Κρυπτογραφία

Ομάδα Ασκήσεων 2

Ανακοίνωση Παρασκευή 30/12/2016 - Προθεσμία Δευτέρα 30/01/2017

Γεώργιος Στεφανίδης, Σοφία Πετρίδου

Άσκηση 1

Κρυπτοσύστημα RSA

1. Υλοποιείστε τη συνάρτηση **keygen(bits)** για τη δημιουργία των κλειδιών RSA η οποία θα δέχεται ως είσοδο το πλήθος των bit των κλειδιών (με τιμές στο διάστημα [512,1024]) και θα επιστρέφει τα κλειδιά: δημόσιο και ιδιωτικό.
2. Υλοποιείστε τη συνάρτηση **rsa_enc(m,e,n)** για τη λειτουργία κρυπτογράφησης RSA η οποία θα δέχεται ως είσοδο το μήνυμα m και το δημόσιο κλειδί (e, n) και θα επιστρέφει το κρυπτοκείμενο.
3. Υλοποιείστε τη συνάρτηση **rsa_dec(c,d,n)** για τη λειτουργία αποκρυπτογράφησης RSA η οποία θα δέχεται ως είσοδο το κρυπτοκείμενο c, το ιδιωτικό κλειδί d και το modulus n και θα επιστρέφει το απλό κείμενο.
4. Υλοποιείστε τη συνάρτηση **str2num(s):** για τη λειτουργία κωδικοποίησης βάσει του εκτεταμένου ASCII η οποία θα δέχεται ως είσοδο μια συμβολοσειρά s και θα επιστρέφει έναν μεγάλο ακέραιο.
5. Υλοποιείστε τη συνάρτηση **num2str(n):** για τη λειτουργία αποκωδικοποίησης βάσει του εκτεταμένου ASCII η οποία θα δέχεται ως είσοδο έναν μεγάλο ακέραιο n και θα επιστρέφει τη συμβολοσειρά στην οποία αντιστοιχεί.
6. Μέσα από κατάλληλες κλήσεις και έχοντας ορίσει ένα απλό κείμενο της επιλογής σας δείξτε τις λειτουργίες κρυπτογράφησης και αποκρυπτογράφησης.

Απάντηση

In [3]:

```
def keygen(n):
    a=next_prime(ZZ.random_element(2^(n//2 +1)))
    b=next_prime(ZZ.random_element(2^(n//2 +1)))
    x=a*b
    phi_x = (a-1) * (b-1)
    while True:
    #random_element: if two integers are given, return an integer between x and y-1 inc
lusive 1,phi_x
        e = ZZ.random_element(512,1024)
        if gcd(e,phi_x) == 1:
            break

    d=inverse_mod(e,phi_x)
    return x,e,d, a , b


keygen(20)

def rsa_enc(m,e,n):
    return power_mod(m,e,n)

def rsa_dec(c,d,n):
    return power_mod(c,d,n)

def str2nums(s):
    return ZZ(map(ord,s),128)

def num2str(n):
    dgs=n.digits(128)
    return ''.join(map(chr,dgs))
```

In [16]:

```
n,e,pri ,p,q = keygen(1024)
message = 'We are all proletariat'
plaintext=str2nums(message)
ciphertext = rsa_enc(plaintext,e,n)
print ciphertext
print "encryption", num2str(ciphertext)
decrtext = rsa_dec(ciphertext,pri,n)
print "decryption", num2str(decrtext)
```

```
28580402600624996242754049744018842601902241457330216145423645307421494327
33352374970814908822581604137369476447784025059377774460158792856444500581
70688948466299302835604779427830772215185722000032720905469286585011220830
42904838337052833443414813248349077282923941078712012883627717960886660866
24926161851
encryption ;7s□□L<3□U?□.□RcAH4,S□V□.k <uf□%ts□"(□Pr;&/%`&vp□8=eK    zS
9@□;□□rYV□□9□□Fg5/Gr□%4v~□X^D@c      □>4g□BD3K□BH□□h       3□?□□)?2J
0!□bB$□□F[nA□□vwu`XgB"□,^bl{□u□□
decryption We are all proletariat
```

