El_Gamal-Verfahren

El-Gamal-Verfahren und DSS-Signatur

www.mathematik-verstehen.deHaftendorn 2012 **Signatur im 2.Teil**

1 Schlüsselerzeugung

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

1.1. p wird als Primzahl gewählt: (Mehr als 14 Stellen ging nicht.)

$\mathbf{p}:=\mathbf{kry\backslash nextprime}\left(\text{floor}\left(\text{randint}\left(1000,10^{14}\right)\right)\right) \blacktriangleright 93515877911021$

$\mathbf{g}:=\text{randint}\left(1000,\mathbf{p}-2\right) \blacktriangleright 101007867311165$

1.2. Verschlüsselungsfunktion

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

$\mathbf{f}(k,\mathbf{m}):=k\cdot\mathbf{m} \blacktriangleright \textit{Fertig}$ $\mathbf{finv}(k,c):=c/k \blacktriangleright \textit{Fertig}$ Die Divison findet "normal" statt, nicht modulo p wie fast alle nachfolgenden Rechnungen. Darum kann die Nachricht auch viel länger sein als die Schlüssel.

1.1

1.3 Öffentliche Schlüsselliste

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

Anton: **ta**:=randInt(10,**p**−2)

tanton:=**kry\pmod**(**g**,**ta**,**p**)

Berta **tb**:=randInt(10,**p**−2) ▸ 585718616759

tberta:=**kry\pmod**(**g**,**tb**,**p**) ▸ 50089582435202

Tobi **tt**:=randInt(10,**p**−2) ▸ 51338924569939

ttobi:=**kry\pmod**(**g**,**tt**,**p**) ▸ 24289576046835

Die öffentliche Schlüsselliste ist nun

Anton **tanton** ▸ 12472395282269

Berta **tberta** ▸ 50089582435202

Tobi **ttobi** ▸ 24289576046835

1.2

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 ein a und bildet eine Zahl **antonoffen** , die der zusammen mit der verschlüsselten Nachricht schicken wird.

a:=randInt(10,**p**−2) ▸ 80010344218109

antonoffen:=**kry\pmod**(**g**,**a**,**p**) ▸ 28709408957110

Er berechnet für die Kommunikation mit Berta einen Schlüssel

k_ab:=**kry\pmod**(**tberta**,**a**,**p**) ▸ 56923304170682

Er berechnet für die Kommunikation mit Tobi einen Schlüssel

k_at:=**kry\pmod**(**ttobi**,**a**,**p**) ▸ 30217639169933

3.2 Verwendung der Nachricht Text-to-Zahl-Funktionensind am Ende!

m:=**txtozoo**("Nach dem Krieg um Acht. ")
▸ 506971760472738104478677737504898104377176881 8

1.3

a_an_berta:=f(k_ab,m)

▸ 28858507727335808854631419518653189877296788555787717393876

a_an_tobi:=f(k_at,m)

▸ 153194897273109015550947713776349156172737004245048692549194

Achtung: Berta und Tobi erhalten für dieselbe Nachricht verschiedene Kryptogramme. Beide erhalten aber dieselbe Zahl **antonoffen** ▸ 28709408957110 dazu.

3.3. Entschlüsselung der empfandenen Daten

Berta errechnet aus antonoffen mit ihrem geheimen tb auch einen Kommunikationsschlüssel

kb_ab:=kry\pmod(antonoffen,tb,p) ▸ 56923304170682

Tobi errechnet aus antonoffen mit seinem geheimen tt auch einen Kommunikationsschlüssel

kt_ab:=kry\pmod(antonoffen,tt,p) ▸ 30217639169933

Diese Schlüssel stimmen mit denen, die Anton geheim erzeugt hatte ,überein:

k_ab ▸ 56923304170682 k_at ▸ 30217639169933

1.4

Diese Übereinstimmung muss man beweisen!!! (Siehe unten)

Aber wegen dieser Übereinstimmmung können Berta und Tobi nun mit Erfolg die inverse Verschlüsselungsfunktion anwenden.

Berta rechnet **b_von_anton:=finv(kb_ab,a_an_berta)**

▸ 506971760472738104478677737504898104377176881 8

und **klarb:=zootot x(b_von_anton)** ▸ Nach dem Krieg um Acht.

Tobi rechnet **t_von_anton:=finv(kt_ab,a_an_tobi)**

▸ 506971760472738104478677737504898104377176881 8

und **klart:=zootot x(t_von_anton)** ▸ Nach dem Krieg um Acht.

