Inhalt

Nachdem wir uns im ersten Teil um den geschichtliche Hintergrund gekümmert haben, einen Blick auf einige mathematische Grundlagen geworfen haben und letztlich ein Verfahren zur Erzeugung des Schlüsselpaares (*öffentlicher und privater Schlüssel*) erarbeitet haben, geht es in diesem Teil um die eigentliche Verschlüsselung. Sie sollten die Artikel „Symmetrie vs. Asymmetrie – Teil I“ und „Symmetrie vs. Asymmetrie – Teil II“ und natürlich „RSA Verschlüsselung Teil I“ gelesen haben.

Kurzer Rückblick *

Wir sollten noch wissen, dass man aus zwei Primzahlen p und q den **RSA Modul** N berechnet. Zusammen mit den **Verschlüsselungsexponenten e** erhält man den **öffentlichen Schlüssel**. Der **private Schlüssel** hingegen besteht aus dem so genannten **Entschlüsselungsexponenten d** und ebenfalls **N**.

Öffentlicher Schlüssel = e,N

Privater Schlüssel = d,N

Division mit Rest *

Um die folgenden Formeln zu verstehen, müssen wir nochmals einen kurzen Ausflug in die Mathematik machen. Wir müssen noch wissen, was es mit der **Division mit Rest** auf sich hat.

> Hat man zwei natürliche Zahlen a und b, wobei b ungleich 0 (durch Null darf man nicht teilen), kann man die Division a / b auch so darstellen.
>
> $$\frac{a}{b} = c, Rest r$$
>
> bzw.
>
> $$a = b \times c + r$$
>
> In Worten: Wie oft passt a in b (c Mal; normale ganzzahlige Division) und was bleibt als Rest r?

Beispiel:

$$\frac{7}{3} = 2 \ Rest \ 1$$
$$\frac{15}{6} = 2 \ Rest \ 3$$
$$\frac{21}{7} = 3 \ Rest \ 0$$

Was uns im Zusammenhang mit **RSA** interessiert, ist lediglich **der Rest**. Die Funktion,

die jedem Zahlenpaar a und b einen Teilerrest r zuordnet, nennt man **Modulo**. Man kürzt dies in einer Formel meist mit **mod** ab.

Beispiel:
7 mod 3 = 1
15 mod 6 = 3
21 mod 7 = 0

Auf Mathe24 können Sie den Modulo berechnen.

Verschlüsseln mit RSA *

Das Tolle an unserem asymmetrischen Verschlüsselungsverfahren ist bekanntlich der Umstand, dass man mit zwei Schlüsseln arbeitet. Mit zwei Schlüsseln, die man wechselseitig nicht auseinander berechnen kann. **Der öffentliche Schlüssel kann frei verteilt werden.** Nichts an ihm ist geheinhaltungswürdig. Jeder kann uns jetzt mit Hilfe dieses öffentlichen Schlüssels eine vertrauliche Nachricht zukommen lassen, die nur wir mit unserem privaten Schlüssel entziffern können. Entsprechend darf man den privaten Schlüssel niemals herausgeben!

Wir haben also eine Nachricht einmal als **Klartext** (*nicht verschlüsselt*) und einmal als **Geheimtext** (*verschlüsselt*).

Beim **Verschlüsseln** kommt folgende Formel zum Einsatz:

$$Geheimtext = Klartext^{e} \, mod \, N$$

[*Wir erinnern uns, e und N bilden zusammen den öffentlichen Schlüssel*]

Beim **Entschlüsseln** verwenden wir folgende Formel:

$$Klartext = Geheimtext^{d} \, mod \, N$$

[*Wir erinnern uns, d und N bilden zusammen den privaten Schlüssel*]

Gehen wir doch einmal ein Beispiel an. In unserem ersten Teil haben wir uns ein

Schlüsselpaar berechnet.

Der **Öffentliche Schlüssel** besteht aus **e und N**, also aus **11 und 221**,

der **Private Schlüssel** aus **d und N**, also aus **35 und 221**.

Nehmen wir einmal an, wir wollen dem Besitzer des öffentlichen Schlüssels eine Botschaft zukommen lassen.