Άσκηση 2

Το Κινέζικο Θεώρημα Υπολοίπων

1. Επιταχύνετε τη διαδικασία αποκρυπτογράφησης του RSA με χρήση του Κινέζικου Θεωρήματος υπολοίπων (CRT). Υλοποιείστε τη συνάρτηση **rsa_decrt(c,d,p,q)** για τη λειτουργία αποκρυπτογράφησης RSA η οποία θα δέχεται ως είσοδο το κρυπτοκείμενο c, το ιδιωτικό κλειδί d και τους πρώτους p, q και θα επιστρέφει το απλό κείμενο.
2. Μέσα από κατάλληλες κλήσεις των συναρτήσεων της Άσκησης 1 (με εξαίρεση την **rsa_dec**) και της συνάρτησης **rsa_decrt** και έχοντας ορίσει ένα απλό κείμενο της επιλογής σας δείξτε τις λειτουργίες κρυπτογράφησης και αποκρυπτογράφησης.

Απάντηση

In [4]:

```
n,e,pri ,p,q = keygen(1024)
message = 'paok'
plaintext=str2nums(message)
ciphertext = rsa_enc(plaintext,e,n)

print "encryption", num2str(ciphertext)

def rsa_decrt(c,d,n,p,q):
    n=p*q
    phi_n = (p-1) * (q-1)
    cp=c%p
    cq=c%q
    dp=d%(p-1)
    dq=d%(q-1)
    mp=power_mod(cp,dp,p)
    mq=power_mod(cq,dq,q)
    tp=inverse_mod(q,p)
    tq=inverse_mod(p,q)
    m=(q*tp*mp+p*tq*mq)%n
    return m

print "decryption", num2str(rsa_decrt(ciphertext,pri,n,p,q))
```

```
encryption �{�Chv̈�P�S□N�[����Q��        4�Jf□O��□��!□�&��□8
i□□
□□,□□□w+□/□□□□T□□5Xڭ�Rj□/�)�□Y�����N4��ىaM4�U"r�z□J��sCY�
y.�Dw�l�%�2��□��EKk��
decryption paok
```

Άσκηση 3

Επίθεση κοινού modulus

Υλοποιείστε τη συνάρτηση **crack_rsa_comoda(p,q,e1,e2,c1,c2)** για την επίδειξη μιας επίθεσης κοινού modulus. Δημιουργείστε ένα δικό σας ρεαλιστικό σενάριο στο οποίο η συνάρτηση θα δέχεται ως είσοδο τους πρώτους p, q, τους δημόσιους εκθέτες e_1, e_2 και τα κρυπτοκείμενα c_1, c_2 και θα επιστρέφει το απλό κείμενο m.

Απάντηση

In [5]:

```
#εβαλα για παραδειγμα το παραδειγμα που εχετε κανει εσεις στο Lab6 για ν επαληθευσω γρη
γορα το αποτελεσμα

def crack_rsa_comoda(p,q,e1,e2,c1,c2):
    #cn1=str2nums(c1) αναλογα αν στειλουμε αριθμους ή κειμενο σε ASCII
    #cn2=str2nums(c2)
    x,s,t=xgcd(e1,e2)

    mn= (power_mod(c1,s,p*q)*power_mod(c2,t,p*q))%(p*q)

    #m=num2str(mn) αν ειναι κειμενο πρεπει ν μετατρεψω τους αριθμους σε κειμενο
    return mn


print crack_rsa_comoda(37,43,17,5,849,22)
```

```
500
```

