Public Key Cryptography 1

Diffie Hellman Key exchange

RSA Public key

Read

- Chapter 8-10, W. Stalling “Cryptography and Network security”
- Chapter 6-7, C.Paar“Understanding Cryptography”
- บทที่ 2, 5 กฤดากร, วิทยาการรหัสลับ

Prime Numbers

- prime numbers only have divisors of 1 and self
  - they cannot be written as a product of other numbers
  - note: 1 is prime, but is generally not of interest
- eg. 2,3,5,7 are prime, 4,6,8,9,10 are not
- prime numbers are central to number theory
- list of prime number less than 200 is:

```
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59
61 67 71 73 79 83 89 97 101 103 107 109 113 127
131 137 139 149 151 157 163 167 173 179 181 191
193 197 199
```

Modular Exponential : 23^{391} mod 55

exp	$23^{exp/2} * 23^{exp/2}$	$(23^{exp-1}) * (23^{exp/2})$ mod 55
1	[exception] $23^1=23$	23 mod 55 = 23
2	23*23= 529	529 mod 55 = 34
4	34*34=1156	1156 mod 55 = 1
8	1*1=1	1 mod 55 = 1
16	1*1=1	1 mod 55 = 1
32	1*1=1	1 mod 55 = 1
64	1*1=1	1 mod 55 = 1
128	1*1=1	1 mod 55 = 1
256	1*1=1	1 mod 55 = 1
512	1*1=1	1 mod 55 = 1

$$
\begin{aligned}
23^{391} &= 23^{256} * 23^{128} * 23^{4} * 23^{2} * 23^{1} \\
&= 1*1*1*34*23 \\
&= 722
\end{aligned}
$$

722 mod 55 = 12

Euler's Totient Function

- $\phi(N)$ = the numbers between 1 and *N* - 1 which are relatively prime to N
- Thus:
  - $\phi(4) = 2$ (1 and 3 are relatively prime to 4)
  - **$\phi(5) = 4$** (1, 2, 3, and 4 are relatively prime to 5)
  - $\phi(6) = 2$ (1 and 5 are relatively prime to 6)
  - **$\phi(7) = 6$** (1, 2, 3, 4, 5, and 6 are relatively prime to 7)
  - $\phi(8) = 4$ (1, 3, 5, and 7 are relatively prime to 8)
  - $\phi(9) = 6$ (1, 2, 4, 5, 7, and 8 are relatively prime to 9)

Fermat's Theorem

- $a^{p-1} = 1 \pmod{p}$
  - where p is prime and gcd(a,p)=1
- also have: $a^p = a \pmod{p}$

- a generalisation of Fermat's Theorem
- $a^{ø(n)} = 1 \pmod{n}$
  - for any a,n where gcd(a,n)=1
- **Euler's Theorem**

Euler's Totient Function, cont

- Note that $\phi(N)$ = N-1 when N is prime
- Somewhat obvious fact that $\phi(N)$ is also easy to calculate when N has exactly two different prime factors:

$$\phi(p*q) = (p-1)*(q-1)$$

- **Example**: Find ϕ (15)

$$\phi (15) = \phi (3*5) = (3-1) * (5-1) = 4*2 = 8$$

{1, 2, 4, 7, 8, 11, 13, and 14}

นิยาม ถ้ากรุป G ที่ประกอบด้วยสมาชิก a มีลำดับ $ord(a) = |G|$ แล้ว กรุปแบบนี้เรียกว่า กรุปวัฏจักร(cyclic group) และสมาชิก a ที่ให้กำเนิดกำลังสูงสุดว่าสมาชิกแบบปฐมฐาน

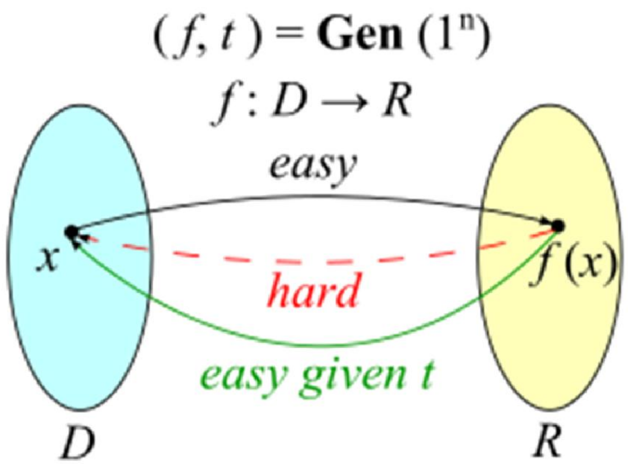
ตัวอย่างที่ 2.18 แสดงตาราง $a^i \; mod \; p$ โดย $p = 11$

