El-Gamal-Verfahren und DSS-Signatur

Mathematik in wxMaxima www.mathematik-verstehen.de Haftendorn Jan 2011+Dez 1

0.1 Handlinghilfen

Achtung: Durch Anklicken der linken Zellmarkierung kann man die
Abschnitte und auch einzelne Zellen aufklappen und auch wieder zuklappen.
Dazu Shift halten, dann werden auch alle Unterebenen aufgeklappt.

Endung *.wxmx ist komfortabel. Ist die Endung *.wxm muss man
erst noch alle Ausgaben neu erzeugen. Mit Strg r werden alle aufgeklappten
Zum Lernen ist es besser die Zellen einzeln (mit Shift+Enter) auszuwerten.

Werte einzelne Zellen aus mit Shift-Enter.
Auswertung in einem Rutsch: Falte alle Abschnitte auf (mit sh klick),
werte alle Zellen mit Strg r aus (auch Menu Cell Alle Zellen auswerten).

0.2 Inhalt

Figure 1:

1 Schüsselerzeugung

Anton, Berta und Tobi wollen gemeinsam kommunizieren.

- 1.1 Verabredung von p und g
- 1.2 Verschlüsselungsfunktion
- 1.3 Öffentliche Schlüsselliste

2 Hilfsfunktionen

3 Sendung

Jeder der drei kann sich nun entschließen, den beiden

- 3.1 Vorbereitung
- 3.2 Verwendung der Nachricht
- 3.3 Sendung
- 3.4 Empfang und Entschlüsselung

4 Signatur mit ElGamal= DSS

DSS= digital Signatur Standard, Anton will eine Nachricht signieren, Berta und Tobi sollen sie verifizieren können

- 4.1 Offene Nachricht und Vorbereitung
- 4.2 Erzeugung der Signatur
- 4.3 Prüfung der digitalen Unterschrift
- 4.4 Veränderung von m

1 Schüsselerzeugung

Anton, Berta und Tobi wollen gemeinsam kommunizieren. Dazu müssen
sie sich auf eine Verschlüsselungsfunktion einigen, und zu zwei
Zahlen p und g ein gemeinsame öffentliche Schlüsselliste erstellen.

1.1 Verabredung von p und g

p wird als Primzahl gewählt.

```
(%i2) p:next_prime(floor(sqrt(random(1.0)*10^122)));
      10^60;
(%o2) 955933888724997527912523177176307803595071404182033940034365 9
(%o3) 100000000000000000000000000000000000000000000000000000000000
```

```
(%i4) g:random(p-2);
      10^60;
(%o4) 770041426341855220060379899895198142903795653803261477366170 1
(%o5) 100000000000000000000000000000000000000000000000000000000000
```

Als kleines Beispiel eignet sich p:113; g:97

1.2 Verschlüsselungsfunktion

Es muss irgendeine invertierbare Funktion gewählt werden.
Sie soll die Nachricht und den jeweiligen Kommunikationsschlüssel miteinander verrechnen.
Der Einfachheit haber wird hier das gewöhnliche Produkt genommen.

```
(%i6) f(k,m):=k*m; f_inv(k,c):=c/k;
```

(%o6) $f(k,m):=k\,m$

(%o7) $f\_inv(k,c):=\frac{c}{k}$

1.3 Öffentliche Schlüsselliste

Die drei wählen jeder für sich geheime Zahlen ta,tb,tt und
bilden den öffentlichen Teil Ihres Schlüssels

Anton

```
(%i8) ta:random(p-2);
      tAnton:power_mod(g,ta,p);
(%o8) 44317933269899592774225259779013058822073778070878532873511111
(%o9) 51119497198417811416208576948705578318485112242393173188900014
```

Berta

```
(%i10) tb:random(p-2);
       tBerta:power_mod(g,tb,p);
(%o10) 70862665705635872589852005553283795109228167782279792755659 35
(%o11) 41602586933000627824267687434748468615736372379026989270426 66
```

Tobi

```
(%i12) tt:random(p-2);
       tTobi:power_mod(g,tt,p);
(%o12) 16954506122776904856571224641968848082594849924454631328879 83
(%o13) 74024682471734127825761696805568115468381216175970585531579 06
```

Die Öffentliche Schlüsselliste ist nun

```
(%i14) tAnton; tBerta; tTobi;
(%o14) 51119497198417811416208576948705578318485112242393173188900014
(%o15) 41602586933000627824267687434748468615736372379026989270426 66
(%o16) 74024682471734127825761696805568115468381216175970585531579 06
```