Άσκηση 4

Ανταλλαγή κλειδιών Diffie-Hellman

1. Υλοποιείστε τη συνάρτηση **generate_parameters(bits)** η οποία δέχεται ως είσοδο το πλήθος των bits ενός μεγάλου πρώτου p και επιστρέφει 4 τιμές: p, q, g και F. Θα πρέπει τα p και q να είναι πρώτοι αριθμοί τέτοιοι ώστε $p = 2 * q + 1$, g θα είναι ένας ακέραιος γεννήτορας του $\mathbb{Z}_{\mathrm{p}}{}^*$ και F ένα πεπερασμένο σώμα με p στοιχεία.
2. Υλοποιείστε τη συνάρτηση **public_private_pair(p,q,g,F)** η οποία παίρνει ως είσοδο την έξοδο της **generate_parameters** και επιστρέφει το ζευγάρι τιμών (X, x), όπου $X = g^x \mod p$ και $x \in \{2, \ldots, p-2\}$.
3. Υλοποιείστε τη συνάρτηση **generate_secret(X,y)** η οποία παίρνει ως είσοδο τη δημόσια πληροφορία του άλλου μέλους της επικοινωνίας κατά την ανταλλαγή DH και τον ιδιωτικό εκθέτη και επιστρέφει το κοινό μυστικό κλειδί.
4. Χρησιμοποιώντας τις παραπάνω συναρτήσεις δείξτε την ανταλλαγή κλειδιών DH.

Απάντηση

In [11]:

```
#ερωτημα 1,2,3,4
def generate_parameters(n):
    p=next_prime(ZZ.random_element(2^(n//2 +1)))
    F=GF(p)
    q=(p-1)/2
    i=1
    g=mod(primitive_root(p),p)
    Zp = IntegerModRing(p)
    #while true:
     #   g=F(i)
      #  if multiplicative_order(Zp(g))==(p-1) :
       #     break
        #i+=1

    return p,q,g,F



def public_private_pair(p,q,g,F):
    x=F(randint(2,p-1))
    X=F(g^x)
    return X,x

def generate_secret(X,y):
    return X^y

P,Q,G,f=generate_parameters(128)
print 'p=',P,'q=',Q,'g=',G,'F=',f
X,x=public_private_pair(P,Q,G,f)
print 'Alice private and public',x,X
Y,y=public_private_pair(P,Q,G,f)
print 'Bob private and public',y,Y
print 'Alice common', generate_secret(X,y)
print 'Bob common', generate_secret(Y,x)
```

```
p= 145074220581674074472870973813964321333 q= 72537110290837037236435486906982160666 g= 2 F= Finite Field of size 14507422058167407447287097381396432 1333
Alice private and public 55766843132163315408823307883198983034 5036857842373383884791277992904291 0839
Bob private and public 117013009097234648107835461144432946899 89541862445996060862946016579325818096
Alice common 24598733449484254821980851557397053075
Bob common 24598733449484254821980851557397053075
```

Άσκηση 5

Πρωτόκολλο κρυπτογράφησης El Gamal

Κωδικοποιείστε κατάλληλα το μήνυμα "Next Monday" και εν συνεχεία κρυπτογραφήστε το με το κρυπτοσύστημα El Gamal. Εν συνεχεία προβείτε σε αποκρυπτογράφηση και αποδωδικοποίηση προκειμένου να ανακτήσετε το αρχικό μήνυμα. Επιλέξτε έναν τυχαίο πρώτο p τάξης 128 bits, ενώ για την επιλογή μιας μεγάλης υποομάδας του $\mathbb{Z}_p$ χρησιμοποιείστε την εντολή "g=mod(primitive_root(p),p)".

