# Verschlüsselung

https://iuk.one/DS-ENC.pdf

Clemens H. **Cap**
ORCID: 0000-0003-3958-6136

Department of Computer Science
University of **Rostock**
Rostock, Germany
clemens.cap@uni-rostock.de

2026-11-17 18:17



# Übersicht

1. Grundbegriffe

2. Voraussetzungen

3. RSA-Algorithmus

4. Facetten und Grenzen

1. Grundbegriffe

2. Voraussetzungen

3. RSA-Algorithmus

4. Facetten und Grenzen

## 1. Grundbegriffe

Auf ersten Blick ganz einfach...

# Motivation: Einweg-Funktion

## Kernidee der Einweg-Funktion

Die Berechnung ist einfach.
Die Umkehrung ist (typischerweise) kompliziert.

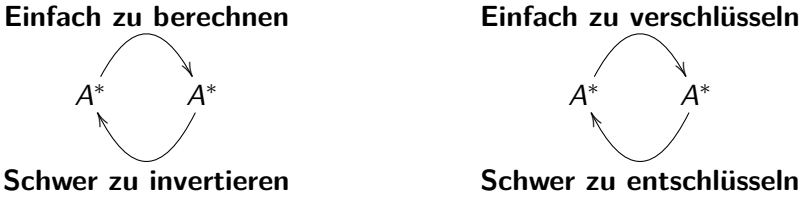



**Abb. 1:** Prinzip der Einweg-Funktion Rechte s. Anhang.

# Beispiel: Einweg-Funktion

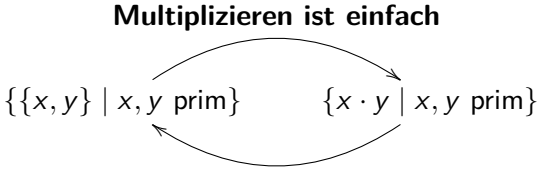



**Abb. 2:** Beispiel für eine Einweg-Funktion im allgemeinen Fall. Rechte s. Anhang.

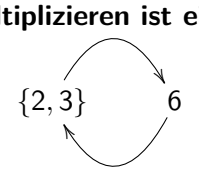



**Abb. 3:** Gegenbeispiel für die behauptete Einweg-Funktion. Grund: Das Produkt enthält eine sehr kleine Primzahl. Rechte s. Anhang.

# Begriff der Typikalität

## Typikalität

Nicht **alle** Instanzen müssen kompliziert sein – nur die **typische** Instanz.

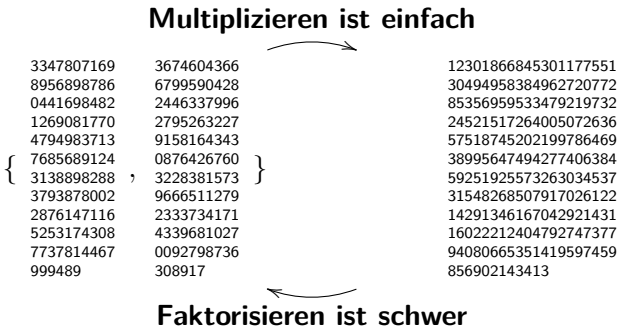



**Abb. 4:** RSA-785 als Instanz einer schwierig zu faktorisierenden Zahl. Rechte s. Anhang.

# Testfrage

Ist die Faktorisierung einer zufällig ausgewählten großen Zahl schwierig?

- Ja
- Nein

https://twbk.de/2va2

# Wichtige Punkte

**Beachte 1:** Jedes Faktorisierungs-Problem, das wir selber aus einer Multiplikation generieren, ist einfach, **wenn** wir uns die Faktoren merken.

**Beachte 2:** Faktorisierungs-Probleme mit "**besonderen**" Strukturen sind einfach.
Beispiel: $10^{300} = 2^{300} \cdot 5^{300}$

**Beachte 3:** Das **generische, typische, zufällig** ausgewählte Faktorisierungsproblem, bei dem die zu faktorisierende Zahl Produkt zweier großer Primzahlen ist, ist schwierig.

**Hinweis:** Das kann (und muß) noch viel präziser ausgearbeitet werden
– aber nicht in einer Einführungsvorlesung.

# Was bedeutet schwierig genau?

**Frage:** Was bedeuten "**einfach**" und "**schwierig**" genau?

**Oft:** Bezug zu NP-vollständigen Problemen bemüht. (siehe VL Komplexität)

**Auch:** Asypmtotische Zeitkomplexität der besten bekannten Algorithmen benutzt.

**Hier:** Sei $n$ die Anzahl Stellen in der Binärdarstellung der Eingabe.

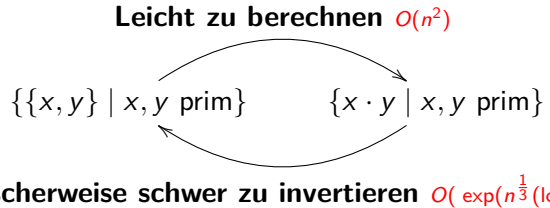



**Abb. 5:** Naive Komplexität von Primzahlprodukten und Komplexität der Zerlegungen nach dem besten bekannten Algorithmus. **Hinweis:** Die Berechnung des Produkts kann über die naive quadratische Komplexität hinaus noch weiter beschleunigt werden, was zunächst einmal intuitiv widersinnig erscheint. Rechte s. Anhang.

# Motivation: Einweg-Funktion mit Falltüre

## Kernidee der Einweg-Funktion mit Falltür

Die Berechnung ist einfach.
Ihre Umkehrung ist (typischerweise) kompliziert.
Mit Zusatzinformation ("Schlüssel") wird die Umkehrung aber wieder einfach.




