ーRSA暗号ー

課題

以下のRSAモジュールN，復号化指数DDの組み合わせで送られてきたRSA暗号文ciphertextを復号化せよ．ただし，数字列から平文への逆変換関数は後述の関数numtostringを使え．

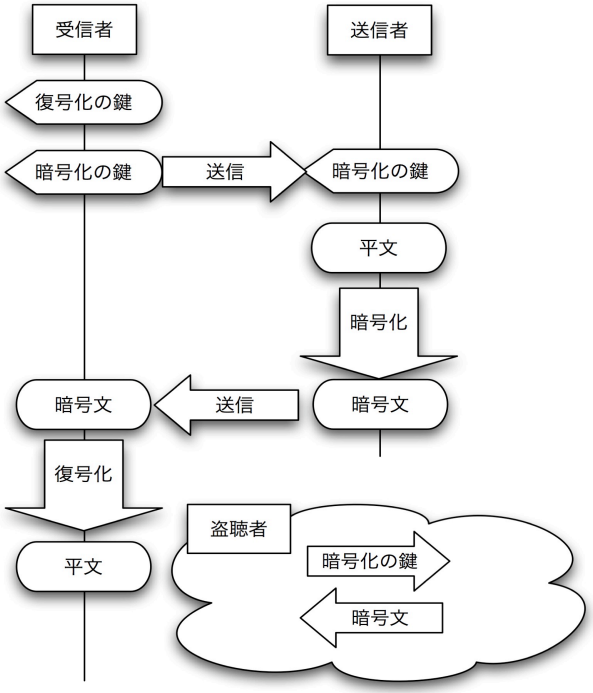
```
DD :=
971602377879271972683993828475271284 45¥
498538373913403577113306467649594367 11¥
591703407500757079507769846783079881 66¥
436532015569829063892652426693841104 87¥
8601025;

N :=
538895607981329106921046915511034649 36¥
802959626025412408880118195415973815 48¥
473287204656043493800663757534977606 45¥
711932907838570181520831781648891826 98¥
06475731;

ciphertext :=
356173341936837581214588765635817779 48¥
886867886408263016527251919216002281 40¥
682631777960221709035310114726103780 53¥
226914055689449701871105283280969329 89¥
73447073
```

課題の背景

90年代の終わり頃のゴルゴ13に，ゴルゴ13が新しい暗号の開発を支援するという逸話（373話，最終暗号)があった．暗号が通信の要であり，非常に高度なしかし簡素な数学を使っているという，さいとうたかおのいつもながらの先見性に感心した．その作り話の元にもなっていたRSA暗号についてどのような原理なのかをMapleで見ていこう．
公開鍵暗号システムはWhitfield DiffieとMartin Hellmannによって1976年にそのアイデアが公表された．そこでは，暗号化の鍵と復号化の鍵を分けることで鍵配送問題を解決している．受信者は，前もって「暗号化の鍵」を送信者に知らせておく．この「暗号化の鍵」は盗聴者に知られてもかまわない．送信者は，その「暗号化の鍵」を使って暗号化して受信者に送る．復号化できるのは「復号化の鍵」を持っている人(受信者)だけ．こうすれば「復号化の鍵」を受信者に配送する必要がなくなる．対称暗号の鍵配送問題は，公開鍵暗号を使うことで解決することになる．



この公開鍵暗号のアイデア実現に必要な一方向関数は，1978年にRon Rivest, Adi Shamir, Leonard Adlemanによって発表され，そのRSA暗号では素因数分解に非常に時間がかかることを利用している．素因数分解などのRSA暗号の基礎的な数学などに興味のある人は付記1を参照せよ．

RSA暗号は
公開鍵：数Eと数N
秘密鍵：数Dと数N
暗号化：暗号文=平文^E mod N (平文をE乗してNで割った余り)
復号化：平文=暗号文^D mod N (暗号文をD乗してNで割った余り)
である．そこで必要となるE,D,Nなどの鍵ペアは
N： 二つの素数p,qから，N=p*qで求める．

L： p-1とq-1の最小公倍数(lcm)を求める.
E： Lと互いに素なEを求める.
D： E*D mod L =1となるDを求める.
で生成する.

Mapleの関連コマンド

公開鍵，暗号鍵を生成するMapleコマンドをみよう.