Απάντηση

In [18]:

```
def generate_parameters5(n):
    p=next_prime(ZZ.random_element(2^n))
    F=GF(p)
    q=(p-1)/2
    i=1
    g=mod(primitive_root(p),p)
    Zp = IntegerModRing(p)
    #while true:
     #   g=F(i)
      #  if multiplicative_order(Zp(g))==(p-1) :
       #     break
        #i+=1
    return p,q,g,F

def public_private_pair(p,q,g,F):
    x=F(randint(2,p-1))
    X=F(g^x)
    return X,x

def generate_secret(X,y):
    return X^y

def str2nums(s):
    return ZZ(map(ord,s),128)

def num2str(n):
    dgs=n.digits(128)
    return ''.join(map(chr,dgs))

P,Q,G,F=generate_parameters5(128)
print 'p=',P,'q=',Q,'g=',G,'F=',F
X,x=public_private_pair(P,Q,G,F)
print 'Alice private',x,' and public=',X
Y,y=public_private_pair(P,Q,G,F)
print 'Bob private=',y ,'and public=',Y
A=generate_secret(X,y)
B=generate_secret(Y,x)
print 'secretA=',A
print 'secretB=',B

k=str2nums("Next Monday")
print ('text=Next Monday')
print 'text from strings to numbers,(with base 128)=',k

ciphertext=F(k*A)
print 'ciphertext=',ciphertext
plaintext=F(ciphertext*lift(B^(-1)))
print 'plaintext=',plaintext
ore=(plaintext)


print 'plaintext=',num2str(Integer(ore))
```

```
p= 104945676306263783098786810432391632831 q= 52472838153131891549393405216195816415 g= 3 F= Finite Field of size 104945676306263783098786810432391632831
Alice private 25521182403357241671039552670969679646  and public= 102047847284946052478006066467302157604
Bob private= 5798263878769055830478897150051865333 and public= 39468259196828598651064085356896330215
secretA= 87963619728028938315731208581542866234
secretB= 87963619728028938315731208581542866234
text=Next Monday
text from strings to numbers,(with base 128)= 143753521369118047875790
ciphertext= 85570927765862989464608323833808704053
plaintext= 143753521369118047875790
plaintext= Next Monday
```

In [0]:

```

```

Άσκηση 6

Το Πρόβλημα του Διακριτού Λογαρίθμου (DLP)

1. Έστω $p = 499$, $g = 7$ και $X = 297$. Να βρεθεί x τέτοιο ώστε $X = g^x$.
2. Έστω $p = 863$, $g = 5$, $X = 543$ και $Y = 239$. Να βρεθούν x και y τέτοια ώστε $X = g^x$ και $Y = g^y$.
3. Έστω $p = 7589$, $g = 2$, $X = 6075$ και $Y = 1318$. Να βρεθεί το κοινό μυστικό κλειδί σε μια ανταλλαγή κλειδιού DH με αυτές τις παραμέτρους.

Απάντηση

In [60]:

```
#ερωτημα 3
p=7589
F=GF(p)
g=F(2)
X=F(6075)
Y=F(1318)
x=X.log(g)
print x
y=Y.log(g)
print y
print g^x==X
print g^y==Y
print 'X common key',X^y
print 'Y common key',Y^x
```

```
3239
5413
True
True
X common key 6803
Y common key 6803
```

In [53]:

```
#ερωτημα 1
p = 499
F = GF(p)
g = F(7)
X = F(297)
x=X.log(g)
print x
print g^x==X
```

```
362
True
```

In [56]:

```
#ερωτημα 2
p=863
F=GF(p)
g=F(5)
X=F(543)
Y=F(239)
x=X.log(g)
print x
y=Y.log(g)
print y
print g^x==X
print g^y==Y
```

```
536
762
True
True
```

In [0]:

```

```

Άσκηση 7

Ανταλλαγή κλειδιών – ECDH

Δίνονται:

- μια ελλειπτική καμπύλη $E(\mathbb{F}_p)$, όπου $p = 63709$, με εξίσωση $Y^2 = X^3 + 26484 * X + 15456$
- μια μεγάλη υποομάδα της με τάξη $q = 63839 \leq order(E)$
- ένα σημείο βάσης $G = (53819, 6786)$ (= γεννήτορας της υποομάδας)

Αναπτύξτε:

1. μια συνάρτηση που θα δέχεται μια καμπύλη E, ένα σημείο βάσης P επί της E και θα επιστρέφει (παράγει) τον ιδιωτικό βαθμωτό πολλαπλασιαστή (ιδιωτικό κλειδί) x και το δημόσιο κλειδί (σημείο) X.
2. μια συνάρτηση που θα δέχεται το δημόσιο και ιδιωτικό κλειδί και θα υπολογίζει το (κοινό) μυστικό κλειδί-σημείο
3. εφαρμόστε τις συναρτήσεις αυτές προκειμένου να προσομοιώσετε μια ανταλλαγή κλειδιών (ECDH) για τις παραπάνω παραμέτρους.

