RSA für kleine Primzahlen

RSA-Verfahren, Version mit kleinen Primzahlen Haftendorn Nov.2010, 2011

Anton möchte, dass jeder ihm Nachrichten schicken kann, die nur er selbst lesen.

Niemand, der die Kommunikation abfängt, soll eine Chance haben, den Klartext herauszubekommen.

Anton bereitet seine Schlüssel vor:

Er wählt zwei Primzahlen **kry\nextprime**(randInt(1,6)) ▸ 7

p:=**kry\nextprime**(randInt(50,200)) ▸ 163 und **q**:=**kry\nextprime**(randInt(50,200)) ▸ 97 .

Das Produkt wird der erste Teil seines öffentlichen Schlüssels **n**:=**p**·**q** ▸ 15811 .

In der Gruppe Z*(n) wird später potenziert, daher braucht er die Ordnung von Z*(n).

phi:=(**p**−1)·(**q**−1) ▸ 15552 . Nun wählt er ein "gutes" **e** aus Z*(phi). "Schlechte" **e** sind solche mit zu kleiner Ordnung. Das passiert bei unseren "kleinen" Beispielen leicht.

e:=randInt(50,**phi**−2) ▸ 713 und prüft sofort **kry\ordo**(**e**,**n**) ▸ 864 und **li**:=**kry\ggte**(**e**,**phi**) ▸ [1 −6151 282]

Wenn hier vorn keine 1 steht oder die Ordnung klein ist, schickt er den Befehl für das Zufalls-e nochmal ab.

Zu diesem **e** braucht das Inverse in Z*(phi). Es ist das zweite

Element dieser Liste (wenn vorn 1 steht) **d**:=mod(**li**[1,2],**phi**) ▸ 9401 , modulo phi genommen.

Falls es negativ war, ist ein n aufaddiert worden.

Probe: mod(**e**·**d**,**phi**) ▸ 1 . Hier muss 1 stehen. Sein d hält er geheim, sein öffentlicher Schlüssel ist

das Zahlenpaar [**e** **n**] ▸ [713 15811]

1.1

Auf **Antons Internetseite steht nun für jeden zu lesen**:

Mein öffentlicher RSA−Schlüssel ist das Zahlenpaar $[e \quad n] \blacktriangleright [713 \quad 15811]$

Anwendungsphase

Berta will Anton eine Nachricht senden

$m := 12884$

$c := kry\backslash pmod(m,e,n) \blacktriangleright 13866$ Dies erhält Anton.

Entschlüsselung, Anton kann die Nachricht lesen

$mm := kry\backslash pmod(c,d,n) \blacktriangleright 12884$ Zur Erinnerung es war $m \blacktriangleright 12884$

Berta darf ihre Nachricht nicht größer als n machen.

$mf := 19847$ wäre so eine falsche Nachricht.

$cf := kry\backslash pmod(mf,e,n) \blacktriangleright 4197$ Dies erhält Anton.

Entschlüsselung, Anton kann die Nachricht nicht !!!!!! lesen

$mmf := kry\backslash pmod(cf,d,n) \blacktriangleright 4036$ Zur Erinnerung es war $mf \blacktriangleright 19847$ Es ist $mod(mf,n) \blacktriangleright 4036$.

1.2

Digitale Signatur mit RSA

Anton will in seinem Internet eine Botschaft signieren, also digital unterschreiben.

Wer diese Klartext−Botschaft liest soll prüfen können, ob sie wirklich von Anton ist.

Mister X könnte ja evt. als Spion im Rechenzentrum Antons Botschaft manipuliert haben.

−−−−−−−−−−−−−−−−

Signatur Anwendungsphase

Mit diesen kleinen Zahlen können wir keinen wirklichen Text bearbeiten. Seine

Klartext−Botschaft sei: **ms**:=9876

Anton berechnet **sig**:=**kry\pmod**(**ms**,**d**,**n**) ▸ 8457 und stellt die "Signatur" daneben.

−−−−−−−−−−−−−−−−

Signatur Prüfungsphase

Berta liest die beides und prüft: **mp**:=**kry\pmod**(**sig**,**e**,**n**) ▸ 9876

Wenn sie auf diese Weise auch wieder die Klartext−Nachricht **ms** ▸ 9876 erhält, hat niemand

Antons Site manipuliert.

−−−−−−−−−−−−−−−−

Info für Informatik−Studierende: Es werden, um das Datenvolumen kleiner zu halten, noch sogenannte Hash−Funktionen zwischengeschaltet (siehe Wikipedia). Das ist für das Verstehen aber unerheblich.