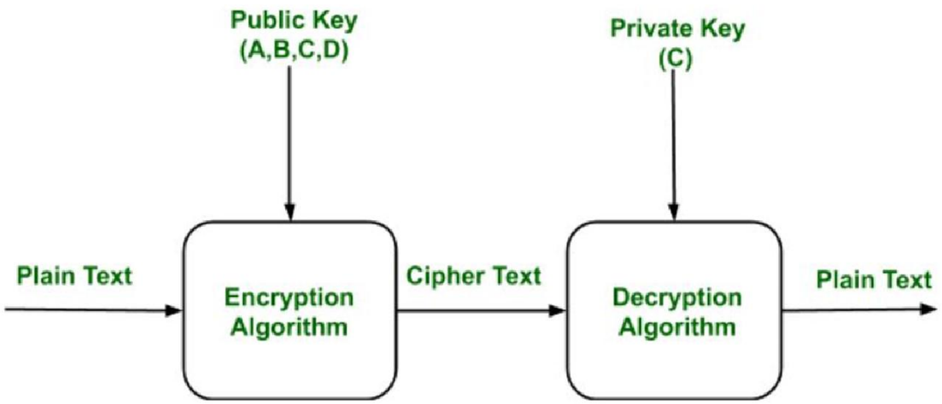
$a \backslash i$	1	2	3	4	5	6	7	8	9	10
2	2	4	8	5	10	9	7	3	6	1
3	3	9	5	4	1	3	9	5	4	1
4	4	5	9	3	1	4	5	9	3	1
5	5	3	4	9	1	5	3	4	9	1
6	6	3	7	9	10	5	8	4	2	1
7	7	5	2	3	10	4	6	9	8	1
8	8	9	6	4	10	3	2	5	7	1
9	9	4	3	7	1	9	4	3	7	1
10	10	1	10	1	10	1	10	1	10	1

Primitive Roots mod prime (19)

a	a^2	a^3	a^4	a^5	a^6	a^7	a^8	a^9	a^{10}	a^{11}	a^{12}	a^{13}	a^{14}	a^{15}	a^{16}	a^{17}	a^{18}
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	4	8	16	13	7	14	9	18	17	15	11	3	6	12	5	10	1
3	9	8	5	15	7	2	6	18	16	10	11	14	4	12	17	13	1
4	16	7	9	17	11	6	5	1	4	16	7	9	17	11	6	5	1
5	6	11	17	9	7	16	4	1	5	6	11	17	9	7	16	4	1
6	17	7	4	5	11	9	16	1	6	17	7	4	5	11	9	16	1
7	11	1	7	11	1	7	11	1	7	11	1	7	11	1	7	11	1
8	7	18	11	12	1	8	7	18	11	12	1	8	7	18	11	12	1
9	5	7	6	16	11	4	17	1	9	5	7	6	16	11	4	17	1
10	5	12	6	3	11	15	17	18	9	14	7	13	16	8	4	2	1
11	7	1	11	7	1	11	7	1	11	7	1	11	7	1	11	7	1
12	11	18	7	8	1	12	11	18	7	8	1	12	11	18	7	8	1
13	17	12	4	14	11	10	16	18	6	2	7	15	5	8	9	3	1
14	6	8	17	10	7	3	4	18	5	13	11	2	9	12	16	15	1
15	16	12	9	2	11	13	5	18	4	3	7	10	17	8	6	14	1
16	9	11	5	4	7	17	6	1	16	9	11	5	4	7	17	6	1
17	4	11	16	6	7	5	9	1	17	4	11	16	6	7	5	9	1
18	1	18	1	18	1	18	1	18	1	18	1	18	1	18	1	18	1

Trapdoor function





A **trapdoor function** is a function that is easy to compute in one direction, yet difficult to compute in the opposite direction (finding its inverse) without special information, called the "trapdoor".