**Abb. 6:** Prinzip der Einweg-Funktion mit Falltüre. Rechte s. Anhang.

# Anwendung

**Entschlüsselung:** Nur mit bestimmter Zusatzinformation kann ich etwas lesen.

**Authentication:** Nur mit bestimmter Zusatzinformation kann ich meine Identität nachweisen.

**Signature:** Nur mit bestimmter Zusatzinformation kann ich Dokument unterschreiben.

## Wichtige implizite sozio-technologische Regel

| | |
|---|---|
| **Regel:** | **Ich muß diese Zusatzinformation geheim halten.** |
| **Socio-technologisch:** | Technik alleine kann es nicht gewährleisten. |
| **Implizit:** | Nur selten klar ausgedrückt. |

2.1. Modulo-Rechnung

2.2. Euklidischer Algorithmus

2.3. Bezoutsche Identität

2.4. Modulo-Division

2.5. Eulersche Phi-Funktion

2.6. Theorem von Euler

1. Grundbegriffe

2. Voraussetzungen

3. RSA-Algorithmus

4. Facetten und Grenzen

Die mathematischen Voraussetzungen für ein Verständnis des RSA-Verfahrens erwerben.

# Modulo Rechnung

**Definition: Kongruenz**

Zwei ganze Zahlen $a$ und $b$ heißen **kongruent modulo** einer ganzen Zahl $m$, wenn sie **bei Division** durch $m$ **denselben ganzzahligen Rest** haben.

**Wir schreiben:** $a \equiv b \bmod m$

**Formal:** Für $a, b \in \mathbb{Z}, n \in \mathbb{N}$ sei definiert:
$a \equiv b \bmod n \ :\Leftrightarrow \ \exists \alpha, \beta, r \in \mathbb{Z} : a = \alpha \cdot n + r \ \wedge \ b = \beta \cdot n + r$

**Beispiel:** $9 \equiv 4 \bmod 5$ da $9 = 1 \cdot 5 + 4$ und $4 = 0 \cdot 5 + 4$.

# Euklidischer Algorithmus: Eine Wiederholung an einem Beispiel

Der **Euklidische Algorithmus** bestimmt den größten gemeinsamen Teiler zweier Zahlen.

Wir lernen ihn hier kennen (bzw. wiederholen ihn) an einem Beispiel.

**Aufgabe:** Bestimme den größten gemeinsamen Teiler von 228 und 174.

1. Dividiere die größere Zahl (228) durch die kleinere (174).
2. Erhalte ganzzahligen Quotienten und Rest.
3. Erhaltener Quotient: Wird neue größere Zahl.
4. Erhaltener Rest: Wird neue kleinere Zahl.
5. Wiederhole – bis zur Termination bei Rest 0.

$$228 = 1 \cdot 174 + 54$$
$$174 = 3 \cdot 54 + 12$$
$$54 = 4 \cdot 12 + 6$$
$$12 = 2 \cdot 6 + \mathbf{0}$$



**Abb. 7: Euklid** von Alexandrien. Griechischer Mathematiker, 350-279 v. Chr. Erfinder des euklidischen Algorithmus. Rechte s. Anhang.

# Stimmt der Algorithmus?

**Zu zeigen ist hier:**

1. **Termination:** Der Algorithmus **endet** irgend wann mit einem Ergebnis.
2. **Korrektheit:** Das Ergebnis **ist Teiler** von 228 und 174.
   Das Ergebnis ist unter allen Teilern von 228 und 174 der **größte**.

# Euklidischer Algorithmus: Termination

$$228 = 1 \cdot 174 + 54$$
$$174 = 3 \cdot 54 + 12$$
$$54 = 4 \cdot 12 + 6$$
$$12 = 2 \cdot 6 + \mathbf{0}$$

1. Wenn wir mit zwei gleichen Zahlen beginnen, dann ist diese Zahl der ggT.
2. Wir beginnen also mit zwei Zahlen, von denen eine die größere ist.
3. Der Rest ist immer echt kleiner als der Divisor.
4. Der Forschritts-Mechanismus macht also den Divisor immer kleiner.
5. Früher oder später muß das Terminationskriterium erreicht werden.

# Euklidischer Algorithmus: Teiler-Eigenschaft

$$
\begin{aligned}
228 &= 1 \cdot 174 + 54 \\
174 &= 3 \cdot 54 + 12 \\
54 &= 4 \cdot 12 + 6 \\
12 &= 2 \cdot 6 + \mathbf{0}
\end{aligned}
$$

Wir lesen von unten nach oben:

1. 6 teilt 12.
2. 6 teilt 6 und 12, also auch 54.
3. 6 teilt 12 und 54, also auch 174.
4. 6 teilt 54 und 174, also auch 228.
5. 6 ist (ein) gemeinsamer Teiler von 228 und 174.

# Euklidischer Algorithmus: Größte-Eigenschaft

$$228 = 1 \cdot 174 + 54$$
$$174 = 3 \cdot 54 + 12$$
$$54 = 4 \cdot 12 + 6$$
$$12 = 2 \cdot 6 + \mathbf{0}$$

Wir lesen von oben nach unten:

1. Sei $X$ Teiler von 228 und 174.
2. Dann ist $X$ auch Teiler von 54.
3. $X$ ist Teiler von 174 und 54, also auch von 12.
4. $X$ ist Teiler von 54 und 12, also auch von 6.
5. Also ist $X$ kleiner oder gleich 6.
6. Damit ist 6 der größte unter allen Teilern von 228 und 174, weil für jeden Teiler $X$ von 228 und 174 gilt, daß $6 \geq X$ ist.