2 Hilfsfunktionen

```
(%i17) txToZoo(text):=block([i,li,intli,n,z],
                            li:charlist(text),n:length(li), intli:[],
                            for i:1 thru n do (
                               intli:append(intli,[cint(li[i])])   ), print(
            intli:intli-28, print(intli),z:0,
                             for i:1 thru n do (z:z*100+intli[i]), return(z)
                            )$
```

```
(%i18) txToZoo("Affe");
[65,102,102,101]
[37,74,74,73]
(%o18) 37747473
```

```
(%i19) zooToTx(z):=block([i,li,zz,tx],
                          intli:[],zz:z,tx:"",
                          for i:1 while zz>=1 do(
                             intli:append([mod(zz,100)],intli), zz:floor(zz/
                          print(intli), intli:intli+28,  print(intli),
                          for i:1 thru length(intli) do (tx:concat(tx,ascii(
                          return(tx)
                          )$
```

```
(%i20) m:zooToTx(37747473);
[37,74,74,73]
[65,102,102,101]
(%o20) Affe
```

3 Sendung

Jeder der drei kann sich nun entschließen, den beiden anderen etwas zu senden. Hier will Anton an Berta und Tobi eine geheime Nachricht senden.

3.1 Vorbereitung

Anton wählt geheim ein a und bildet eine Zahl antonOffen, der er zusammen mit der verschlüsselten Nachricht verschicken wird.

```
(%i21) a:random(p-2);
       antonOffen:power_mod(g,a,p);
(%o21) 36352497017597837767440605183728295629232224345072312300491 5
(%o22) 37194052843449236618471557397904138378795084455115419910152 93
```

Er berechnet für die Kommunikation mit Berta einen Schlüssel k_AB.
Er berechnet für die Kommunikation mit Tobi einen Schlüssel k_AT.

```
(%i23) k_AB:power_mod(tBerta,a,p);
       k_AT:power_mod(tTobi,a,p);
(%o23) 9386932223782158351904995832311011024926072642407898747919021
(%o24) 5322778752892651722004466638850986717630666597985762944836462
```

3.2 Verwendung der Nachricht

```
(%i36) m:txToZoo("Erster Montag 2012 um Zehn");
[69,114,115,116,101,114,32,77,111,110,116,97,103,32,50,48,49,50,32
,117,109,32,90,101,104,110]
[41,86,87,88,73,86,4,49,83,82,88,69,75,4,22,20,21,22,4,89,81,4,62
,73,76,82]
(%o36) 41868788738604498382886975042220212204898104627376 82
```

```
(%i37) A_an_Berta:f(k_AB,m);
       A_an_Tobi:f(k_AT,m);
(%o37)
39301948218113411271911848159 9[53 digits]185926300789800778760101249322
(%o38)
22285829910719515248980567728 9[53 digits]796832208491801547338294961084
```

Achtung, Berta und Tobi erhalten für dieselbe Nachricht verschiedene Kryptogramme.

3.3 Sendung

Berta erhält

```
(%i39) antonOffen;
       A_an_Berta;
(%o39) 37194052843449236618471557397904138378795084455115419910152 93
(%o40)
39301948218113411271911848159 9[53 digits]185926300789800778760101249322
```

Tobi erhält

```
(%i41) antonOffen;
       A_an_Tobi;
(%o41) 37194052843449236618471557397904138378795084455115419910152 93
(%o42)
22285829910719515248980567728 9[53 digits]796832208491801547338294961084
```

3.4 Empfang und Entschlüsselung

Berta stellt sich auch ihren Kommnuikationsschlüssel kb_AB
für Nachrichen von Anton her. Es ist derslbe Schlüssel wie k_AB.

```
(%i43) kb_AB:power_mod(antonOffen,tb,p);
       is(kb_AB=k_AB);
(%o43) 93869322237821583519049958323110110249260726424078987479190 21
(%o44) true
```

Berta rechnet

```
  -->  B_von_Anton:f_inv(kb_AB,A_an_Berta);
       klarB:zooToTx(B_von_Anton);
 (%o36) 41868788738604498382886975042220212104898104377176 88
[41,86,87,88,73,86,4,49,83,82,88,69,75,4,22,20,21,21,4,89,81,4,37
,71,76,88]
[69,114,115,116,101,114,32,77,111,110,116,97,103,32,50,48,49,49,32
,117,109,32,65,99,104,116]
 (%o37) Erster Montag 2011 um Acht
```