Text aus "Woyzeck" von Georg Büchner, siehe Wikipedia

Beweis: kb_ab:=antonoffen $^{tb}=(g^{a})^{tb}=(g^{tb})^{a}=(tberta)^{a}=k\_ab$ und naürlich auch

kt_ab:=antonoffen $^{tt}=(g^{a})^{tt}=(g^{tt})^{a}=(ttobi)^{a}=k\_at$ q.e.d.

Im Beispiel **kb_ab=k_ab** ▸ true **kt_ab=k_at** ▸ true

1.5

Hilfsfunktionen 1

ctest:=txtozoo ("Der Affe rennt! ")

▸ 407386043774747304867382828805

Grundidee:

Text to Zahl 00 (zweistellig):

Von dem Text werden von links immer ein Buchstabe weggenommen, mit ord wird seine ASCII-Nummer bestimmt. Von der werden 28 abgezogen, damit das Ergebnis zweistellig ist. Aus der Liste li dieser zweistelligen Zahlen wird dann eine Zahl gebildet. Dazu werden die immer das bisherige Ergebnis mit 100 multipliziert und dann eine neue Zahl der Liste genommen.

txtozoo 11/11

```
Define LibPub txtozoo (tex)=
Func
Local i,li,intli,n,z,d,te
li:={}:d:=dim(tex): te:=tex
For i,1,d
 li:=augment(li,{ord(left(te,1))-28})
 te:=right(te,d-i)
EndFor
z:=0
For i,1,d
z:=100·z+li[i]
EndFor
Return z
EndFunc
```

1.6

Hilfsfunktionen 2

zoototx(**ctest**) ▸ Der Affe rennt!

state:=**txtozoo**("Krypto ist gut ")
▸ 4786938488830477878804758988

zoototx(**state**) ▸ Krypto ist gut

Grundiee:

Die letzten beiden Ziffen werden genommen und um 28 vermehrt in eine Liste geschrieben. Die Zahl wird hinten um zwei Stellen gekürzt.

Dies wird wiederholt, wobei die Liste nach links verlängert wird.

Mit char werden die Buchstaben aus den ASCII−Nummern wieder hergestellt

und dann von links nach rechts aneinandergehängt.

zoototx 0/11

```
Define LibPub zoototx(z)=
Func
Local i,zz,li,te,teli
li:={}: zz:=z: te:=""
While zz≥1
 li:=augment({mod(zz,100)+28},li)
 zz:=floor(zz/100)
EndWhile
teli:=char(li)
For i,1,dim(li)
  te:=te&teli[i]
EndFor
Return te
EndFunc
```

1.7

El_Gamal-Signatur

Signatur mit dem El-Gamal-Verfahren, DSS-Signatur

www.mathematik-verstehen.deHaftendorn 2012

Im 1. Teil ist das El-Gamal-Verfahren kommentiert dargestellt. Dieses ist der 2. Teil **Signatur** mit dem El-Gamal-Verfahren, **DSS Digital Signation Standard.** Hier agiert nur Anton, er will einen Text signieren. Er verwendet p prim und g aus dem El-Gamalverfahren. Die Verifizierer, also Berta und Tobi brauchen nur **p,g,** den El-Gamal-Schlüssel **tanton**, den Klartext und die zur Signatur von Anton erzeugten drei Zahlen. Sie brauchen nicht ihre eigenen El-Gamal-Schüssel.

1. Schlüsselerzeugung 1.1. p wird als Primzahl und g werden gewählt: (Mehr als 14 Stellen ging nicht.)**p**:=**kry\nextprime**(floor(randint(100000,10 ^{14}))) ▸ 97708424649959

g:=randint(1000,**p**−2) ▸ 27193117001079

1.2. Antons geheime Zahl **ta**:=randInt(10,**p**−2) ▸ 26890755181010

und sein El-Gamal-Schlüssel **tanton**:=**kry\pmod**(**g**,**ta**,**p**) ▸ 66659475980037

2. Nachricht und ihre Signatur **Text-to-Zahl-Funktionensind am Ende!**

2.1 **klar**:="Vertraut Emil " ▸ "Vertraut Emil "

m:=**txtozoo**(**klar**) ▸ 587386888669898804418177 80

2.1

2.2 Erzeugung der Signatur

Antor wählt zufällig ein $\mathbf{ra} \leq \mathbf{p}-3$ also sinnvoll aus Z(p−1)*:

Anton: **ra**:=randInt(10,**p**−3) ▸ 11899922277605 und bestimmt dazu das Inverse in dieser Gruppe, die für die Exponenten von Potenzen in Z(p) relevant ist.

ea:=**kry\ggte**(**ra**,**p**−1) ▸ [1 40250718941437 -4902140513848] Achtung! oben ra neu Wählen, wenn die erste Zahl nicht 1 ist.

ri:=mod(**ea**[1,2],**p**−1) ▸ 40250718941437 Probe mod(**ra**·**ri**,**p**−1) ▸ 1 Hier muss 1 stehen. Denn: Eulerscher Satz: Für s aus Z(p)* gilt $s^{\mathbf{p}-1}=1$ modulo p

kry\pmod(**g**,**ra**·**ri**,**p**)−**g** ▸ 0

kra:=**kry\pmod**(**g**,**ra**,**p**) ▸ 14185746229194 probe

kry\pmod(**g**,**ri**·**ra**,**p**) ▸ 27193117001079 **g** ▸ 27193117001079

sa:=mod((**m**−**ta**·**kra**)·**ri**,**p**−1) ▸ 53502626673406

-------- Anton gibt bekannt ----------

klar ▸ Vertraut Emil **m** ▸ 587386888669898804418177 80

sa ▸ 53502626673406 **kra** ▸ 14185746229194

2.2

3. Verifizierung

Jeder, der p und g und tanton kennt, kann nun den Klartext verifizieren.

test1:=**kry\pmod**(**g**,**m**,**p**) ▸ 83210723030884

test2:=mod(**kry\pmod**(**tanton**,**kra**,**p**)·**kry\pmod**(**kra**,**sa**,**p**),**p**) ▸ 83210723030884

Beweis: $\mathbf{test2} := \mathbf{tanton}^{\mathbf{kra}} \cdot \mathbf{kra}^{\mathbf{sa}} = \left(\mathbf{g}^{\mathbf{ta}}\right)^{\mathbf{kra}} \cdot \left(\mathbf{g}^{\mathbf{ra}}\right)^{\mathbf{sa}} = \mathbf{g}^{\mathbf{ta}\cdot\mathbf{kra}} \cdot \mathbf{g}^{\mathbf{ra}\cdot\mathbf{sa}} =$

$$\mathbf{g}^{\mathbf{ta}\cdot\mathbf{kra}+\mathbf{ra}\cdot\left(\mathbf{m}-\mathbf{ta}\cdot\mathbf{kra}\right)\cdot\mathbf{ri}} = \mathbf{g}^{\mathbf{ta}\cdot\mathbf{kra}+\mathbf{m}-\mathbf{ta}\cdot\mathbf{kra}} = \mathbf{g}^{\mathbf{m}} = \mathbf{test2}$$

wegen $\mathbf{g}^{\mathbf{ra}\cdot\mathbf{ri}} = \mathbf{g}^{1} = \mathbf{g}$ q.e.d.

Anmerkung: Der Klartext kann ruhig länger sein, als m > p ist unschädlich, denn ma muss ja nicht wiederhergestellt werden. Sowohl bei test1 als auch bei test2 wirkt m nur modulo p−1.

2.3

Hilfsfunktionen 1

ctest:=**txtozoo** ("Der Affe rennt! ")

Grundidee:

Text to Zahl 00 (zweistellig):

Von dem Text werden von links immer ein Buchstabe weggenommen, mit ord wird seine ASCII-Nummer bestimmt. Von der werden 28 abgezogen, damit das Ergebnis zweistellig ist. Aus der Liste li dieser zweistelligen Zahlen wird dann eine Zahl gebildet. Dazu werden die immer das bisherige Ergebnis mit 100 multipliziert und dann eine neue Zahl der Liste genommen.

Die andere Übersetzungsrichtung ist im oberen Teil.

txtozoo1 11/11

```
Define LibPub txtozoo1(tex)=
Func
Local i,li,intli,n,z,d,te
li:={}:d:=dim(tex): te:=tex
For i,1,d
 li:=augment(li,{ord(left(te,1))-28})
 te:=right(te,d-i)
EndFor
z:=0
For i,1,d
z:=100·z+li[i]
EndFor
Return z
EndFunc
```

2.4