# Bezoutsche Identität: Aussage und Algorithmus

**Lemma von Bezout**

Der **größte gemeinsame Teiler** $g = \mathrm{ggT}(a, b)$ zweier ganzer Zahlen $a, b$ läßt sich **als ganzzahlige Linearkombination** $g = \alpha \cdot a + \beta \cdot b$ dieser zwei Zahlen darstellen.

**Algorithmus:**

1. Die Quotienten und Reste im Euklidischen Algorithmus unterstreichen.
2. Mit der untersten nicht-trivialen Darstellung des ggT beginnen, dh. dort, wo nicht beide Koeffizienten der Linearkombination 0 sind.
3. Die Quotienten und Reste in dieser Darstellung unterstreichen.
4. Jeweils die kleinere der unterstrichenen Zahlen ersetzen durch einen Ausdruck der sich aus der obersten möglichen Euklidschen Gleichung ergibt.

# Bezoutsche Identität: Beispiel

$$228 = 1 \cdot 174 + 54$$
$$174 = 3 \cdot 54 + 12$$
$$54 = 4 \cdot 12 + 6$$
$$12 = 2 \cdot 6 + \mathbf{0}$$



**Abb. 8:** **Etienne Bezout**, französischer Mathematiker, 1739-1783. Entdecker der Bezoutschen Identität.
Rechte s. Anhang.

$$6 = 1 \cdot 54 - 4 \cdot \mathbf{12}$$
$$6 = -4 \cdot 174 + 13 \cdot 54$$
$$6 = 13 \cdot 228 - 17 \cdot 174$$

es ist: $12 = 1 \cdot 174 - 3 \cdot 54$

es ist: $54 = 1 \cdot 228 - 1 \cdot 174$

**Fertig:** Wir erhalten den ggT 6 als Linearkombination von 228 und 174:

$$6 = 13 \cdot 228 - 17 \cdot 174$$

# Modulo-Division

## Satz von der Modulo-Division

Seien zwei ganze Zahlen $x$ und $n$ teilerfremd. Dann existiert eine ganze Zahl $y$ so, daß $x \cdot y \equiv 1$ **mod** $n$. Die Zahl $y$ nennt man die **Inverse** zu $x$ **modulo** $n$.

**Beweis:**
Seien $x$ und $n$ teilerfremd. Also ist **ggT** $(x, n) = 1$.

Nach dem Lemma von Bezout existieren $\alpha$, $\beta$ so, daß $1 = \alpha \cdot x + \beta \cdot n$.

Diese Gleichung modulo $n$ betrachtet ergibt $1 \equiv \alpha \cdot x + 0$ **mod** $n$

| | |
|---|---|
| **Beispiel:** | Bestimme die Inverse von 5 modulo 9. |
| Nach Bezout: | $1 = (-1) \cdot 9 + 2 \cdot 5$ |
| Wir erhalten: | $1 \equiv 2 \cdot 5$ **mod** 9. |
| **Antwort:** | Die Inverse von 5 modulo 9 ist 2. |

# Eulersche Phi-Funktion

## Definition: Eulersche Phi-Funktion

Die Eulersche Phi Funktion $\phi(n)$ einer natürlichen Zahl ist die **Anzahl** aller positiven ganzen Zahlen kleiner oder gleich $n$, **die keinen gemeinsamen Teiler** mit $n$ haben.

Per Konvention: $\phi(1) = 1$.

**Beispiele:** $\phi(5) = 4$ denn teilerfremd sind: 1, 2, 3, 4.

Ist $p$ Primzahl, dann ist $\phi(p) = p - 1$. Streiche in $\{1, 2, \ldots, p\}$ die Zahl $p$ selber.

$\phi(2 \cdot 3) = 2$. Streiche aus $\{1, 2, 3, 4, 5, 6\}$ die durch 2 teilbaren Zahlen und die durch 3 teilbaren Zahlen, die echt kleiner als 6 sind (das sind jeweils disjunkte Mengen); streiche schließlich die Zahl 6.

Nach demselben Prinzip erkennt man für zwei *verschiedene* Primzahlen $p$ und $q$

$$\phi(p \cdot q) = p \cdot q - \underbrace{(q-1)}_{\text{durch } p \text{ teilbar}} - \underbrace{(p-1)}_{\text{durch } q \text{ teilbar}} - \underbrace{1}_{\text{Zahl } p \cdot q} = (p-1) \cdot (q-1).$$

# Theorem von Euler

## Theorem von Euler

Seien $x$ und $n$ zwei teilerfremde natürliche Zahlen, dann ist $x^{\phi(n)} \equiv 1 \textbf{ mod } n$.

Sei $A = \{a_1, a_2, \ldots, a_{\phi(n)}\}$ die Menge aller positiven Zahlen kleiner oder gleich $n$, die mit $n$ keinen gemeinsamen Teiler haben. Davon gibt es $\phi(n)$ Stück.

Multiplikation mit $x$ ist eine bijektive Abbildung,
da durch $x$ dividiert werden kann (es gibt eine Inverse!).

Die Menge $x \cdot A = \{x \cdot a_1, x \cdot a_2, \ldots, x \cdot a_{\phi(n)}\}$ ist also, modulo $n$ betrachtet, genau dieselbe Menge wie $A$.

Das Produkt aller Elemente in $A$ und aller Element in $x \cdot A$ ist also, modulo $n$ betrachtet, dieselbe Zahl: Also ist $x \cdot a_1 \cdot x \cdot a_2 \ldots x \cdot a_{\phi(n)} = a_1 \cdot a_2 \ldots a_{\phi(n)}$

Wir können die mit $n$ teilerfremden Zahlen $a_j$ alle abdividieren und erhalten:
$x^{\phi(n)} \equiv 1 \textbf{ mod } n$.

## 3. RSA-Algorithmus

Das RSA-Verfahren in seiner Textbuch-Variante (die so in der Praxis nicht genutzt wird) kennenlernen.