Tobi stellt sich auch seinen Kommnuikationsschlüssel kc_AT
für Nachrichen von Anton her. Es ist derslbe Schlüssel wie k_AT.

```
(%i45) kt_AT:power_mod(antonOffen,tt,p);
       is(kt_AT=k_AT);
 (%o45) 532277875289265172200446663885098671763066659798576294483646 2
 (%o46) true
```

Tobi rechnet

```
(%i47) T_von_Anton:f_inv(kt_AT,A_an_Tobi);
       klarT:zooToTx(T_von_Anton);
 (%o47) 41868788738604498382886975042220212204898104627376 82
[41,86,87,88,73,86,4,49,83,82,88,69,75,4,22,20,21,22,4,89,81,4,62
,73,76,82]
[69,114,115,116,101,114,32,77,111,110,116,97,103,32,50,48,49,50,32
,117,109,32,90,101,104,110]
 (%o48) Erster Montag 2012 um Zehn
```

4 Signatur mit ElGamal= DSS

DSS= digital Signatur Standard, Anton will eine Nachricht
signieren, Berta und Tobi sollen sie verifizieren können

4.1 Offene Nachricht und Vorbereitung

```
(%i49) klar:"Vertraut Emil nicht!";
 (%o49) Vertraut Emil nicht!
```

```
(%i50) m:txToZoo(klar);
[86,101,114,116,114,97,117,116,32,69,109,105,108,32,110,105,99,104,
116,33]
[58,73,86,88,86,69,89,88,4,41,81,77,80,4,82,77,71,76,88,5]
 (%o50) 587386888669898804418177800482771768805
```

Anton wählt eine Zahl ra teilerfremd zu p-1 und
bestimmt ihre Inverse ri bezüglich p-1.

```
(%i51) p;
 (%o51) 955933888724997527912523177176307803595071404182033940034 3659
```

```
(%i52) ra:2$  while  gcd(ra,p-1)#1  do ra:random(p-1)$ ra;
 (%o54) 732638866678262915059521985258939688438867688756006125415 4527
```

```
(%i55) gcd(ra,p-1);
(%o55) 1

(%i60) ri:gcdex(ra,p-1)[1]$ if ri<0 then ri:ri+p-1$ ri;
(%o62) 47777993791451802987835835535780728732185481166755704131480 83
```

Probe

```
(%i63) mod(ra*ri,p-1);
(%o63) 1

(%i64) kra:power_mod(g,ra,p);
(%o64) 254511685344681945881372997319483887178198996481591403688 9965
```

4.2 Erzeugung der Signatur

```
(%i65) sa:mod((m-ta*kra)*ri,p-1);
(%o65) 188773214742214384645558962068974418935877679903502997562 2916
```

Anton sendet nun

```
 -->  [klar,m,sa,kra];
(%o54) [Vertraut Emil nicht!,5873868886698988044181778004827771768805,
188773214742214384645558962068974418935877679903502997562 2916,
254511685344681945881372997319483887178198996481591403688 9965]
```

4.3 Prüfung der digitalen Unterschrift

Jeder, der p,g und tAnton kennt, kann prüfen, ob dies
wirklich Antons Text ist.
Er berechnet zwei Zahlen und prüft, ob die gleich sind.

```
(%i72) test1:power_mod(g,m,p);
       test2:mod(power_mod(tAnton,kra,p)*power_mod(kra,sa,p),p);
(%o72) 33276941508998169598057690475839262001144325119817966592132 96
(%o73) 33276941508998169598057690475839262001144325119817966592132 96

(%i74) if test1=test2 then print("Anton hat wirklich geschrieben: ",klar)
          else print("Vorsicht, der Text ",zooToTx(m), " ist nicht von Anton"
Anton hat wirklich geschrieben:  Vertraut Emil nicht!
```

4.4 Veränderung von m

```
(%i75) klarx:"Vertraut Emil!"$
       m:txToZoo(klarx);
[86,101,114,116,114,97,117,116,32,69,109,105,108,33]
[58,73,86,88,86,69,89,88,4,41,81,77,80,5]
(%o76) 5873868886698988044181778005

(%i77) test1:power_mod(g,m,p);
       test2:mod(power_mod(tAnton,kra,p)*power_mod(kra,sa,p),p);
(%o77) 47857035808003899214652950598152957620860676096547246941 0562
(%o78) 33276941508998169598057690475839262001144325119817966592132 96
```

```
(%i79) if test1=test2 then print("Anton hat wirklich geschrieben: ",klar)
          else print("Vorsicht, der Text ",klarx, " ist nicht von Anton")$
Vorsicht, der Text  Vertraut Emil!  ist nicht von Anton
```