El_Gamal-Verfahren

El-Gamal-Verfahren und DSS-Signatur

www.mathematik-verstehen.de Haftendorn 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.)

p:=kry\nextprime(floor(randint(1000,10^14))) • 93515877911021

g:=randint(1000,p−2) • 10100786731165

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.

f(k,m):=k·m • *Fertig* finv(k,c):=c/k • *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) • 513389245699 39

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. ")

• 506971760472738104478677737504898104377 1768818

1.3

a_an_berta:=f(k_ab,m)

• 288585077273358088546314195186531898772967885557787717393876

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)

• 506971760472738104478677737504898104377 1768818

und klarb:=zoototx(b_von_anton) • Nach dem Krieg um Acht.

Tobi rechnet t_von_anton:=finv(kt_ab,a_an_tobi)

• 506971760472738104478677737504898104377 1768818

und klart:=zoototx(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! ")

• 4073860437747473048673828828805

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 ")

• 47869384888304778788047589 88

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.

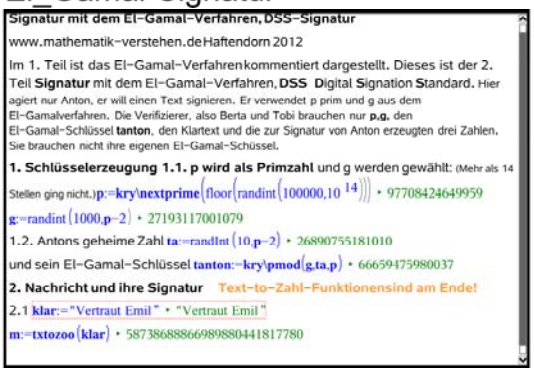
Mit char werden die Buchstaben aus den ASCII−Nummern wieder hergestellt und dann von links nach rechts aneinandergehängt.

zoototx 0/11

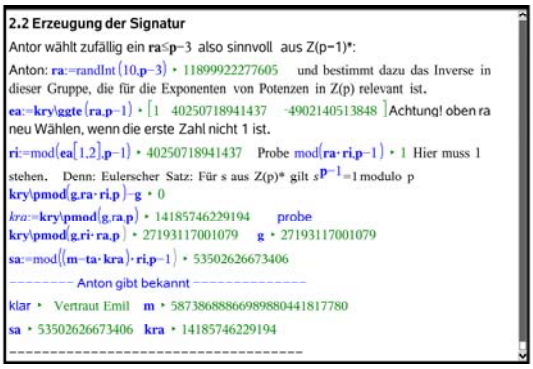
```
Define LibPub zoototx (z)=
Func
Local i,zz,li,te,teli
li:={}: zz:=z: te:=""
While zz≥1
 li:=augment({mod(zz,100)+28},li)
 zz:=floor(zz/100)
EndWhile
teli:=char(li)
For i,1,dim(li)
 te:=te&teli[i]
EndFor
Return te
EndFunc
```

1.7

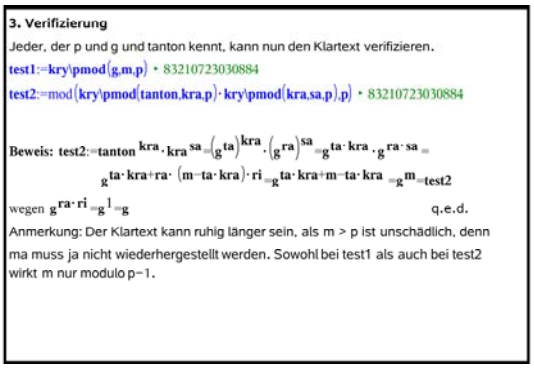
El_Gamal-Signatur



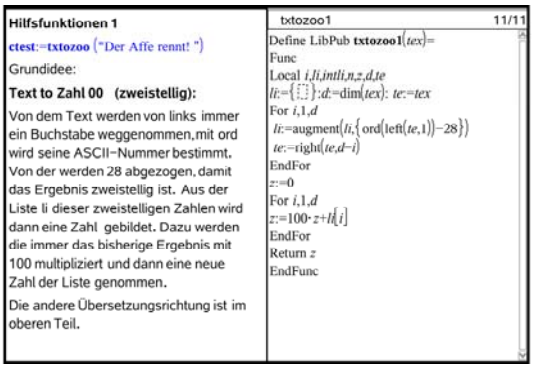
2.1



2.2



2.3



2.4