1. Grundbegriffe

2. Voraussetzungen

3. RSA-Algorithmus

4. Facetten und Grenzen

# Historie und Namensgebung



**Abb. 9:** Ron **R**ivest Rechte s. Anhang.



**Abb. 10:** Adi **S**hamir Rechte s. Anhang.



**Abb. 11:** Leonard **A**dleman Rechte s. Anhang.

Das **RSA** Verfahren ist das erste öffentlich bekannte asymmetrische Verschlüsselungsverfahren. 2002 wurde an die drei Erfinder, **R**ivest, **S**hamir und **A**dleman der Turing Preis verliehen, der oft als "Nobelpreis der Informatik" bezeichnet wird.

# RSA-Verfahren

**Vorgehensweise:**

1. Wähle zwei große Primzahlen $p$ und $q$.
2. Berechne $n = p \cdot q$
3. Dann ist $\phi(n) = (p-1) \cdot (q-1)$.
4. Wähle eine Zahl $e$, die teilfremd ist zu $\phi(n)$.
5. Bestimme das Inverse $d$ zu $e$ modulo $\phi(n)$: $e \cdot d = 1$ **mod** $\phi(n)$.
6. Nutze die RSA-Operation $R_z(x) := x^z$ **mod** $n$.

## RSA-Haupteigenschaft

Es gilt: $R_d(R_e(x)) = R_e(R_d(x)) \equiv x$ **mod** $n$.

**Beweis:**

$(x^e)^d = x^{e \cdot d} = x^{1+k \cdot \phi(n)} = x \cdot (x^{\phi(n)})^k = x \cdot 1^k = x.$

# Sicherheit des RSA-Verfahrens: (1) Faktorisierungs-Problem

**Definition: Faktorisierungs-Problem**

Gegeben sei eine Zahl $n$. Finde die Primzahl-Zerlegung von $n$.

Man **vermutet**, daß das Faktorisierungs-Problem für ein $n$, das Produkt zweier großer Primzahlen ist, sehr schwer zu lösen ist.

Man **weiß**, daß die Aufgabe, zur Zahl $e$ die Inverse $d$ modulo $n$ zu bestimmen, gleich schwer ist wie das Faktorisierungs-Problem.

**Jeder Person** sind zugeordnet:

1. Ein **öffentlicher Schlüssel** $(n, e)$, bei dem jeder weiß, daß er dieser Person zugeordnet ist.
2. Ein **privater Schlüssel** $d$, den nur die Person selber kennt und den sie geheim hält.

# Sicherheit des RSA-Verfahrens: (2) RSA-Problem

**Definition: RSA-Problem**

Gegeben seien $y$, $n$ und $e$. Finde ein $x$ so, daß $y = x^e$ **mod** $n$.

**Beachte:**

- Das RSA-Problem ist das Ziehen der $e$-ten Wurzel **mod** $n$: $\sqrt[e]{x^e} = x$.
- $\sqrt[e]{\ }$ modulo $n$ ist verschieden von der bekannten $\sqrt[e]{\ }$ im Reellen!
- Man **vermutet**, daß das RSA-Problem für ein $n$, welches das Produkt zweier großer Primzahlen ist, sehr schwer zu lösen ist.
- Wer ein $d$ kennt mit $d \cdot e \equiv 1$ **mod** $\phi(n)$ kann das Problem sehr schnell lösen: Nach der RSA-Haupteigenschaft ist die Lösung durch $x = y^d$ **mod** $n$.
- Wer die Faktorisierung $n = p \cdot q$ kennt, der kennt auch $\phi(n)$ und kann sich daher zu dem $e$ das $d$ berechnen.

# RSA-Operation als Einweg-Funktion mit Falltüre

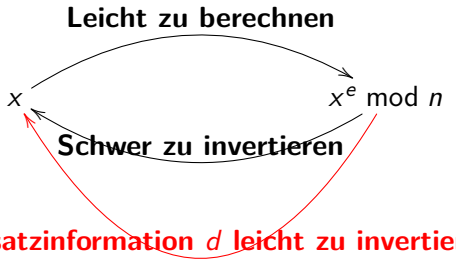



**Abb. 12:** RSA-Operation Einweg-Funktion mit Falltüre. Rechte s. Anhang.

# Sicherheit des RSA-Verfahrens: Praktische Bewertung

**Kerckhoffsches Prinzip:** Jeder weiß, wie das RSA Verfahren funktioniert.

**Sicherheit** hängt an der Schwierigkeit der Lösung zweier mathematischer Probleme:

1. Faktorisierungs-Problem
2. RSA-Problem

Wer für eines der Probleme einen effizienten Algorithmus findet
wird das voraussichtlich publizieren und wird dann berühmt.

Die Sicherheit des RSA-Verfahrens ist daher *auch* eine soziale,
weil man (derzeit) (noch nicht) die genaue Komplexität der Faktorisierung kennt.

Bei vielen anderen Verfahren beruht die Sicherheit auf

- Bekannter Komplexität des Problems — Beispiel: NP-vollständige Probleme.
- Statistischer Sicherheit — Beispiel: One Time Pad.

# Erzeugung der Schlüssel: Mathematische Probleme

**Problem 1:** Erzeuge zwei große Primzahlen $p$ und $q$.
Wie prüfe ich auf Primzahl, wenn das Faktorisierungsproblem kompliziert ist?
**Lösung:** Probabilistische Primzahl-Tests.
Diese können effizient gestaltet werden, sind aber mit einem Zufallsfehler behaftet.
Pro Runde verringert sich Fehlerwahrscheinlichkeit um einen festen Faktor.
Nach einer gewissen Anzahl Runden ist die Wahrscheinlichkeit für die Anwendung klein genug.

