Public Key Cryptography

April 30, 2018

Prime Numbers and Factorisation

Question 1

The program is listed on page 5. The program divides n by every integer between 2 and $\sqrt{n}$. For large integers this will be significantly faster than dividing by every integer up to n, which is unnecessary since factors occur in pairs. This is the output for a known 15-digit prime number:

```
>> primality(311111111111113)
Elapsed time is 1.639951 seconds.
ans =
    "Prime"
```

This the output for the product of the prime numbers 19 328 417 and 12 211 937:

```
>> primality(236037410713729)
Elapsed time is 1.168332 seconds.
ans =
    "Not Prime"
```

We could make the algorithm faster by just dividing by the odd integers and 2. This would approximately half the runtime.

Question 2

The program is listed on page 5. When a factor is found by trial division it is saved into an array, then the same algorithm is applied to n/d beginning with the factor d. This repeats until the factor n/d is found to be prime. Here is some sample output:

```
>> factorise(2^(2^5)+1)
Elapsed time is 0.000827 seconds.
ans =
         641     6700417

>> factorise(236037410713729)
Elapsed time is 1.166735 seconds.
ans =
    12211937    19328417

>> factorise(2^1000)
Elapsed time is 0.039431 seconds.
ans =
  Columns 1 through 9
     2     2     2     2     2     2     2     2     2
     .
     .     (middle entries omitted)
     .

  Column 1000
     2
```

The worst case is when n is a prime number, or a product of two primes which are both approximately $\sqrt{n}$. In this case the complexity is $\mathcal{O}(\sqrt{n})$ since approximately $\sqrt{n}$ divisions will be required to determine the factorisation. The best case would be a number of the form 2^k. The complexity is the same as the program in Question 1.

Linear Congruences

Question 3

The program is listed on page 6. This is the program output, where d is the greatest common divisor and u and v are coefficients of Bézout's identity.

```
>> [d u v] = euclid(1996245783,192784863)
d = 3
u = -11108123
v = 115022224

>> [d u v] = euclid(2825746811,758295345)
d = 1
u = -28353319
v = 105657118

>> [d u v] = euclid(249508543104,338063357376)
d = 46656
u = 482574
v = -356165

>> [d u v] = euclid(249508543140,338063357367)
d = 9
u = 11497409916
v = -8485693393
```

So to summarise, we have:
$1\,996\,245\,783 \times (-11\,108\,123) + 192\,784\,863 \times 115\,022\,224 = 3$
$2\,825\,746\,811 \times (-28\,353\,319) + 758\,295\,345 \times 105\,657\,118 = 1$
$249\,508\,543\,104 \times 482\,574 + 338\,063\,357\,376 \times (-356\,165) = 46656$
$249\,508\,543\,140 \times 11\,497\,409\,916 + 338\,063\,357\,367 \times (-8\,485\,693\,393) = 9.$

Question 4

Solving $ax \equiv b \pmod{m}$ for x is equivalent to solving the diophantine equation $ax + my = b$ for $x, y \in \mathbb{Z}$. There exist solutions if and only if $d = \gcd(a, m) \mid b$. In this case we can divide throughout by d to obtain $(a/d)x + (m/d)y = b/d$. Since $\gcd(a/d, m/d) = 1$ we can use the Euclidean Algorithm to find s and t such that $(a/d)s + (m/d)t = 1$. Then $x_0 = sb/d$ and $y_0 = tb/d$ are particular solutions. In general the soluions are $x = x_0 + nm/d$ and $y = y_0 - na/d$ for $n \in \mathbb{Z}$, so there are exactly d solutions modulo m.

Finding Inverses and Breaking RSA

Question 5

The program is listed on page 7. The output is as follows:

```
>> cong(146295,2017,313567)
ans =
      267975

>> cong(93174,2015,267975)
'No solutions exist.'

>> cong(113314,2014,660115)
ans =
  Columns 1 through 9
     11721 24176 36631 49086 61541 73996 86451 98906 111361
        .
        .    (middle entries omitted)
        .

  Columns 46 through 53
     572196 584651 597106 609561 622016 634471 646926 659381
```

The first congruence has a unique solution modulo m because $\gcd(146295, 313567) = 1$. The second congruence has no solutions because $\gcd(93174, 267975) = 3$ which does not divide 2015. The solutions modulo m to the final congruence are $11721 + 12455n$ for $n = 0, 1, \ldots, 52$.

Question 6

Approximately $O(\sqrt{n})$ operations will be required to factorise n into p and q if p and q are both approximately $\sqrt{n}$. This would be the worst case for a given n as noted in Question 2. To determine d from $\varphi(n)$ and e we will need to use the Euclidean Algorithm. We may assume that e will be chosen to be smaller than $\varphi(n)$. The worst-case would be when e and $\varphi(n)$ are consecutive Fibonacci numbers, where at each step we descend through the Fibonacci sequence. That this is the worst case can be shown by induction on the number of steps in the algorithm. Since the nth Fibonacci number is approximatley $\phi^n/\sqrt{5}$ where ϕ is the golden ratio, $O(\log_\phi e)$ iterations are required. Using the change of base formula for the logarithm it follows that the complexity is $O(\log e)$. This analysis ignores the fact that the complexity of each step will increase as n increases.