Απάντηση

In [2]:

```
def key_gen7(E,P):
    x=randint(1,P.order())
    return x,x*P

def common_key7(Qx,y):
    return Qx*y

p=63709
K=GF(p)
E = EllipticCurve(K,[26484,15456])
G=E(53819,6786)
a,Qa=key_gen7(E,G)
b,Qb=key_gen7(E,G)
print 'a=',a,'Qa=',Qa
print 'b=',b,'Qb=',Qb
A=common_key7(Qb,a)
B=common_key7(Qa,b)
print 'A common key=',A
print 'B common key=',B
```

```
a= 22869 Qa= (26306 : 54335 : 1)
b= 52786 Qb= (61425 : 63271 : 1)
A common key= (4572 : 33039 : 1)
B common key= (4572 : 33039 : 1)
```

Άσκηση 8

Κρυπτοσύστημα ελλειπτικού El Gamal

Έστω η ελλειπτική καμπύλη $\mathrm{E} : Y^2 = X^3 + X + 6$ πάνω στο $\mathbb{F}_{11}$.

1. Προσδιορίστε τα σημεία της.
2. Επειδή κάθε ομάδα με τάξη πρώτο αριθμό είναι κυκλική και από το ερώτημα (1) έχετε διαπιστώσει ότι η τάξη της E είναι 13, έπεται ότι κάθε σημείο της εκτός από το $\mathcal{O}$ είναι ένας γεννήτορας. Υποθέστε ότι επιλέγουμε ως γεννήτορα το $G = (2, 7)$ και υπολογίστε όλα τα πολλαπλάσια (δυνάμεις) του G αντιστοιχίζοντάς τα με τα σημεία της $\mathrm{E}(\mathbb{F}_{11})$.
3. Υποθέστε τώρα ότι η Alice και ο Bob συμφωνούν να χρησιμοποιήσουν το κρυπτοσύστημα ελλειπτικού ElGamal με ελλειπτική καμπύλη την E και το σημείο $G \in \mathrm{E}(\mathbb{F}_{11})$. Περιγράψτε τις διαδικασίες κρυπτογράφησης και αποκρυπτογράφησης σε περίπτωση που ο Bob θελήσει να στείλει το μήνυμα $m = (10, 9)$.

Απάντηση

In [40]:

```
#ερωτημα 3
m=E(10,9)
print E.is_on_curve(10,9)
#αρα μπορω να συνεχισω

rA = randint(2,E.order()-1)
rB = randint(2,E.order()-1)
print 'A chooses his private key=',rA
print 'B chooses his private key=',rB
Q_b=rB*G
Q_a=rA*G
print 'A computes his public info=',Q_a
print 'B computes his public info=',Q_b



C_2=m+rA*Q_b
print "The ciphertext is: ", C_2

plaintext=C_2-rB*Q_a
print "The plaintext is: ", plaintext

print m==plaintext
```

```
True
A chooses his private key= 6
B chooses his private key= 12
A computes his public info= (7 : 9 : 1)
B computes his public info= (2 : 4 : 1)
The ciphertext is:  (8 : 3 : 1)
The plaintext is:  (10 : 9 : 1)
True
```

In [36]:

```
#ερωτημα2
G=E(2,7)


array=E.points()
print array[4]
x=[(G*i) for i in range(0,13)]
print x
print sorted(x)
print E.points()
```