**Problem 2:** Finde eine Zahl $e$, die zu $\phi(n)$ teilerfremd ist.
Wie teste ich auf Teilerfremdheit, wenn Faktorisierung schwierig ist?
**Lösung:** Zufallszahl erzeugen
Mit Euklidischem Algorithmus den ggT zu $\phi(n)$ berechnen.
Der Euklidische Algorithmus ist schnell!

**Problem 3:** Bestimme die Inverse $d$ zu $e$.
**Lösung:** Euklidischer Algorithmus verwenden

# Erzeugen der Schlüssel: Organisatorische Probleme

**Variante 1:** Von **Vertrauensperson** erzeugen lassen.

- **Problem:** Vertrauensperson kennt den privaten Schlüssel.

**Variante 2:** **Selber** Erzeugen.

- **Problem 1:** Muß Know-How und ggf. Programm dafür haben.
- **Problem 2:** Wie prüfe ich das Programm auf Korrektheit?
  Stammt es vom Angreifer und sendet ihm den privaten Schlüssel?
- **Problem 3:** Wählt es Schlüssel nicht beliebig sondern in kleinerer Menge,
  die ein Angreifer kennt und mit brute force durchprobieren kann?

**Variante 3:** In **Hardware** erzeugen lassen.

- **Vorteil:** Privater Schlüssel bleibt in Hardware, nicht auslesbar oder verlierbar.
- **Problem:** Muß der Hardware vertrauen.
  Wählt sie den Schlüssel beliebig? Problem wie oben!

**Auswahl** je nach individuellem Angreifer-Modell.

# Nutzung als Verschlüsselung

**Vorgehen:**

1. Alice möchte Bob die Nachricht $x$ verschlüsselt zusenden.
2. Alice weiß: Bob nutzt das RSA-Verfahren mit öffentlichem Schlüssel $(n, e)$.
3. Alice bestimmt $R_e(x) = x^e$ **mod** $n$ und sendet diese Zahl an Bob.
4. Bob empfängt von Alice die Zahl $y$ & holt privaten Schlüssel $d$ aus dem Safe.
5. Bob berechnet $R_d(y) = y^d$.
6. Das ist nach der RSA-Haupteigenschaft aber gerade $x$.

**Kerneigenschaften der Verschlüsselung:**

- **Jeder** kann an Bob **verschlüsseln**.
- Bob **kann entschlüsseln**, also $x$ bestimmen, dank seiner Kenntnis von $d$.
- **Nur** Bob kann entschlüsseln, da das RSA-Problem schwer zu lösen ist.

# Nutzung als Unterschrift

**Kerneigenschaften der Unterschrift:**

- **Nur Bob kann** als Bob unterschreiben.
- **Jeder kann** die Unterschrift von Bob prüfen.

**Vorgehen:**

1. Bob möchte die Nachricht $x$ unterschreiben.
2. Bob holt seinen privaten Schlüssel $d$ aus dem Safe.
3. Bob berechnet $y := R_d(x) = x^d$ **mod** $n$ und veröffentlicht den Wert.
4. Jeder kann prüfen, daß Bob die Nachricht $x$ unterschrieben hat.
5. Dazu besorgt man sich den öffentlichen Schlüssel $(n, e)$ von Bob...
6. und prüft ob $x = R_e(y)$ **mod** $n$.

# Testfrage

Eignet sich diese Zahl als $n$ für das RSA Verfahren:

184769970321174147430683562020016440301854933866341017147178577491065169671116124985933768430543574458561606154457179405222971773252466096064694607124962372044202269756756687378427562389508764678440933285157496578843415088475528298186726451339863364931908084671990431874381283363502795470282653297802934916155811881049844908319545009848393775227257052578591944993870073695755688436933812779613089230392569695253261620823676490316036551371447914939234 7169566988069

- Ja
- Nein
- Kann man nicht so ohne weiteres sagen

Warum?

https://twbk.de/2va2

# Testfrage

Eignet sich diese Zahl als $n$ für das RSA Verfahren:

12409872345098274598172340987435029874301987234098
12340245273459173403485734582340583745982734982544
34982734928374928357394857294857293847293847293847
34958734958723948729483749583745983745983745983745
23498734982734982734982734928749238479283472983479
49587349583745983745983745983745983745983745983753
22947294729473874834783742938798798479879879879878
23498374598374598347598347598347593847593847593847
23498734598734598374598374598374598347575757575755

- Ja
- Nein
- Kann man nicht so ohne weiteres sagen

Warum?

https://twbk.de/2va2

## 4. Facetten und Grenzen

4.1. Homomorphie und RSA-Praxis

4.2. Blinde Unterschrift

Wir lernen verschiedene weitere Facetten des RSA-Verfahrens und mit ihnen auch die Grenzen der Textbuch-Variante kennen.

1. Grundbegriffe

2. Voraussetzungen

3. RSA-Algorithmus

4. Facetten und Grenzen

# Was ist eine große ganze Zahl?

25195908475657893494027183240048398571429282126204032027777137836043662020707
59555626401852588078440691829064124951508218929855914917618450280848912007284
49926873928072877767359714183472702618963750149718246911650776133798590957000
97330459748808428401797429100642458691817195118746121515172654632282216869987
54918242243363725908514186546204357679842338718477444792073993423658482382428
11981638150106748104516603773060562016196762561338441436038339044149526344321
90114657544454178424020924616515723350778707749817125772467962926386356373289
91215483143816789988504044536402352738195137863656439121201039712282212072035
7

**Tab. 1: Achtung:** Wer diese Zahl in Primfaktoren zerlegen kann, konnte 200.000$ bei den RSA Laboratories für den RSA-2048 Challenge gewinnen.
Weitere Quellen dazu: Faktorisierungs-Challenges, Wikipedia über Faktorisierungs-Challenges, Bericht über Faktorisierungs-Challenges. Rechte s. Anhang.

