암호함수 계산 및 연습 (Sage)

이 은 정
(Ewha Institute of Mathematical Sciences)

NIMS summer school, 2012/7/5

Goal of this talk

- RSA 암호
- 타원곡선 암호
- 페어링 기반 암호

Sage 로 만들어보기

(I want to share how to program
when I use a new package at first.)

Sage ?

- SAGE 가 무엇인가?
  - Software for Arithemetic Geometry Experimentation
- Mission of SAGE team(www.sagemath.org)
  - 쓸만한 공짜의 공개의
  - Magma, Maple, Mathematica, Matlab 대체할수 있도록
- 왜 SAGE인가?
  - 공짜, 브라우저인터페이스, 정수론 관련 함수 많음
  - 암호 알고리듬들을 배울 때, 기초연구단계에 유용

SAGE 만드는 사람들

+ Contributors like you

B. Moretti, R. Miller, W. Stein, P. Clark, T. Boothby, J. Kantor, Y. Qiang, R. Bradshaw

SAGE programming*

Arithmetic

```
sage: a + b
11

sage: a*b, a^b
(28, 2401)
sage: a/b, a % b
(7/4, 3)
sage: a//b, a % b
(1, 3)

sage: float(a)/b
1.75
sage: int(_)
1
```

Types

```
sage: parent(1)
Integer Ring
sage: parent(7/2)
Rational Field
sage: parent(3.1416)
Real Field with 53 bits of precision

sage: sqrt(-1)
I
sage: parent(_)
Symbolic Ring
sage: a, b = 1 + I, 1 - I
sage: a*b
(1 - I)*(I + 1)
sage: expand(_)
2
```

* CIMAT Lectures, Feb. 26, 2008
Leture 3a: Programming in Sage and Python, by James Carlson

SAGE programming*

Lists

```
sage: a = [1, 7, 2]; b = [4, 5]
sage: c = a + b; c
[1, 7, 2, 4, 5]
sage: c.sort(); c
[1, 2, 4, 5, 7]

sage: c.<tab>

c.append   c.extend   c.insert   c.remove   c.sor
c.count    c.index    c.pop      c.reverse

sage: c.append("foo"); c
[1, 2, 4, 5, 7, 'foo']
```

Manipulating lists

```
sage: c; c[0]
['foo', 7, 5, 4, 2, 1]; 'foo'

sage: c[0] = 11
sage: c
[11, 7, 5, 4, 2, 1]

sage: c[0:2]
[11, 7]

sage: c[2:]
[5, 4, 2, 1]

sage: c[:2]
[11, 7]
```

SAGE programming*

Constructing lists

```
sage: [ n^2 for n in range(2,10) ]
[4, 9, 16, 25, 36, 49, 64, 81]

sage: QR = [ n^2 % 11 for n in range(1,11)
sage: QR
[1, 4, 9, 5, 3, 3, 5, 9, 4, 1]
sage: 8 in QR
False
```

Functions

```
sage: def f(x):
....:     if x % 2 == 0:
....:         return x/2
....:     else:
....:         return rand(10)*x + rand(3)

sage: def run(f,a,N):
....:     while a > 1 and a < N:
....:         print a,
....:         a = f(a)

sage: run(f,10,1000)
10 5 25 27 54 27 190 95 191
sage: run(f,10,1000)
10 5 31 125 876 438 219
```

SAGE programming*

Loops

In the preceding examples we used loops to execute repetitive action.

There are two kinds of loops in Python / Sage.

The for loop:

```
for x in GF(7):
  print x, x^2
```

The while loop:

```
N = 12928
while N % 2 == 0:
  N = N/2
  print N,
```

Conditionals

We also used conditionals to make decisions:

```
if n % d == 0:
  n = n / d
  print d,
else:
  d = d + 1
```

RSA

RSA

Keygen:

p, q : primes

n = p*q

e = random integer such that gcd(phi(n), e)=1

d = e^-1 mod phi(n)

Encrypt:

c = m^e mod n

Decrypt:

m = c^d mod n

To implement, what we need?

prime gen

gcd

inverse mod

exponentiation

equality

(ascii to integer)

after implementation, what should we do?

at least, check the efficiency...

