University College London
Department of Computer Science

Cryptanalysis Lab 8

P. Spacek

April 26, 2018

peter.spacek@stuba.sk
Version 1.0

Implementing the Matsumoto Imai (C *)

Our task now is to generate public key in Matsumoto-Imai cryptosystem. We will use construction described in Neal Koblitz: "Algebraic Aspects of Cryptography"; (Springer, ACM3, 1998, Chapter 4: "Hidden Monomial Cryptosystems", pp. 80-102.).
Let x be column vector $x = (x_1, x_2, ..., x_n)$ and y analogicly $y = (y_1, y_2, ..., y_n)$. We generate two random n x n affine matrices S and T (T has to be invertible), and two random vectors v_s and v_t. We compute two vectors u and v:

$$u = (S * x) + v_s$$

and

$$v = (T * y) + v_t$$

More pricesly we want to compute y from this equation:

$$y = T^{-1} * (v - v_t)$$

The relation between u and v is given by equation:

$$v = u^{q^\theta} * u$$

In practice we do it like this:

$$u' = (u_i * b_i^{q^\theta}) * (u_i * b_i)$$

EXERCISE 1.

(a) Copy this code into sage:

```
q=2; n=5; th=3

K=GF(q)
P=PolynomialRing(K, 'x', n)
P.inject_variables()
K2=GF(q^n, 'x')
fx=K2.modulus()

l=var('x%i' % n)
MVP=PolynomialRing(K, 'x', n+1)
MVP.inject_variables()
xn=MVP(l)
```

```
with localvars(fx.parent(), [l]):
    fx=MVP(fx)
```

(b) First, we are going to write vector x (message). Write Sage code for column vector $mes = (x_1, x_2, ..., x_n)$

(c) We need to define the basis. Write code in Sage for definition of the base.

(d) Now we try to test if q^θ and q^{n-1} are comprime. We need this condition to be able to invert q^θ (decryption). Write Sage code for this test.

(e) We generate two random n x n affine matrices S and T (T has to be invertible), and two random vectors v_s and v_t of length n. Do it in Sage.

(f) Compute $u = (S * mes) + v_s$ in Sage.

(g) Compute u' in Sage. Note that u' is computed in this way: $u' = (u_i * b_i^{q^\theta}) * (u_i * b_i)$. Don't forget that we are in the field, so use mod $f(x)$. This is where we get quadratic equation.

(h) Because the relation between u and v is given by equation $v = u^{q^\theta} * u$, we need to extract v from u' (back from the basis). Do it

in Sage.

(i) The last step is to calculate y (public key) from v. Note that $v = (T * y) + v_t$. Print the result. Nice, isn't it?

Hidden Field Equations

Rewrite previous example to more general HFE crypto-system. Read this: https://en.wikipedia.org/wiki/Hidden_Field_Equations

EXERCISE 2.

(a) What part do we need to change to be able to use previous code for HFE?

(b) Change the code.

Solutions to Exercises

Exercise 1(b)

```
MSm = MatrixSpace(P,n,1)
mes=[]
for i in (0..n-1):
    mes.append(var('x%i' % i))
mes=MSm(Matrix(n,1,mes))
```

□

Exercise 1(c)

```
b=[]
for i in (0..n-1):
    b.append(xn^i)
```

□

Exercise 1(d)

```
h=(q^th)+1
if (gcd(h,q^n-1)!=1):
    print "error"
hp=inverse_mod(h,q^n-1)
```

□

Exercise 1(e)

```
MSnn_2 = MatrixSpace(K,n,n)
S=MSnn_2.random_element()
T=MSnn_2.random_element()

while (T.is_invertible()==false):
    T=MSnn_2.random_element()

MSn1_2 = MatrixSpace(K,n,1)
vs=MSn1_2.random_element()
vt=MSn1_2.random_element()
```

Exercise 1(f)

```
u=(S*mes)+vs
```

Exercise 1(g)

```
upa=0
upb=0
for i in (0..n-1):
    upa=upa+u[i]*Bas[i]
    upb=upb+u[i]*Bas[i]^(q^th)
up=(upa*upb%fx)
```

Exercise 1(h)

```
v=[]
for i in (1..n):
    v.append([])
tmp=expand(up)
for i in (1..n-1):
    v[i] = P(expand(up).coefficient(xn^i))
    tmp=tmp-v[i]*xn^i
v[0] = P(tmp)
v=Matrix(n,1,v)
```

Exercise 1(i)

```
y= T.inverse()*(v-vt)
print y
```

□

Exercise 2(a)

The difference is in the polynomial used. Vector v is no longer equal to $u^{q^{\theta}+1}$ Instead, $v = P(u)$ where

$$P(u) = \sum c_i * u^{q^{s_i}+q^{t_i}}$$

□

Exercise 2(b)

This is a solution for simple $P(x)$ with one term.

```
upa=0
upb=0
for i in (0..n-1):
    upa=c*(upa+u[i]*Bas[i]^(q^s) )
    upb=upb+u[i]*Bas[i]^(q^t)
up=(upa*upb%fx)
```

To be consistent with the definition, you need to do this in loop for every term up to degree d and sum it. □