# Beweiskraft einer Unterschrift

Bob findet eine große ganze Zahl $s$.

Bob besorgt sich den public key $(e, n)$ von Alice.

Bob stellt fest, daß $s^e$ **mod** $n$ die Zahl $x = 736772327673696669326679 66$ ergibt.

$x$ ist die Nachricht `ICH LIEBE BOB`. Ascii 73 ist das I. Ascii 67 ist das C usw.

**Bob weiß nun:** Alice hat ihren private key verwendet,
um die Nachricht $x$ digital zu unterschreiben.
(oder Alice hat ihren privaten Schlüssel verloren oder weitergegeben)
(oder Alice hat diesen privaten Schlüssel von seinem Erzeuger erhalten).

Auf andere Weise kann die Nachricht $s$ nicht entstanden sein.

$s$ kann **nicht zufällig** erzeugt worden sein (im Sinne eines statistischen "kann").

Niemand anderes kann den selben privaten Schlüssel wie Alice erzeugt haben.

Die Existenz einer digitalen Unterschrift ist ein extrem starkes Indiz,
daß der Inhaber des privaten Schlüssels den Text unterschrieben hat
(oder ein Fehler bei der Behandlung des Schlüssels passiert ist)

# Beweiskraft der Existenz einer großen Zahl (1)

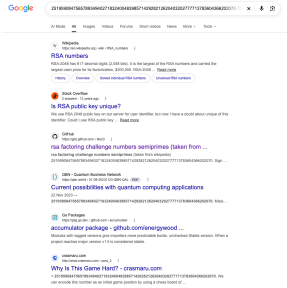



**Abb. 13:** Wir prüfen das anhand der Zahl von zuerst und googlen nach:
251959084756578934940271832400483985714292821262040320277771378360436620207075955562640185258807844069182906412495150821892985591491761845028084891200728449926873928072877767359714183472702618963750149718246911650776133798590957000973304597488084284017974291006424586918171951187461215151726546322822168699875491824224336372590851418654620435767984233871847744479207399342365848238242811981638150106748104516603773060562016196762561338441436038339044149526344321901146575444541784240209246165157233507787077498171257724679629268363563732899121548314381678998850404453640235273819513786365643912120103971228221207203570 Rechte s. Anhang.

# Beweiskraft der Existenz einer großen Zahl (2)

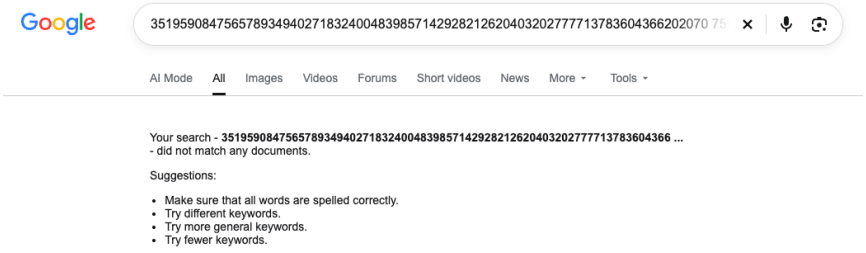



**Abb. 14:** Jetzt googlen wir nach einer ganz leicht veränderten Zahl:
3519590847565789349402718324004839857142928212620403202777713783604366202070
7595556264018525880784406918290641249515082189298559149176184502808489120072
8449926873928072877767359714183472702618963750149718246911650776133798590957
0009733045974880842840179742910064245869181719511874612151517265463228221686
9987549182422433637259085141865462043576798423387184774447920739934236584823
8242811981638150106748104516603773060562016196762561338441436038339044149526
3443219011465754445417842402092461651572335077870774981712577246796292638635
6373289912154831438167899885040445364023527381951378636564391212010397122822
120720357 Rechte s. Anhang.

# Ein weiteres Problem

Bob findet die Zahl $a^d$.

Also hat die Besitzerin von $d$ die Nachricht $a$ unterschrieben.

Bob findet die Zahl $b^d$.

Also hat die Besitzerin von $d$ die Nachricht $b$ unterschrieben.

Bob bildet das Produkt $p$ der beiden Zahlen: $p = a^d \cdot b^d$.

Bob stellt fest, daß $p = (a \cdot b)^d$.

Überrascht fragt sich Bob, ob die Besitzerin von $d$
auch die Nachricht $a \cdot b$ unterschrieben hat.

Vermutlich nicht!

**Fazit:** Houston, we have a problem!

# Homomorphie-Eigenschaft

**Die RSA-Operation ist ein multiplikativer Homomorphismus**

$$R_d(a \cdot b) = R_d(a) \cdot R_d(b)$$

**Angriff**

Hole Unterschrift auf Dokument $a$ und $b$ ein, die beide harmlose Verträge sind.
Bekomme dadurch Unterschrift auf $a \cdot b$, das eine große Schenkung darstellt.

**Frage 1:** Wie verhindert man Angriffe, die sich daraus ergeben?

**Frage 2:** Kann diese Eigenschaft noch anderweitig genutzt werden?

# Voraussetzungen für einen Homomorphie-Angriff

1. $a$ muß ein Dokument sein, welches das Opfer unterschreibt.
2. $b$ muß ein Dokument sein, welches das Opfer unterschreibt.
3. $a \cdot b$ muß ein Dokument sein, auf dem der Angreifer eine Unterschrift haben will.

