

# Das RSA-Verfahren

5. März 2023

## Inhaltsverzeichnis

1	Das Problem der symmetrischer Verschlüsselung	2
2	Ver-und Entschlüsseln von Nachrichten	5
3	Herstellung eines Schlüsselpaars	5
4	Angriffsmöglichkeiten	5
5	Einsatzgebiete	5

# 1 Das Problem der symmetrischer Verschlüsselung

Alice und Bob arbeiten an einem streng geheimen Projekt. Da beide in verschiedenen Städten wohnen, sind sie sich noch nie begegnet. Alice und Bob müssen nun aber wichtige Nachrichten austauschen. Beide haben Angst, dass Oscar versuchen könnte ihre Nachrichten abzufangen. Sie müssen deshalb verschlüsseln.

Leider hat Alice nicht ein einziges Mal Zeit, sich persönlich mit Bob zu treffen. Aus diesem Grund sendet sie alle ihre Nachrichten mit der Post. Eine symmetrische Verschlüsselung der Nachrichten ist dabei keine gute Idee. Warum?

- ☐ Alice müsste Bob einmal das Entschlüsselungsverfahren mitteilen. Wird der Postbote dabei von Oscar abgefangen, kann Oscar alle Nachrichten entschlüsseln.
- ☐ Alle symmetrischen Verfahren sind einfach zu knacken.
- ☐ Da sich symmetrische Verfahren nur für sehr kurze Nachrichten eignen, müssen die beiden viel zu viele Nachrichten hin und her schicken.

Alice wendet deshalb eine asymmetrische Verschlüsselung, das RSA-Verfahren, an. Das RSA-Verfahren wird in vielen Bereichen des alltäglichen Lebens verwendet. Davon wissen aber nur die wenigsten. Es kommt zum Einsatz bei

- Apps zum Chatten (Telegram, Threema, usw.),
- jedem Öffnen einer sicheren Website (<https://...>) im Internet,
- Bankgeschäften oder
- der Fernwartung von Computern.

In diesem Kapitel wollen wir uns nur den Ablauf anschauen. Wie und warum das Ganze funktioniert und absolut sicher ist, erfährst du in den nächsten Abschnitten.

Öffne die Website

<https://mx3030.github.io/rsa/>

und versuche das Senden einer Nachricht von Alice an Bob nachzuvollziehen und selber auszuprobieren.

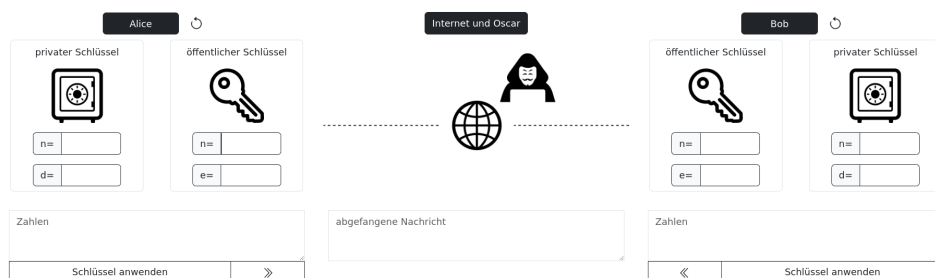



Abbildung 1: Startbildschirm beim Öffnen der Website.

## Alice sendet eine Nachricht an Bob

Drücke auf den Button von Bob um die Perspektive von Bob einzunehmen.

Bob ↻


öffentlicher Schlüssel



n= 33

e= 17

privater Schlüssel



n= 33

d= 13

Durch Drücken von ↻ baut sich Bob ein Schlüsselpaar (siehe Abschnitt 3) bestehend aus einem öffentlichen und einem privaten Schlüssel.


Der öffentliche Schlüssel besteht aus zwei Zahlen  $(n, e)$  und ist frei zugänglich für alle Personen.

Der private Schlüssel besteht aus zwei Zahlen  $(n, d)$  und ist ein Geheimnis von Bob.

Drücke auf den Button von Alice um die Perspektive von Alice einzunehmen.

Alice ↻


privater Schlüssel



n= 143

d= 113


öffentlicher Schlüssel



n= 143


e= 17

Internet und Oscar



Bob ↻

öffentlicher Schlüssel



n= 55

e= 17

Hallo Bob


Schlüssel anwenden >>

Alice verfasst die Nachricht „Hallo Bob“. Sie wählt den öffentlichen Schlüssel von Bob und verschlüsselt ihre Nachricht durch Drücken von Schlüssel anwenden. Durch Drücken von >> sendet sie die verschlüsselte Nachricht zu Bob.

Wechsle in die Perspektive von Bob.

Bob
↺

öffentlicher Schlüssel




n=

55

e=

17

privater Schlüssel



n=

55

d=

33

allezeg

«
Schlüssel anwenden

Nur Bob kann die Nachricht entschlüsseln, da er im Besitz des privaten Schlüssels ist. Wähle den privaten Schlüssel aus und drücke auf Schlüssel anwenden.

### Aufgabe

Sende eine Antwort von Bob an Alice.

### Aufgabe

Wie könnte Oscar versuchen die beiden hereinzulegen?

## 2 Ver-und Entschlüsseln von Nachrichten

Wir wollen nun verstehen, was beim Ver-und Entschlüsseln der Nachrichten passiert. Was läuft also im Hintergrund ab, wenn auf Schlüssel anwenden gedrückt wird.

## 3 Herstellung eines Schlüsselpaars

Damit das Ver-und Entschlüsseln auf diese Art und Weise funktioniert, muss dass eigene Schlüsselpaar nach einer festgelegten Methode gebaut werden. Bei Drücken auf ↻ passiert genau das.

## 4 Angriffsmöglichkeiten

In diesem Abschnitt wollen wir uns in die Perspektive von Oscar versetzen.

## 5 Einsatzgebiete