RSA

```
p=random_prime(200)
q=random_prime(200,lbound=100)
p;q;
```

```
def prime_gen(bn):
    p = next_prime(2**bn)
    q = next_prime(p)
    return p,q
```

```
p,q = prime_gen(600)
n = p*q; n
```

```
e = random(); e
```

```
while 1:
    e = ZZ.random_element(2^100)
    if gcd(e,m) == 1:
        break
```

```
xgcdout = xgcd(e,m)
d = Integer(mod(xgcdout[1], m))
mod(e*d,m)
```

```
M = ZZ.random_element(n)
type(M);print(gcd(M,n))
```

```
C = mod(M**e, n)
```

```
time C = power_mod(M, e, n)
```

```
time Mt = C.powermod(d,n)
print(mod(M-Mt,n))
Integer(M) == Integer(Mt)
```

Powering algorithm

```
def rspow(g,x,n):
    xbs = x.bits()
    h = 1
    for i in range(len(xbs)-1,-1,-1):
       h = mod(h*h, n)
     #  print(i,h)
       if xbs[i] == 1:
           h = mod(h*g,n)
     #      print(h)
    return h
```

```
time Ct = rspow(M,e,n)
```

ECC

Diffie Hellman Key Exchange using Elliptic curve

E / Fq : elliptic curve over Fq

P a point in E(Fq) with large prime order, say r

Key of Alice = aP for an random integer < r

What do we need to know?

Finite field

Elliptic curve

Point

order of the point

Point addtions

aP "scalar multiplication"

Finite field

```
q = random_prime(2^100)
Fq = FiniteField(q)
F.<b> = GF(2^5)
Fqk = GF(q^3, 'w')
type(Fq);type(F);Fq.polynomial();F.polynomial()
```

```
E = EllipticCurve(Fq,[1,1])
```

Point and its order

```
P = E.random_point(); P; factor(P.order()); E(0);
```

Some commands on Elliptic curve

```
print  E.discriminant()
print  E.j_invariant()
N = E.cardinality()
print N
```

Point addition

$P1 = (x1 : y1), P2=(x2 : y2), P1+P2 = (x3 : y3)$

$\lambda = (y1-y2) / (x1 - x2)$ or $(3x1^2+A) /2y1$

$x3 = \lambda^2 - x1-x2$

$y3 = -(\lambda(x3-x1) + y1)$

```
def Add(P1,P2,A):
    x1 = P1[0]
    y1 = P1[1]
    x2 = P2[0]
    y2 = P2[1]
    if P1 == E(0):
        return P2
    if P2 == E(0):
        return P1
    if P1 == -P2:
        return E(0)
    if P1 == P2:
        ld = (3*x1*x1+A)/(2*y1)
    else:
        ld = (y1-y2)/(x1-x2)
    x3 = ld*ld - x1 - x2
    y3 = -(ld*(x3-x1)+y1)
    return E((x3,y3))
```

Scalar multiplication

-- Use repeated doubling algorithm which is an analogue of repeated squaring algorithm

```
def ScalMul(P,a,A):
    abts = a.bits()
    T = E(0)
    for i in range(len(abts)-1,-1,-1):
       T = Add(T,T,A)
       if abts[i] == 1:
           T = Add(T,P,A)
    return T
```

```
ScalMul(P,ZZ.random_element(10000),1)
```

Q: How many number of additions in ScalMul ??

Pairing 기반 암호

- 예1
  - $y^2 = x^3+7x$ over F13
  - P in E(F13)
  - What is f such that (f) = 6(P)-(O)?

Pairing 기반 암호(예2)

- k = 12
- q = 82434016654300679721217353503190038836571781811386228921167322412819029493183
- A = 0
- B = 3
- N = 82434016654300679721217353503190038836284668564296686430114510052556401373769
- t = q+1-N

We will work on the ate pairing (P, Q) for P in G_1, Q in G_2

Pairing 기반 암호(예 3)

- q =11734412966758325511610870894960132055009313621180553826424735854427431416927843151600948313176295757;
- A = 1
- B = 0
- t = 218777514637794667701275816233170393418180314 18;
- N = q - t +1
- r = 157979637185282574909590358449806730811902876 89
- k=?

We will work on the ate pairing (P, Q) for P in G_1, Q in G_2