b**) Kodowanie definiujemy tak: $P(M) \equiv M^e \bmod n$, dla $M \in \mathbb{D}$.
(**c**) Dekodowanie definiujemy tak: $P(M) \equiv M^d \bmod n$, dla $M \in \mathbb{D}$.

Zadania:

1. Uzasadnić wykonalność wszystkich kroków oraz wytłumaczyć dlaczego powiedzie się dekodowanie.
2. Dlaczego rozłożenie na czynniki pierwsze liczby n jest ważne dla powodzenia protokołu.
3. Przeprowadzić dyskusję poprawności implementacji przedstawionego *algorytmu RSA* w programie:

```
1  // Najwiekszy Wspolny Dzielnik
2  int NWD(int p, int q)
3  {
4      int r;
5
6      if( p<1 || q<1 )
7      {
8          puts("Ups ...");
9          exit(-1);
10     }
11     for(r=p%q; r!=0; r=p%q)
12     {
13         p = q;
14         q = r;
15     }
16     return q;
17 }
18
```

[1] Został stworzony w 1978 przez zespół: Ronald Rivest, Adi Shamir, Leonard Adleman
[2] http://en.wikipedia.org/wiki/RSA

```
19 // rozszerzony algorytm Euklidesa: p*x+q*y=NWD(p,q)
20 int NWDr(int p, int q, int *x, int *y)
21 {
22     int r,t,xp,xr,yp,yr;
23 
24     if( p<1 || q<1 )
25     {
26         puts("Ups ...");
27         exit(-1);
28     }
29     xp = *y = 1;
30     *x = yp = 0;
31     for(r=p%q; r!=0; r=p%q)
32     {
33         t  = p/q;
34         p  = q;
35         q  = r;
36         xr = *x;
37         *x = xp - t*xr;
38         xp = xr;
39         yr = *y;
40         *y = yp - t*yr;
41         yp = yr;
42     }
43     return q;
44 }
45 
46 // naiwna potega b^e mod m
47 int pmod(int b, int e, int m)
48 {
49     int c;
50 
51     for(c=1; 1<=e; e=e-1)
52     {
53         c = (c*b)%m;
54     }
55     return c;
56 }
57 
58 int main()
59 {
60      int n,y,p,q,fi_n,e,d,M0,M1,P;
61 
62 // wybieram dwie liczby pierwsze:
63      p = 61;
64      q = 53;
65      n = p*q;
66      fi_n = (p-1)*(q-1);
67 // wybieram liczbe wzglednie pierwsza z fi_n
68      e = 17;
69      if( NWD(e,fi_n)!=1 )
70      {
71          puts("Upss ...");
72          exit(-1);
73      }
74 // obliczam d spelniajace ed = 1 mod fi_n
75      NWDr(e,fi_n,&d,&y);
76      while( d<0 )
```

```
77      {
78          d = (d+fi_n) % fi_n;
79      }
80      printf("Klucz publiczny: (%i, %i)\n", e,n);
81      printf("Klucz prywatny : (%i, %i)\n\n", e,d);
82  // kodowanie liczby M
83      M0 = 123;
84      P  = pmod(M0,e,n);
85      printf("liczbe: %i zakodowano jako: %i\n\n", M0, P);
86  // dekodowanie P
87      M1 = pmod(P,d,n);
88  // sprawdzenie
89      printf("sprawdzenie: ");
90      if( M0!=M1 )
91      {
92          puts("Upss ...");
93      }
94      else
95      {
96          puts("OK !");
97      }
98      return 0;
99  }
```

Kompilacja i działanie programu:

```
$ gcc -Wall testRSA.c -o testRSA
$ ./testRSA
Klucz publiczny: (17, 3233)
Klucz prywatny : (17, 2753)

liczbe: 123 zakodowano jako: 855

sprawdzenie: OK !
$
```

Wrocław, dnia 07/12/2009