> Unsere Botschaft lautet im Klartext **5**

Klartext und Werte aus dem öffentlichen Schlüssel eingesetzt in unsere Formel

$$Geheimtext = Klartext^{e} \, mod \, N$$

erhalten wir

$$Geheimtext = 5^{11} \; mod \; 221 = 48828125 \; mod \; 221 = 164$$

(*Anm.: 221 passt 220941 Mal in* 5^{11} *, es verbleibt ein Rest von 164*)

Wir schicken jetzt dem Besitzer des öffentlichen Schlüssels unseren Geheimtext 164. Jeder kann diesen Wert mitlesen. Ohne den privaten Schlüssel, den nur der Besitzer kennt, kann man nichts mit dem Geheimtext anfangen.

Der Besitzer setzt Geheimtext und die Werte seines privaten Schlüssels in die Entschlüsselungsformel ein

$$Klartext = Geheimtext^{d} \, mod \, N = 164^{35} \, mod \; 221 = 5$$

Wir erhalten also tatsächlich unsere ursprüngliche **Klartextbotschaft** zurück!

(**Anm**.: *Denken Sie immer daran, dass der Computer im Prinzip nur Zahlen kennt. Es gibt keine Bilder, es gibt keine Töne und es gibt auch keine Buchstaben. Im Speicher Ihres Rechners sind all diese Daten nur Zahlen. Dies leuchtet nicht jedem sofort ein, wodurch im Beispiel der Eindruck erweckt werden könnte, das Verfahren würde nur bei reinen Zahlen funktionieren.*)

Faktorisierungsproblem *

Im Artikel „Primzahlen – Bausteine der natürlichen Zahlen“ lernten wir den **Fundamentalsatz der Arithmetik** kennen.

> Jede natürliche Zahl lässt sich als Produkt von endlich vielen Primzahlen darstellen. Ordnet man die Faktoren der Größe nach, ist die Darstellung eindeutig.

> Und wir haben gesehen, dass man große (Prim)Zahlen leicht miteinander multiplizieren kann, es aber unglaublich schwierig ist, große Zahlen in die Primfaktoren zu zerlegen.

An welcher Stelle im RSA-Verfahren stoßen wir auf dieses Problem?

Wir können beliebig auf den **öffentlichen Schlüssel** zugreifen und haben damit die Werte **e und N**. Aus der folgenden Formel kann man sich den Entschlüsselungsexponenten d berechnen. Zusammen mit dem bekannten N hätte man damit den privaten Schlüssel und die RSA Verschlüsselung wäre geknackt!

> e * d = 1 mod Phi(N)

> Sie erinnern sich an die Eulersche Phi-Funktion?
> Phi(N) = (p-1) * (q-1)

Haben Sie es bemerkt. Das Problem in der obigen Formel besteht, weil wir zwar N, aber nicht die zur Berechnung von Phi(N) nötigen Werten p und q kennen. **Man müsste N faktorisieren!**

> Zur Erinnerung: Diese Primfaktorzerlegung ist für große Zahlen mit heute

> bekannten Verfahren praktisch **undurchführbar**. Ein Beweis, dass es sich bei der **Primfaktorzerlegung** um ein **schwieriges Problem** handelt, existiert jedoch nicht.

OK könnte man denken, über den öffentlichen Schlüssel alleine komme ich nicht zum Ziel. Wie sieht es aber aus, wenn ich zusätzlich einen Geheimtext abfange.

Womit hat man es dann zu tun?

Klartext ^ e = Geheimtext mod N

Wir haben es hier nicht auf d und damit den geheimen privaten Schlüssel abgesehen, sondern auf den Klartext. Um diesen zu erhalten müssten wir die **Wurzel modulo N** berechnen. Ein mit der Faktorisierung, hinsichtlich der Schwierigkeit bei der Berechnung, vergleichbares Problem.

Als Fazit können wir hinsichtlich der Sicherheit sas Folgende festhalten:

- Mit zunehmender Rechenleistung müssen die Primzahlen p und q immer größer gewählt werden, weil die zunehmende Rechenleistung in der Lage sein wird, immer größere Zahlen zu faktorisieren.
- Es ist theoretisch denkbar, dass eine bahnbrechende mathematische Entdeckung die Sicherheit des RSA Systems komplett zunichtemacht.

Im nächsten Teil werden wir uns etwas eingehender damit befassen, wo genau RSA zum Einsatz kommt.