Mathematisches Institut
der Universität München
Pascal Reisert

Wintersemester 2015/2016
Übungsblatt 11
20. Januar 2016

Übungen zur Algorithmischen Zahlentheorie

Aufgabe 44

```
p:=0x16C8_F2E4_92E3_2F0C_F57A_20CB_BF63_E5CE_16A4_55CC_FB06_CC8D_8DFA_69ED;
g:=0xAF9_7F3A_C27A_13B5_C1D4_5748_5D58_C47E_B418_44FC_9AD1_36AD_BCE3_C93B;
a:=0xB42_348F_4F59_1BD1_75C3_0926_8B6D_C88C_7DA0_CE38_17F9_F89F_5013_4F05;
b:=0xBE7_4D5F_15BF_BAFC_52C5_A29F_4D3B_799D_B153_065B_9001_91E7_80E7_362E;
y:=$9CA2_7011_D687_67EC_6FF0_9FE5_2604_EB9F_8BFD_19A0_C6D1_C8EC;
```

Da $q|p-1$ gilt, bestimmen wir zunächst alle Primfaktoren von $p-1$:

```
p1:=p2:=2; p3:=p4:=3; p5:=5; p6:=p7:=7; p8:=107; p9:=11131_46207;
p10:=3_39743_00747; p11:=4_88665_90133; p12:=4_98756_19037;
p13:=27584_92626_89640_64337.
```

Da $q \leqslant 2^{36}$ können wir `p13` ausschließen. Nachrechnen liefert

```
v:=(p1,p3,p5,p6,p8,p9,p10,p11,p12);

function ordn(p;g:integer; v:array; var q:integer):integer;
var
k:integer;
begin
for k:=0 to length(v)-1 do;
if g**(v[k]) mod p=1 then q:=v[k] end;
end;
return q;
end;

ordn(p,g,v,q).
-: 4_98756_19037
```

Zur Berechnung des diskreten Logarithmus verwenden wir den folgenden `Aribas`-Code aus Prof. Forsters Buch zur Algorithmischen Zahlentheorie, 21.3:

```
function dlog_rho(p,g,x,q: integer): integer;
var
nu, mu, nu1, mu1, nu2, mu2, y, z, i: integer;
found: boolean;
begin
nu1 := nu2 := 1;
mu1 := mu2 := 1 + random(q-1);
y := z := (g**mu1 mod p) * (x mod p);
found := false;
for i := i to 10*isqrt(q) do
```

```
    y := lrho_step(p,g,x,y,q,nu1,mu1);
z := lrho_step(p,g,x,z,q,nu2,mu2);
z := lrho_step(p,g,x,z,q,nu2,mu2);
    if i mod 1024 = 1 then write('.'); end;
if z = y then
    writeln(" (",i," steps)");
found := true; break;
end;
end;
if found then
    nu := nu2 - nu1; mu := mu1 - mu2;
if gcd(nu,q) = 1 then
    return (mod_inverse(nu,q)*mu) mod q;
end;
end;
return -1;
end;

function lrho_step(p,g,x,y,q:integer; var nu,mu:integer): integer;
var
s: integer;
begin
s := y mod 3;
if s = 0 then
y := y*g mod p;
inc(mu);
elsif s = 1 then y := y**2 mod p;
nu := 2*nu mod q;
mu := 2*mu mod q;
else
y := y*x mod p;
inc(nu);
end;
return y;
end;

alpha:=dlog_rho(p,g,a,q).
==> alpha.
-: 1_05289_90041

beta:=dlog_rho(p,g,b,q);
==> beta.
-: 4_80904_62340

K:=g**(alpha*beta) mod p.
-: 66030_13704_33807_22081_73286_60421_87845_55321_63832_59860_65603_30249_
97361

b:=byte_string(K);
```

```
mem_shift(b,-2*8);
mem_xor(b,y);
string(b).
```

Pohlig-Hellman reduction

Falls die Faktorisierung von $p-1$ nicht gelingt, können wir ähnlich wie in Aufgabe 43 vorgehen. Mittels

```
function dlog_coeff(p,g,x: integer; var nu,mu:integer): array;
var
kappa, chi, kappa1, chi1, kappa2, chi2, y, z, i: integer;
w:array[2];
found: boolean;
begin
kappa1 := kappa2 := 1;
chi1 := chi2 := 1 + random(p-2);
y := z := (g**chi1 mod p) * (x mod p);
found := false;
for i := i to 10*isqrt(p-1) do
y := lrho_step(p,g,x,y,p-1,kappa1,chi1);
z := lrho_step(p,g,x,z,p-1,kappa2,chi2);
z := lrho_step(p,g,x,z,p-1,kappa2,chi2);
if z = y then
found := true; break;
end;
end;
if found then
kappa := kappa2 - kappa1; chi := chi1 - chi2;
nu:=kappa; mu:=chi;
w:=(kappa,chi);
return w;
end;
(-1,-1);
end.
```

erhalten wir ein ν und μ mit $x^\nu = g^\mu \Rightarrow \alpha\nu = \mu \bmod q$. Wir suchen ein $q'|p-1$ sodass $\gcd(\nu, q') = 1$. Da q prim ist gilt entweder $q|\gcd(\nu, p-1)$ oder $q|q'$. Im zweiten Fall finden wir ein λ mit $\nu\lambda + kq' = 1$. Es folgt $\alpha = \mu\lambda$. Für nicht zu kleine q ist der ersten Fall unwahrscheinlich.

```
function dlog_rho(p,g,x,nu,mu: integer):integer;
var
q1,lambda:integer;
begin
q1:=p-1;
while gcd(nu,q1)/=1 do;
q1:=q1 div gcd(nu,q1);
end;
if g**q1 mod p=1 then
```

```
writeln("g wahrscheinlich von kleiner Ordnung");
else lambda:=mod_inverse(nu,q1);
return lambda*mu mod q1;
end;
end.

function comb(p,g,a:integer):integer;
var
mu,nu,k,l:integer;
begin
for k:=2 to 10**6 do;
if (p-1) mod k=0 and g**k mod p=1 then;
for l:=0 to k do;
if g**l mod p=a then return l;
end; end; end; end;
dlog_coeff(p,g,a,nu,mu);
dlog_rho(p,g,a,nu,mu);
end.
```