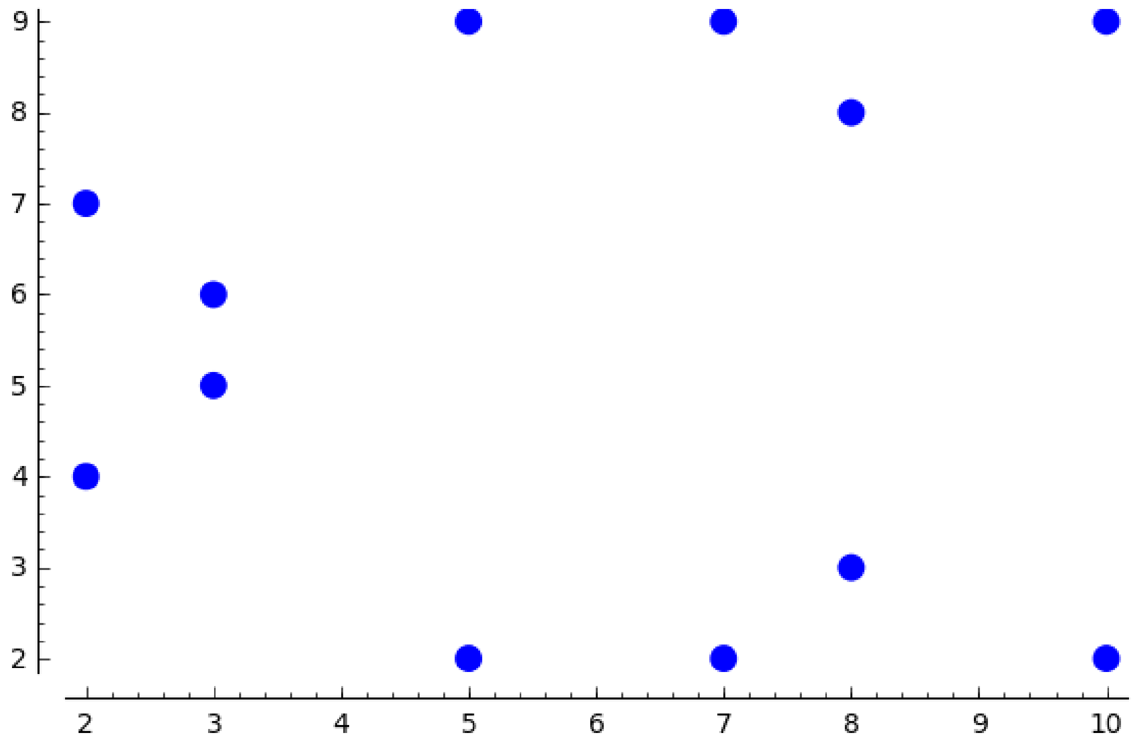
```
(3 : 6 : 1)
[(0 : 1 : 0), (2 : 7 : 1), (5 : 2 : 1), (8 : 3 : 1), (10 : 2 : 1), (3 : 6
 : 1), (7 : 9 : 1), (7 : 2 : 1), (3 : 5 : 1), (10 : 9 : 1), (8 : 8 : 1),
 (5 : 9 : 1), (2 : 4 : 1)]
[(0 : 1 : 0), (2 : 4 : 1), (2 : 7 : 1), (3 : 5 : 1), (3 : 6 : 1), (5 : 2 :
1), (5 : 9 : 1), (7 : 2 : 1), (7 : 9 : 1), (8 : 3 : 1), (8 : 8 : 1), (10 :
2 : 1), (10 : 9 : 1)]
[(0 : 1 : 0), (2 : 4 : 1), (2 : 7 : 1), (3 : 5 : 1), (3 : 6 : 1), (5 : 2 :
1), (5 : 9 : 1), (7 : 2 : 1), (7 : 9 : 1), (8 : 3 : 1), (8 : 8 : 1), (10 :
2 : 1), (10 : 9 : 1)]
```

In [9]:

```
#ερωτημα1
p=11
K=GF(p)
E = EllipticCurve(K,[1,6])
print E.order(),E.points()
E.plot(pointsize=100)
```

```
13 [(0 : 1 : 0), (2 : 4 : 1), (2 : 7 : 1), (3 : 5 : 1), (3 : 6 : 1), (5 :
 2 : 1), (5 : 9 : 1), (7 : 2 : 1), (7 : 9 : 1), (8 : 3 : 1), (8 : 8 : 1),
 (10 : 2 : 1), (10 : 9 : 1)]
```

Out[9]:



In [0]:

```

```

Άσκηση 9

Κωδικοποίηση Koblitz και κρυπτοσύστημα ελλειπτικού El Gamal

Δίνεται μια ελλειπτική καμπύλη $E(\mathbb{F}_p)$, όπου $p = 593899$, με εξίσωση $Y^2 = X^3 + 7 * X + 11$ και το μήνυμα $m = 12345$. Κωδικοποιείστε κατάλληλα το μήνυμα m (δηλαδή αναπαραστήστε το ως σημείο της καμπύλης) και εν συνεχεία κρυπτογραφήστε το με το κρυπτοσύστημα του ελλειπτικού El Gamal. Εν συνεχεία προβείτε σε αποκρυπτογράφηση και αποδωδικοποίηση προκειμένου να ανακτήσετε το αρχικό μήνυμα m.

Απάντηση

In [2]:

```
p=593899
K=GF(p)
E = EllipticCurve(K,[7,11])
m=12345
G = E.random_point()

print G
f(x)=x^3+7*x+11

k=10
print legendre_symbol(f(m*k+0),p)
print legendre_symbol(f(m*k+1),p)

x=m*k+1
print x
z=x^3+7*x+11
y=Mod(z,p).sqrt()
print y
point=E(x,y)
print point
print floor(x/k)

rA = randint(2,E.order()-1)
rB = randint(2,E.order()-1)
print 'A chooses his private key=',rA
print 'B chooses his private key=',rB
Q_b=rB*G
Q_a=rA*G
print 'A computes his public info=',Q_a
print 'B computes his public info=',Q_b



C_2=point+rA*Q_b
print "The ciphertext is: ", C_2

plaintext=C_2-rB*Q_a
print "The plaintext is: ", plaintext
a,b,c=plaintext

print a,x,a/k

print parent(a)
print parent(x)
#πρεπει να μετατρεψω τον a σε integer στο συνολο Z απο το συνολο Zp* που ηδη ανηκει

plain=floor(int(a)/k)
print plain

print m==plain
```

```
(199195 : 17420 : 1)
-1
1
123451
170809
(123451 : 170809 : 1)
12345
A chooses his private key= 22180
B chooses his private key= 433869
A computes his public info= (152339 : 39146 : 1)
B computes his public info= (164770 : 288574 : 1)
The ciphertext is:  (167321 : 482166 : 1)
The plaintext is:  (123451 : 170809 : 1)
123451 123451 71735
Finite Field of size 593899
Integer Ring
12345
True
```

Άσκηση 10

Ψηφιακές υπογραφές

Δοθέντος ενός σχήματος ψηφιακών υπογραφών RSA με δημόσιο κλειδί $n = 9797, e = 131$

- Ποιες από τις παρακάτω υπογραφές είναι έγκυρες;
    1. $(x = 123, sig(x) = 6292)$
    2. $(x = 4333, sig(x) = 4768)$
    3. $(x = 4333, sig(x) = 1424)$
- Δείξτε πώς μπορεί ο αντίπαλος να φέρει εις πέρας μια επίθεση υπαρξιακής πλαστογράφησης, δίνοντας ένα παράδειγμα με τις παραμέτρους αυτού του συστήματος.

Απάντηση

In [34]:

```
n=9797
e=131
xa=123
xb=4333
xc=4333
siga=6292
sigb=4768
sigc=1424

print xa%n==power_mod(siga,e,n)#εγκυρη η 1
print xb%n==power_mod(sigb,e,n)#μη-εγκυρη η 2
print xc%n==power_mod(sigc,e,n)#εγκυρη η 3

#φτιαχνω μια εγκυρη ψηφιακη υπογραφη, ωστε να νομιζει ο B οτι ειμαι η A ,ενω κανονικα ε
ιμαι ο F
s=ZZ.random_element(1,n)
m=power_mod(s,e,n)

print 'Fred signature=',s
print 'Fred message=',m

print m%n==power_mod(s,e,n)#εγκυρη η υπογραφη
```

```
True
False
True
Fred signature= 4536
Fred message= 6847
True
```

In [0]:

```

```