Cryptology and Coding HW7 Suyi Liu sliu92@jhu.edu

expotact.m:

function output = expofact(a,k,n)

%input positive integers a, k, and n such that a^k = 1 mod n

%output nontrivial integers d1, d2 such that d1\*d2 = n

output = [];

b = k;

s = 0;

%compute k = 2^s\*b

while mod(b,2) ~= 1

b = b/2;

s = s+1;

end

%computes mu\_i

mu\_zero = fastexp(a,n,b);

mu = [];

mu = [mu, mu\_zero];

b1 = 0;

b2 = 0;

for i = 1:s

ans = mod(mu(i)^2,n);

mu = [mu, ans];

if ans == 1

if mu(i)~= -1

b1 = extendedeuclid(mu(i)-1,n);

output = [output, b1(1)];

b2 = extendedeuclid(mu(i)+1,n);

output = [output, b2(1)];

break;

else

output = 'fails';

break;

end

end

end

Corresponding diary:

problem1.txt:

temp = expofact(2,341466300,68309797)

temp =

Cryptology and Coding HW7 Suyi Liu sliu92@jhu.edu

8527 8011

temp = expofact(5,341466300,68309797)

temp =

8011 8527

diary off

Helper methods from previous HW assignments:

extendedeuclid.m:

function output = extendedeuclid(a,b)

%we assume input a > b, if a < b, we swap them in the beginning

areal=a;

breal=b;

if a < b

temp=a;

areal=b;

breal=temp;

end;

%initializing our matrix

output=[];

A=[];

Q=[];

X=[];

Y=[];

A(1)=areal;

A(2)=breal;

Q(1)=0;

X(1)=1;

X(2)=0;

Y(1)=0;

Y(2)=1;

i=2;

%do euclid algorithm until A(i) is 0

while A(i) > 0

Q(i)=floor(A(i-1)/A(i));

A(i+1)=A(i-1)-Q(i)\*A(i);

X(i+1)=X(i-1)+Q(i)\*X(i);

Cryptology and Coding HW7 Suyi Liu sliu92@jhu.edu

Y(i+1)=Y(i-1)+Q(i)\*Y(i);

i=i+1;

end

%since in the end, i is 1 greater than the last i recorded in the matrix,

%and in matlab index 1 is actually index 0, these effects cancal out when

%we are deciding the signs of X and Y in the end

%the first column of output is gcd(a,b), second column is x, and third is y

output=[output;A(i-1)];

output=[output;(-1)^(i)\*X(i-1)];

output=[output;(-1)^(i+1)\*Y(i-1)];

end

fastexp.m:

function output = fastexp(a,n,k)

%input a, n, k, output = a^k mod n

%convert k to base 2 representation

array = [];

while k > 0

bit = mod(k,2);

quotient = floor(k/2);

array = [array, bit];

k = quotient;

end

len = length(array);

array\_mod = [];

ans = a;

%compute a^(2^i) for each i

for i = 1:len

ans = mod(ans, n);

array\_mod = [array\_mod, ans];

ans = ans^2;

end

%output = array\_mod;

%compute the final answer

output = 1;

for j = 1:len

if array(j) == 1

output = output\*array\_mod(j);

output = mod(output, n);

end

end