**Problem:** $a \cdot b$ ist meistens gar kein natürlichsprachliches Dokument mehr.

**Einwand:** Angreifer kann 10 Jahre lang nach solchen Dokumenten gesucht haben.

# Wir wollen genauer wissen!

Wir probieren dazu aus: `Hallo` = 72 97 108 108 111 Du = 100 117

7297108108111 × 100117 = 7305645724597489887 (laut `math-gpt.org`, nicht überprüft)

Aufteilen in Zweiergruppen: 73 05 64 57 24 59 74 89 88 7

| | |
|---|---|
| 73 | I |
| 05 | nicht druckbares Steuerzeichen `ENQ` |
| 64 | |
| 57 | 9 |
| 24 | nicht druckbares Steuerzeichen `CAN` |
| 59 | ; |
| 74 | J |
| 89 | Y |
| 88 | X |
| 7 | nicht druckbares Steuerzeichen `BEL` |

# Schutz vor Homomorphie-Eigenschaft

**Ansatz 1:** Verändern.
Bevor man einen Text $a$ unterschreibt, ändert man diesen ein wenig ab in $a'$.
Das zerstört einen möglichen, mittels $a$ vorbereiteten Angriff,
da $a' \cdot b$ kein sinnvolles Dokument mehr ist.

**Ansatz 2:** Randomisieren.
Bevor man einen Text $a$ unterschreibt,
erweitert man diesen links und rechts um Zufallszahlen.
Das ist eine Erweiterung von Ansatz 1.

**Ansatz 3:** Hash Funktion
Man unterschreibt ohnehin nie ein Dokument
sondern nur einen Fingerabdruck (Hash) des Dokuments.
Denn: Dokumente sind meistens länger als der Modul $n$.
Die Hash-Funktion wirkt ähnlich wie (und besser als!) die Ansätze 1 und 2.

# Textbook-RSA

Gegen unsere Version von RSA gibt es eine Vielzahl sehr effizienter Angriffe.

**Gegen die Implementierungen:**

1. Timing Attacke
2. Power Attack
3. Differential Power Attack
4. Accustic Emanation Attacke
5. Belcore Attacke
6. Randomization Attack

**Gegen das Konzept:**

1. Wiener Attacke
2. Coppersmith Attacke
3. ROCA Attacke
4. Chosen Plaintext Attacke
5. Chosen Ciphertext Attacke
6. Adaptive Chosen Ciphertext (Bleichenbacher) Attack
7. Non strong prime number Attacke

Daher wird RSA **hoffentlich nie** in der Textbuch-Variante eingesetzt.

# Seitenkanal-Angriffe

## Seitenkanal-Angriffe

Seitenkanal-Angriffe nutzen Informationen aus der Implementierung eines Verfahrens zur Kompromittierung von Schutzzielen.

# Power Attacke als Beispiel für Seitenkanal-Angriff

**Beobachtung:** Anzahl der 1-en auf dem Bus beeinflußt den Stromverbrauch der CPU.

**Power Attacke** auf RSA-Hardware:

- Messe den Stromverbrauch des Systems während der Verarbeitung.
- Erfasse Ein- und Ausgaben des Systems.
- Bestimme daraus den privaten Schlüssel.

**Spezifische Verteidigung:**

- Zwischenbuffern der Stromversorgung: Hält meßbaren Verbrauch konstant.

Relative alte Attacke: [MDS99], [KJJ99]

Übersichtsartikel: [RD20]

# Praktische Nutzung von RSA

**Leitsatz**

In der Kryptographie sollen stets nur professionelle Implementierungen von ausgewiesenen Experten verwendet werden.

**Konkrete** Aspekte bei RSA

- Seitenkanal-Sicherheit — Durch geeignete Implementierung.
- Randomisierung nutzen.
- Padding nutzen.
- Signatur: Hash signieren statt Dokument
- Verschlüsselung: Hybride Verschlüsselung
  RSA für den Austausch eines symmetrischen Schlüssel.

# Blinde Unterschrift

Alice hat ein Dokument $x$ und braucht darauf eine digitale Unterschrift $s$ von Bob.

Bob soll das Dokument $x$ nicht mit Alice in Verbindung bringen können.

**Anwendungsbeispiel:** Anonymes elektronisches Geld / eCash – Vorläufer von Bitcoin.

Parteien: Alice, Bank Bob und Händler.

# Blinde Signatur nach David Chaum

**Situation:** Bob hat public key $(n, e)$ und private key $d$.

**Aufgabe:** Alice will blinde Unterschrift von Bob auf Dokument $x$.

**Vorgehensweise:**

1. Alice wählt Blinding Factor $B$ teilerfremd zu $n$.
2. Alice sendet $y = x \cdot R_e(B)$ **mod** $n$ an Bob.
3. Bob unterschreibt, sendet also $z = R_d(y)$ an Alice.
4. Alice berechnet multiplikative Inverse $G$ zu $B$.
5. Alice berechnet $z \cdot G$.