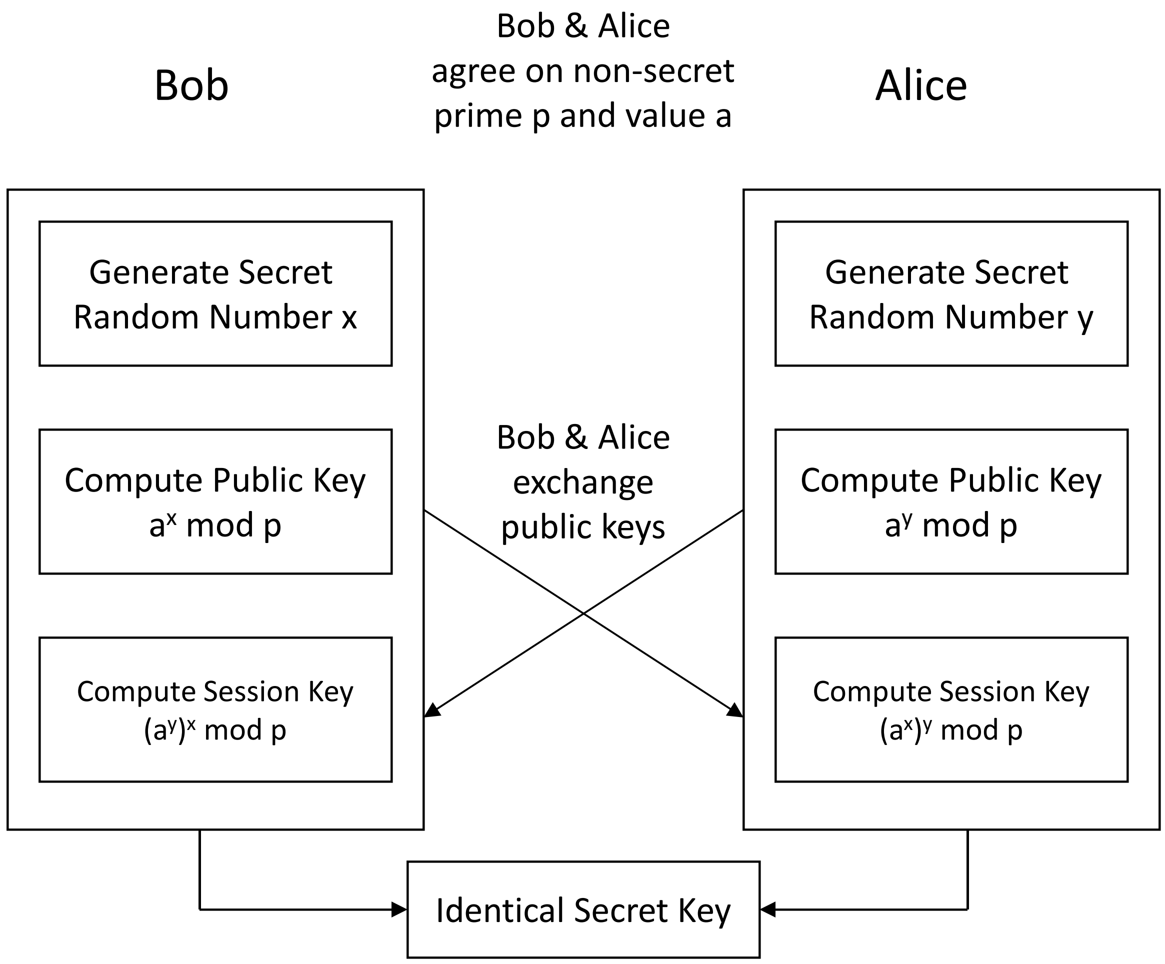
Diffie-Hellman key exchange

- A "key exchange" algorithm(1976)
  - Used to establish a shared symmetric key
- Not for encrypting or signing
- Security rests on difficulty of **discrete log** problem: given g, p and g^x mod p find x

Diffie-Hellman

- Let p be prime, let g be a **generator**
  - For any $x \in \{1,2,\ldots,p-1\}$ there is n s.t. $x = g^n \bmod p$
- **Alice** generates secret value a
- **Bob** generates secret value b
- **Alice** sends $g^a \bmod p$ to Bob
- **Bob** sends $g^b \bmod p$ to Alice
- **Both** compute shared secret $g^{ab} \bmod p$
- Shared secret can be used as symmetric key

