Fiat-Shamir-Verfahren, Zero-Knowledge-Protokoll

Kryptographie mit MuPAD,
Prof. Dr.Dörte Haftendorn, Okt.99, Nov 02 05

Anton will Berta davon überzeugen, dass er ein Geheimnis s kennt,
ohne dass Berta irgendetwas von dem Geheimnis erfährt.

Schlüsselerzeugung:

Anton wählt zwei große Primzahlen p und q und bildet ihr Produkt n.

```
//p:=numlib::prevprime(floor(sqrt(5*10^50)));
//q:=numlib::prevprime(floor(sqrt(29*10^50)));  // Exponent etwa 200 sta
//n:=p*q;
```

$n = p \cdot q$

```
p:=11;  q:=17;  n:=p*q;
```

11

17

187

```
r:= random(p+1..n-1): s:=r(); // s wird zufällig kleiner n gewählt.
v:=powermod(s,2,n);
```

137

69

$v \equiv_n s^2$ Er teilt v und n öffentlich mit und behält s als sein Geheimnis.

```
v;n;  //Antons öffentliche Schlüssel
```

26

187

Anwendungsphase

Anton wählt r teilerfremd zu n und berechnet x

```
repeat
rd:= random(2..n-1): r:=rd():
  until gcd(n,r)=1 end_repeat:r;   // r zufällig gewählt
```

139

```
x:=powermod(r,2,n); //A Csendet x an B.
```

60

$x \equiv_n r^2$

$$X \equiv_n r^2$$

```
rd:= random(0..1): b:=rd():b;  //B sendet 0 oder 1 an A
```

0

```
if b=0 then y:=r else y:=modp(r*s,n)end_if; // A sendet y an B
```

1152093477429929417497536707604477051898818368959648

$$y \equiv_n r s^b$$

```
test:=powermod(y,2,n);
if b=0 then if test=x then erg:="ok": else erg:="Quark": end_if:
    else if test=modp(x*v,n) then erg:="ok": else erg:="Quark": end_if:
end_if:
erg;
```

24611398824981765368900949917524016001378170968291 8

"ok"

$$y^2 \overset{?}{\equiv}_n x v^b$$

Mehrere Durchläufe, das Fiat-Shamir Verfahren ist ein "Challenge and response" verfahren (Anfrage und Antwort), das bei k Anfragen, die zu "ok" führten, eine Irrtumswahrscheinlichkeit von 2^(-k) hat, für die Hypothese: Anton kennt das Geheimnis.

```
teste:=proc()
 local rd,r,x,test,b,y,erg;
         begin
  repeat
   rd:= random(2..n): r:=rd():
   until gcd(n,r)=1 end_repeat:r;
 x:=powermod(r,2,n); //A sendet x an B.
 rd:= random(0..1): b:=rd():b;  //B sendet 0 oder 1 an A
 y:=modp(r*s^b,n); // A sendet y an B
 test:=powermod(y,2,n);
         if test=modp(x*v^b,n) then erg:="ok" else erg:="Quark": end_if:
        return ([b,test,modp(x*v^b,n),erg]);
   end_proc:
```

```
matrix([teste() $ i=1..10]);
```

0	69	69	"ok"
0	36	36	"ok"
0	59	59	"ok"
1	169	169	"ok"
0	169	169	"ok"
0	42	42	"ok"
0	169	169	"ok"
1	53	53	"ok"
1	155	155	"ok"
1	60	60	"ok"

```
testen:=proc(k)
```

```
testen:=proc(k)
         local i, rd,r,x,test,b,y,fehler;
  begin
         fehler:=0;
         for i from 1 to k do
          repeat
     rd:= random(2..n): r:=rd():
     until gcd(n,r)=1 end_repeat:r;
   x:=powermod(r,2,n); //A sendet x an B.
          rd:= random(0..1): b:=rd():b;  //B sendet 0 oder 1 an A
   y:=modp(r*s^b,n); // A sendet y an B
   test:=powermod(y,2,n);
          if test <> modp(x*v^b,n) then fehler:=fehler+1 end_if:
         end_for;
         return (k, " Anfragen, Fehlerzahl = ", fehler );
  end_proc:
testen(300)
```

300, "Anfragen, Fehlerzahl =", 0