N: RSA module, RSAモジュール

```
> N:=P*Q;
```

N :=

(1.3.1.1)

```
34493517372848237512608095050034301122295 73\
54441763581136550586503505578104655863510 63\
76031343593416741225347027722276355988981 06\
56259585500727695288120058393 07
```

L

```
> L:=lcm(P-1,Q-1);
```

L :=

(1.3.2.1)

```
17246758686424118756304047525017150561147 86\
77220881790568275293251752789052327925043 81\
81516508529696240082723414085519940804915 26\
89713771920936714786355150498 48
```

厳密にはL:=lcm(P-1,Q-1);だが，P*Qを法とすると，全ての数が自分自身に戻るべき乗数は，nを任意の整数としてn*(P-1とQ-1の最小公倍数)+1と表すことができる．(P-1)*(Q-1)は，必ずP-1とQ-1の最小公倍数)の倍数になるのでこちらを用いてもよい.

E:Encryption exponent, 暗号化指数

gcd(E,L)=1となるEを求めるのだが，これはたくさんある．以下のスクリプトで全てを表示することが可能.

```
> for i from 1 to L do
    if gcd(i,L)=1 then
      printf("%4d,",i);
    end if;
  od;
```

Warning, computation interrupted

実際は乱数発生器を用いたり，他の適当な素数を使ったりしているらしい.

D:Decryption exponent, 復号化指数

E*D mod L=1となるDを求める．これをMapleではLを法とするEの-1乗として計算することができる．MapleではDは微分を表す予約語なのでDDとかにしておく.

```
> DD:=eval(1/E mod L);
```

大きな素数の生成

現実的なより長い文字でできた平文(plain text)を暗号文(cipher text)にする場合にはその文字数をカバーする長さの素数が必要になる．大きな素数を乱数発生器を使って作る例.

```
> M1 := rand(10^80)();
  M2 := rand(10^80)();
  P := nextprime(M1);
  Q := nextprime(M2);
  E:= 2^16+1;  isprime(E); gcd(E,L);
```

M1 :=

```
99596883909465520335617712166467577059926 06\
82763050096752997634313400642491053 38
```

M2 :=

```
34633129088861013688643347733731974176548 31\
08742226735288661219952314767266342 12
```

P :=

```
99596883909465520335617712166467577059926 06\
82763050096752997634313400642491053 79
```

Q :=

```
34633129088861013688643347733731974176548 31\
08742226735288661219952314767266342 33
```

E := 65537

true

1 (1.3.5.1)

nextprimeはinputの数字より大きな次の素数を返してくれる関数です．Eをここでは2進数で0が並んだ数を使っています.

```
>
```

平文の数字化

次に平文を大きな数の列に換える関数．中身は分からんでもよいが，文字の扱いに興味のある人はばらしてみよ．(Mike May, S. J.の"Using RSA"より)

文字の2桁の16進数表示への変換

```
> twodigithex := a -> substring(convert(a+256,hex),2..3):
```

平文の数字列への変換関数

```
> shortconverter := proc(messagestring)
  local stringofhex, lengthofmess, hexstring, i:
    #First we convert the ASCII string to a list of decimal
  equivalents
    #Then we convert the decimal numbers to hex equivalents
   stringofhex := map(twodigithex, convert(messagestring,
  bytes)):
  print(stringofhex);
   lengthofmess := nops(stringofhex):
  print(lengthofmess);
    #Now we concatenate the hex numbers
   hexstring := cat(seq(stringofhex[i],i=1..lengthofmess)):
    #Finally we convert the big number back to decimal
   convert(hexstring,decimal,hex):
  end:
```

数字列から平文への逆変換関数

```
> numtostring := proc(bignum)
  local hexstr1, listlength, declist, i:
  #convert to a hex string
  hexstr1 := convert(bignum,hex):
  #make sure the hex string has even length
  if (length(hexstr1) mod 2) = 1 then hexstr1 := cat(`0`,
  hexstr1) fi;
  #compute the number of characters
  listlength := length(hexstr1)/2:
  #convert to a vector of decimal numbers
  declist := linalg[vector](listlength);
  for i from 1 to listlength do
  declist[i] := convert(substring(hexstr1,2*i-1..2*i),decimal,hex);
  od;
  #convert the vector to a list, then to an ASCII string
  convert(convert(convert(declist,list),bytes),name);
  end:
```

実際の操作

```
> plaintext := "Good evening Kids,  this afternoon we explore
  around RSA.";
```

```
numberedText := shortconverter(plaintext);
numtostring(numberedText);
```