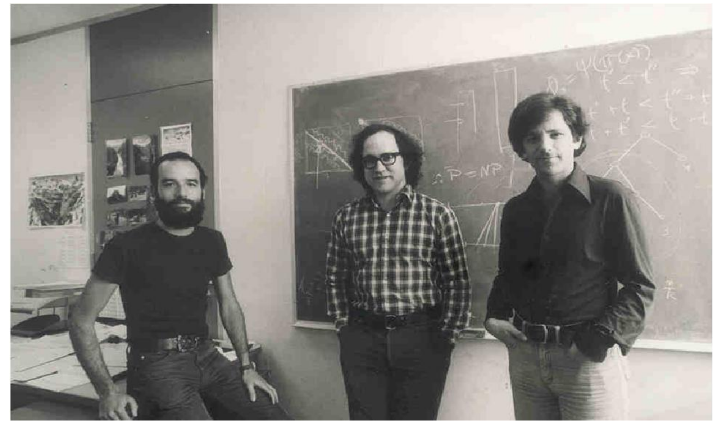
Diffie-Hellman Mathematical Analysis



RSA

- Public key cryptosystem
- Proposed in 1977 by Ron L. **R**ivest, Adi **S**hamir and Leonard **A**dleman at MIT
- Best known & widely used public-key scheme
- Based on exponentiation in a finite (Galois) field over integers modulo a prime
- Main patent expired in 2000

Shamir *Rivest* *Adleman*



Rivest *Shamir* *Adleman*

The Road to crypto

- If we can find two numbers, call them *e* and *d*, such that

$$e*d = [(p\text{-}1)(q\text{-}1)]+1$$
$$n = p*q$$

- Use *e* as the private key and *d* as the public key;

Encrypts: $c \equiv m^e \pmod{n}$

Decrypts: $m \equiv c^d \pmod{n}$

$$\begin{aligned} c^d &= (m^e \pmod{n})^d \\ &= m^{ed} \pmod{n} \\ &= m^{(p-1)(q-1)+1} \pmod{n} \\ &= m^{\phi(n)+1} \pmod{n} \\ &= m \end{aligned}$$

Recall Euler's theorem

$m^{\phi(n)+1} \bmod n = m$

A trapdoor one-way function

Public key

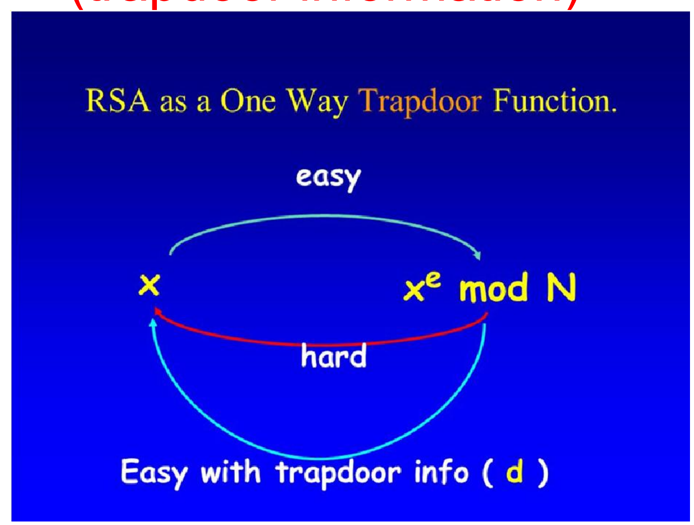
Message m

$c = f(m) = m^e \bmod n$

Ciphertext c

$m = f^{-1}(c) = c^d \bmod n$

Private key (trapdoor information)



$n = p*q$ (p & q: primes)