Question 7

The program is listed on page 7. It first factorises n, then uses the program in Question 5 to calculate d. The output is as follows:

```
>> [n d]=keydecrypt(1764053131,103471)
n = 1764053131
d = 191584927
```

```
>> [n d]=keydecrypt(1805760301,39871477)
n = 1805760301
d = 1452797497

>> [n d]=keydecrypt(9976901028181,837856358917)
n = 9976901028181
d = 3864734962285

>> [n d]=keydecrypt(1723466867,692581937)
n = 1723466867
d = 225248873

>> [n d]=keydecrypt(6734071952813,2017)
n = 6734071952813
d = 4073158775953

>> [n d]=keydecrypt(1603982333,927145)
n = 1603982333
d = 1518941485
```

Decrypting RSA messages

Question 8

The program is listed on page 7. It uses the previous program to compute the private exponent from the public key, and then uses this to decrypt the message. The modulus is taken after each multiplication so that the numbers do not become too large. The algorithm could be made faster by converting the exponent to binary and repeatedly squaring, but this shall not be necessary here. Since in MATLAB we must avoid calculations with numbers exceeding 10^{15}, we need to ensure that $c < \sqrt{10^{15}}$ so that when we calculate $c \times c \pmod{m}$ it will not exceed it. Therefore to be sure that the program will run reliably we will need to ensure $n < \sqrt{10^{15}}$.

Question 9

The private key is (937513,229703). The decrypted message is:

```
092913 001415 200018 050112 122500 091400 010003 080505
190500 131515 042700 141523 002008 012029 190023 080120
000900 030112 120001 140005 070700 190114 042309 030827
```

or in English:

> "I'm not really in a cheese mood now. That's what I call an egg sandwich."

Program `primality.m` for Question 1.

```
function [output] = primality(n)
%PRIMALITYTEST Function to determine whether a natural number under 10^15
%is prime or not.
d=2;
tic
if n<=1
    output = "Not Prime";
    return
end
while d*d <= n
    if mod(n,d) == 0
        output = "Not Prime";
        toc
        return
    end
    d=d+1;
end
output="Prime";
toc
end
```

Program `factorise.m` for Question 2.

```
function [output] = factorise(n)
%FACTORISE Function to determine the prime factorisation of an integer.
d=2;
i=1;
while d*d <= n
    if mod(n,d) == 0
        f(i)=d;
        n=n/d;
        i=i+1;
    else
    d=d+1;
    end
end
f(i)=n;
output = f;
end
```

Program `euclid.m` for Question 3.

```
function [d,u,v] = euclid(a,b)
%EUCLID Function to determine the greatest common divisor of a and b, and
%integers u and v such that au + bv = gcd(a,b) by the Euclidean Algorithm.
e=0;
if a < b % swaps a and b so that a >= b
    c=a;
    a=b;
    b=c;
    e=1;
end
q=zeros(1,100);
r=zeros(1,100);
r(1)=a;
r(2)=b;
i=2;
while r(i) > 0
    i=i+1;
    q(i)=floor(r(i-2)/r(i-1));
    r(i)=mod(r(i-2),r(i-1));
end
d = r(i-1); % the greatest common divisor
l=zeros(1,100);
m=zeros(1,100);
l(1)=0;
m(1)=1;
for k=2:i-2 % reversing the algorithm using recurrence relations
    l(k)=m(k-1);
    m(k)=l(k-1)-q(i-k+1)*m(k-1);
end
u=l(i-2);
v=m(i-2);
if e==1 % swaps u and v in case a < b originally
    w=u;
    u=v;
    v=w;
end
end
```

Program `cong.m` for Question 5.

```
function [x] = cong(a,b,m)
%CONG Function to solve the linear congruence ax = b (mod m).
[d,u,v] = euclid(a,m);
if mod(b,d) ~= 0
    x='No solutions exist';
    return
end
p=a/d;
q=m/d;
[c,s,t] = euclid(p,q);
x0=mod(s*b/d,m/d);
x = zeros(1,d);
for k=1:d
    x(k)=x0+(k-1)*m/d;
end
end
```

Program `keydecrypt.m` for Question 7.

```
function [n,d]=keydecrypt(n,e)
%KEYDECRYPT Function to compute the private decryption key (n,d) from a
%given public encryption key (n,e).
f=factorise(n);
p=f(1);
q=f(2);
phi=(p-1)*(q-1);
d=cong(e,1,phi);
end
```

Program `decrypt.m` for Question 8.

```
function [m]=decrypt(c,n,e)
%DECRYPT Function to decrypt a message c given the public key (n,e).
[n,d]=keydecrypt(n,e);
m=1;
for i=1:d
    m=mod(m*c,n);
end
```