解法のヒント

P:=3;Q:=11;E:=3;として暗号化関数，復号化関数を定義せよ．平文が3の場合に，暗号化した数と，復号化の結果を調べよ．

前問の鍵を使って，N(=33)以下の全ての自然数について暗号化した数を求めよ．また，復号化可能(間違いが起こらない)であることを確かめよ．

twodigithex, shortconverter, numtostringの変数，関数名などを調整して，短い単語の数字列との変換・逆変換を確かめよ．

冒頭のRSA暗号の復号化を試みよ．

付記1：RSA暗号の基礎理論

本文では省略した，RSAの原理となる素数の性質についてエッセンスだけを記しておく．

先ず基本となる中学で習った（はずの），素数の復習．自然数p>1は，二つの整数1,pのみを約数とする場合，素数(prime number)と言われる．素数でない自然数a>1は合成数(composite number)と言われる．なお，1は素数でも合成数でもない．また，素数pが整数aを割り切れれば，pはaの素因数(prime divisor)と言われる．整数a,bについてgcd(a,b)=1が成り立つとき，aとbは互いに素(coprime)であると言われる．全ての正整数は素数の積で表現可能であり，これを素因数分解(factorization)と言う．

暗号文Cが平文Mに戻る条件を導くと，暗号化と復号化の計算式は，

C = M^e (mod n)

M = C^d (mod n)

であるから，ここでmodの「大きくなる前に適宜割っていっても，最終的な余りは変わらない」という性質

{a * b} (mod n) = {(a (mod n)) * (b (mod n))} (mod n)

を使って，前者を後者に代入すると、

M = (M^e)^d (mod n)

= M^(ed) (mod n)

= M^(ed) (mod pq)

となる．この式を変形すると

M^(ed)-M = 0 (mod n)

M*(M^(ed-1)-1)=0 (mod pq)

となる．この式がなりたつ条件はFermatの小定理を使って求まり，RSA暗号が成

立する.

このあたりの詳しくて分かりやすい解説が
http://pgp.iijlab.net/crypt/rsa.html:はやわかりRSA
http://www.maitou.gr.jp/rsa/rsa10.php:さるにもわかるRSA暗号(10. RSA暗号の世界)
にある.

関連するMapleコマンド

mod, modp

nをmで割ったあまりを求める関数.

```
> n mod m;
  modp(n,m);
```

$$\mathrm{modp}(n, m)$$

$$\mathrm{modp}(n, m) \qquad (1.6.1.1)$$

ifactor

ある数の素因数分解を求める関数

```
> ifactor(91);
```

$$(7)\,(13) \qquad (1.6.2.1)$$

isprime

ある数が素数ならtrue，違えばfalseを返す関数

```
> isprime(101);
```

$$\mathrm{true} \qquad (1.6.3.1)$$

Power

modの「大きくなる前に適宜割っていっても，
最終的な余りは変わらない」という性質
{a * b} (mod n) = {(a (mod n)) * (b (mod n))} (mod n)
を使ってベキ計算してくれる関数．modと組み合わせて使う.
例えば，

```
> time(numberedText^E mod N);
```

と

```
> time(Power(numberedText, E) mod N);
```

とをくらべよ．通常のべき表示"^"のかわりに"&^"を使っても
Powerと同じ効果が得られる.

nextprime

ある数より大きな次の素数を返す関数.

```
> nextprime(100);
```

lcm(Least Common Multiple)

最小公倍数を返す関数.

```
> lcm(35,100);
```

gcd(Greatest Common Divisor)

最大公約数を返す関数.

```
> gcd(35,100);
```

gcdの結果が1，つまり1以外に共通に割り切れる数がない場合二つの数は互いに素であるという.

ithprime

i番目の素数を返す関数.

Fermatの小定理

pを素数，aを整数とすると，
modp (a^p-a , p) =0
が成り立つ．さらにaとpとが互いに素であれば，
modp(a^(p-1), p) = 1
が成り立つ.

定理

自然数nより小さい自然数aがnと互いに素であるとき，
mod(ab,n)=1となるb (0<b<n)がただ一つ存在する.

Eulerの定理(の特別な場合)

自然数n(=p*q)より小さい自然数Mがnと互いに素であるとき，
mod(M^((p-1)*(q-1)),n)=1　(0<x<n)

Euclid互除法, 拡張Euclid互除法

ネットで検索してください.