$e*d = 1 \bmod (p-1)(q-1)$

RSA Algorithm

- Uses two keys : *e* and *d* for encryption and decryption
- A message *m* is encrypted to the cipher text by

$$c = m^e \bmod n$$

- The ciphertext is recover by

$$m = c^d \bmod n$$

- Because of symmetric in modular arithmetic

$$m = c^d \bmod n = (m^e)^d \bmod n = (m^d)^e \bmod n$$

- **One can use one key to encrypt a message and another key to decrypt it**

RSA Key Setup

1. Selecting two **large primes** at random : p, q
   – Typically 512 to 2048 bits
2. Computing their system modulus $n=p*q$
   – note $ø(n)=(p-1)(q-1)$
3. Selecting at random the encryption key e
   - where $1<e<ø(n)$, $\gcd(e,ø(n))=1$
   - Meaning: there must be no numbers that divide neatly into e and into $(p-1)(q-1)$, except for 1.
4. Solve following equation to find decryption key d
   – $e*d=1 \bmod ø(n)$ and $0≤d≤n$
   – In other words, d is the unique number less than n that when multiplied by e gives you 1 modulo $ø(n)$
5. Publish public encryption key: $P_U=\{e,n\}$
6. Keep secret private decryption key: $P_R=\{d,n\}$

More in Euler's Theorem

- Multiply both sides of equation by *m*

$$m^{(p-1)(q-1)} \bmod n = 1$$

$$m^{(p-1)(q-1)} * m \bmod n = 1 * m$$

$$m^{(p-1)(q-1)+1} \bmod n = m$$

$$m^{\phi(n)+1} \bmod n = m$$

Euler's Totient Theorem

- One of the important keys to the RSA algorithm
  - If gcd(m, n) = 1 and m < n, then $m^{\phi(n)} \equiv 1 \pmod{n}$

relatively prime

$$m^{\phi(n)} \equiv 1 \pmod{n}$$

where $\phi(n) = (p1-1)*(q-1)$

$$m^{(p-1)(q-1)} \bmod n = 1$$

- Example:

$$m^{(p-1)(q-1)} \bmod n = 1$$

M=38

replace (p-1)(q-1) with (11-1)(5-1)

n=55

$$38^{40} \bmod 55 = 1$$

Example: Key Generation Step by Step

Step	Example
Selecting two large primes at random : p, q (use small numbers for demonstration)	p = 7; q = 19
Computing their system modulus n=p.q ø(n) = (p-1)(q-1)	n = 7*19 = 133 ø(n) = (7-1)*(19-1) = 6*18 = 108
Selecting at random the encryption key e and gcd(e, 108) = 1	e = 5
find decryption key d such that e*d = 1 mod ø(n) and 0≤d≤n	d*5 mod 108 = 1; d = 65 Check : 65*5 = 325; 325 mod 108 = 1
Public encryption key: P_U={e,n}	P_U = {5,133}
Secret private decryption key: P_R = {d,n}	P_R = {65,133}
Encryption and Decryption for m = 6	c = m^e mod n = *6^5 % 133* = 62 m = c^d mod n = 62^{65} % 133 = 6

How to deal with 1024 bits?

- n=**935180754725178127511947151434090865748897271462986652972058341716028661922905915993804021855830241749312943318773824184453712016205812164807908331802801459910407707059282312641427202496094057492449438924081178447725246251346893274769170230684627586807880439860628825319094905627224833418762790651221619242 03**
- **e**=**47609**
- d=**11964515064443823593596316031391223220980346742172807039116148962154908903300678305190741870494784604791247742558447694989408640993739843088166039297214523541519746037912861388519729724288825143561005547814973195750655549449328508806029373024427172453884284448045662068755190227462789262813325769121319683889**

we could still end up with a number with so many digits (before taking the remainder on dividing by p) that we wouldn't have enough memory to store it

Analyzing RSA

- RSA depends on being able to find large primes quickly, whereas anyone given the product of two large primes “cannot” factor the number in a reasonable time.
- If any one of p, q, m, d is known, then the other values can be calculated. So secrecy is important
- 1024 bits is considered in risk
- To protect the encryption, the minimum number of bits in n should be **2048**
- RSA is slow in pratice
  - RSA is primary used to encrypt the session key used for secret key encryption (message integrity) or the message's hash value (digital signature)

RSA-Numbers

- RSA numbers are a set of large semiprimes (numbers with exactly two prime factors) that are part of the RSA Factoring Challenge
- Officially ended in 2007 but people can still attempt to find the factorizations

RSA-100 | RSA-110 | RSA-120 | RSA-129 | RSA-130 | RSA-140 | RSA-150 | RSA-155 | RSA-160
RSA-170 | RSA-576 | RSA-180 | RSA-190 | RSA-640 | RSA-200 | RSA-210 | RSA-704 | RSA-220
RSA-230 | RSA-232 | RSA-768 | RSA-240 | RSA-250 | RSA-260 | RSA-270 | RSA-896 | RSA-280
RSA-290 | RSA-300 | RSA-309 | RSA-1024 | RSA-310 | RSA-320 | RSA-330 | RSA-340 | RSA-350
RSA-360 | RSA-370 | RSA-380 | RSA-390 | RSA-400 | RSA-410 | RSA-420 | RSA-430 | RSA-440
RSA-450 | RSA-460 | RSA-1536 | RSA-470 | RSA-480 | RSA-490 | RSA-500 | RSA-617 | RSA-2048

http://en.wikipedia.org/wiki/RSA_numbers#RSA-768

RSA-768

- RSA-768 has 768 bits (232 decimal digits), and was factored on December 12, 2009

```
RSA-768 =
1230186684530117755130494958384962720772853569595334792197322452151726400
5
          07263657518745202199786469389956474942774063845925192557326303453731548268

          50791702612214291346167042921431160222124047927473779408066535141959745985
          6902143413
```

```
RSA-768 = 33478071698956898786044169848212690817704794983713768568912431388982883793
          878002287614711652531743087737814467999489

×36746043666799590428244633799627952632279158164343087642676032283815739666
          511279233373417143396810270092798736308917
```

RSA security summary

There are two one-way functions involved in the security of RSA.

One-way function	Description
Encryption function	The encryption function is a trapdoor one-way function, whose trapdoor is the private key. The difficulty of reversing this function without the trapdoor knowledge is **believed** (but not known) to be as difficult as factoring.
Multiplication of two primes	The difficulty of determining an RSA private key from an RSA public key is **known** to be equivalent to factoring n. An attacker thus cannot use knowledge of an RSA public key to determine an RSA private key unless they can factor n. Because multiplication of two primes is believed to be a one-way function, determining an RSA private key from an RSA public key is believed to be very difficult.

Finding GCD

Using the Theorem: Given integers $a>0$, b, q, r, such that $b = aq + r$, then $gcd(b, a) = gcd(a, r)$.

Euclidian Algorithm

Find gcd (b, a)

while $a \neq 0$ *do*

$r \leftarrow b \bmod a$

$b \leftarrow a$

$a \leftarrow r$

return b

Multiplicative Inverse

Definition: Given integers $n>0$, a, b, we say that b is a **multiplicative inverse of** a **modulo** n if

$$ab \equiv 1 \pmod{n}.$$

Proposition: Given integers $n>0$ and a, then a has a multiplicative inverse modulo n if and only if a and n are relatively prime.

Towards Extended Euclidian Algorithm

- **Theorem:** Given integers a, b > 0, then d = gcd(a,b) is the least positive integer that can be represented as ax + by, x, y integer numbers.
- How to find such x and y?

- See p-157 understanding crypto

The Extended Euclidian Algorithm

First computes	Then computes	And
$b = q_1 a + r_1$	$x_0 = 0$	$y_0 = 1$
$a = q_2 r_1 + r_2$	$x_1 = 1$	$y_1 = 0$
$r_1 = q_3 r_2 + r_3$	$x_2 = -q_1 x_1 + x_0$	$y_2 = -q_1 y_1 + y_0$
…	…	…
$r_{k-3} = q_{k-1} r_{k-2} + r_{k-1}$	$x_k = -q_{k-1} x_{k-1} + x_{k-2}$	$y_k = -q_{k-1} y_{k-1} + y_{k-2}$
$r_{k-2} = q_k r_{k-1}$		

We have $ax_k + by_k = r_{k-1} = \gcd(a,b)$

Extended Euclidian Algorithm

```
Extended_Euclidian (a,b)
    x=1;  y=0;  d=a;  r=0;  s=1;  t=b;
    while (t>0) {
                    q = ⌊d/t⌋;
        u=x-qr;  v=y-qs;  w=d-qt;
        x=r;     y=s;     d=t;
        r=u;     s=v;     t=w;
    }
    return (d, x, y)
end
```

Invariants:

$ax + by = d$

$ar + bs = t$

Lab1 RSA-129 digits

N=114381625757888867669235779976146612010218296721242362562561842935706935245733897830597123563958705058989075147599290026879543541

e=9007

##THE MAGIC WORDS ARE SQUEAMISH OSSIFRAGE

m=200805001301070903002315180419000118050019172105011309190800151919090618010705

d=106698614368578024442868771328920154780709906633937862801226224496631063125911774470873340168597462306553968544513277109053606095

c=pow(m,9007,N)

print('c=',c)

c=96869613754622061477140922254355882905759991124574319874695120930816298225145708356931476622883989628013391990551829945157815154

m=pow(c,d,N)

print('m=',m)

Lab2 RSA-1024

RSA-1024 has 1,024 bits (309 decimal digits), has not been factored so far

```
RSA-1024 =135066410865995223349603216278805969938881475605667027524485143851526510604859533833940287150571909441798207282164471551373680419703964191743046496589274256239341020864383202110372958725762358509643110564073501508187510676594629205563685529475213500852879416377328533906109750544334999811150056977236890927563
```

```python
from Crypto.PublicKey  import RSA
def generate_RSA(bits=1024):
    new_key = RSA.generate(bits, e=65537)
    public_key = new_key.publickey().exportKey("PEM")
    secret_key = new_key.exportKey("PEM")
    return secret_key , public_key

x=generate_RSA()
print(x)
```

Lab3 RSA small-key(10 digits)

N = 1602475129

p=19801,q=80929

e = 64037 (public key)

ถ้า cipher = 1187226754

แล้วจงหาค่า Message

```
def eea(a, b):
    s, old_s = 0, 1
    t, old_t = 1, 0
    r, old_r = b, a
    while r != 0:
        quotient = old_r // r
        old_r, r = r, old_r - quotient * r
        old_s, s = s, old_s - quotient * s
        old_t, t = t, old_t - quotient * t
    return old_r, old_s, old_t
def inv(n, p):
    gcd, x, y = eea(n, p)
    assert (n * x + p * y) % p == gcd
    if gcd != 1:
        # Either n is 0, or p is not a prime number.
        raise ValueError(
            '{} has no multiplicative inverse '
            'modulo {}'.format(n, p))
    else:
        return x % p
```

Lab 4 Trail division

N = 90668363

e = 9007 (public key)

ถ้า cipher =16765951

แล้วจงหาค่า Message

$i = 2$

while $i < \lfloor\sqrt{N}\rfloor$

if $i|N$ **then** output i

$i = i + 1$

output N is prime

1.เป็นไปได้หรือไม่ที่ใช้ Trial division หาค่า p,q จาก

N=11438162575788886766923577997614661201021829672124236256256

1842935706935245733897830597123563958705058989075147599290026

879543541 เพราะ ? พิสูจน์

2.จากข้อ(1) program วนกี่รอบ

Supplement

- Compute ϕ(N) in C code:

```
phi = 1;
for (i = 2 ; i < N ; ++i)
    if (gcd(i, N) == 1)
        ++phi;
```

1.หาค่า ϕ(94946491)

2.จากโปรแกรม เป็นไปได้หรือไม่ที่หาค่า ϕ(N) ขนาด N=50 digits เพราะ ? พิสูจน์

Lab5 pow(c,d,N)

Expmod(*c,d,N*)

m=1

while $d \geq 1$

if $d mod 2 = 1$ **then**

$m = (m \times c) mod N$

$c = c^2 mod N$

$d = \lfloor d/2 \rfloor$

return(*m*)

1.จาก pseudo code เขียน program

2.จากข้อ(1) ถ้าโปรแกรมยกกำลัง ขนาด 129 digits แล้วการทำงานวนกี่รอบ แสดงโดยโปรแกรม

Note

from math import ceil

from math import floor

from math import sqrt

Lab 6 primitive root mod p

จาก Lab 4 Trail division

หา primitive root ที่น้อยที่สุดของ

1. 17
2. 137
3. 9001

ค่า 6 เป็น primitive root ของ 17 ?

ค่า 11 เป็น primitive root ของ 137 ?

ค่า 1551 เป็น primitive root ของ 9001 ?

Note

Array operate

```
>>> A=[]
>>> A.append(1)
>>> A.append(2)
>>> A.append(3)
>>> A[1]
```

Lab7 discrete log Force

จาก $X = g^{x} \bmod P$ หาค่า x ถ้า

1. $X = 7, \ g = 5, P = 23$
2. $X = 43 \ \ g = 587, P = 9001$