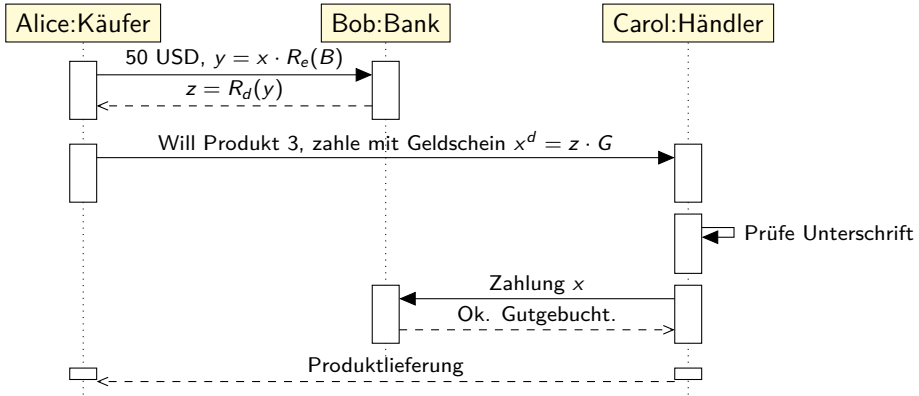
# Korrektheit der Blinden Signatur

**Signatur:** Alice erhält damit eine Signatur von Bob auf $x$.

$$
\begin{aligned}
& z \cdot G && \text{Definition von } z \\
& \equiv y^d \cdot G && \text{Definition von } y \\
& \equiv (x \cdot R_e(B))^d \cdot G = && \text{Definition von } R_e \\
& \equiv (x \cdot B^e)^d \cdot G = && \text{Eigenschaft des Potenzierens} \\
& \equiv x^d \cdot B^{e \cdot d} \cdot G = && \text{RSA-Haupteigenschaft} \\
& \equiv x^d \cdot B \cdot G = && \text{Definition von G} \\
& x^d \ \textbf{mod}\ n
\end{aligned}
$$

**Blindheit:** Bob kann sich $y$ merken, aber daraus nicht auf $x$ schließen.
Denn: Multiplikation mit $R_e(B)$ erlaubt Alice eine *beliebige* Transformation von $x$ innerhalb der Menge aller zu $n$ teilerfremden Werte.

# Digitales Geld: Modell eCash von David Chaum



# Appendix

# Übersicht


| | | |
|---|---|---|
| Übersicht | | 56 |
| Literaturverzeichnis | | 57 |
| Verzeichnis aller Abbildungen | Abb | 58 |
| Verzeichnis aller Tabellen | Tab | 60 |
| Rechtsnachweise | © | 61 |
| Rechtliche Hinweise | § | 63 |
| Zitierweise dieses Dokuments | → | 65 |
| Verzeichnis aller Folien | | 66 |

# Literaturverzeichnis


[KJJ99] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. "Differential Power Analysis". In: *Advances in Cryptology - CRYPTO '99*. Vol. 1666. Lecture Notes in Computer Science. Springer, 1999, pp. 388–397. DOI: 10.1007/3-540-48405-1_25 (cit. on p. 49).

[MDS99] Thomas S. Messerges, Ezzy A. Dabbish, and Robert H. Sloan. "Power Analysis Attacks of Modular Exponentiation in Smartcards". In: *Cryptographic Hardware and Embedded Systems (CHES 1999)*. Vol. 1717. Lecture Notes in Computer Science. Springer, 1999, pp. 144–157. DOI: 10.1007/3-540-48059-5_14 (cit. on p. 49).

[RD20] Mark Randolph and William Diehl. "Power Side-Channel Attack Analysis: A Review of 20 Years of Study for the Layman". In: *Cryptography* 4.2 (2020), p. 15. DOI: 10.3390/cryptography4020015 (cit. on p. 49).

# Verzeichnis aller Abbildungen (1/2)


1 Prinzip der Einweg-Funktion . . . . . 4
2 Beispiel für eine Einweg-Funktion im allgemeinen Fall. . . . . 5
3 Einweg-Funktion: Gegenbeispiel . . . . . 5
4 RSA-785 Primzahl-Zerlegung . . . . . 6
5 Komplexität von Primzahlprodukten und Zerlegungen . . . . . 9
6 Prinzip der Einweg-Funktion mit Falltüre. . . . . 10
7 Euklid von Alexandrien . . . . . 14
8 Etienne Bezout . . . . . 20
9 Ron Rivest . . . . . 25

10 Adi Shamir . . . . . 25
11 Leonard Adleman . . . . . 25
12 RSA-Operation Einweg-Funktion mit Falltüre . . . . . 29
13 Eine besondere Zahl . . . . . 40
14 Keine besondere Zahl . . . . . 41

# Verzeichnis aller Tabellen

1 RSA-2048 Challenge . . . . . 38

# Rechtsnachweise (1/2)

Abb. 1 Eigene Abbildung

Abb. 2 Eigene Abbildung

Abb. 3 Eigene Abbildung

Abb. 4 Eigene Abbildung

Abb. 5 Eigene Abbildung

Abb. 6 Eigene Abbildung

Abb. 7 Quelle: Wikipedia. Nutzung im public domain.

Abb. 8 Quelle: Wikipedia. Nutzung im public domain.

Abb. 9 Quelle: https://upload.wikimedia.org/wikipedia/commons/7/79/Ronald_L_Rivest_photo.jpg, Ronald L. Rivest, Nutzung nach CC BY-SA 4.0 https://creativecommons.org/licenses/by-sa/4.0.

Abb. 10 Quelle: https://upload.wikimedia.org/wikipedia/commons/9/9a/Adi_Shamir_Royal_Society.jpg, The Royal Society, Nutzung nach CC BY-SA 3.0 https://creativecommons.org/licenses/by-sa/3.0.

Abb. 11 Quelle: https://upload.wikimedia.org/wikipedia/commons/a/af/Len-mankin-pic.jpg, len adlmen, Nutzung nach CC BY-SA 3.0 https://creativecommons.org/licenses/by-sa/3.0.

Abb. 12 Eigene Abbildung

Tab. 1 Quellen in der Table Caption benannt.

Abb. 13 Eigener Screenshot

Abb. 14 Eigener Screenshot

# Rechtliche Hinweise (1)

Die hier angebotenen Inhalte unterliegen deutschem Urheberrecht. Inhalte Dritter werden unter Nennung der Rechtsgrundlage ihrer Nutzung und der geltenden Lizenzbestimmungen hier angeführt. Auf das Literaturverzeichnis wird verwiesen. Das Zitatrecht in dem für wissenschaftliche Werke üblichen Ausmaß wird beansprucht. Wenn Sie eine Urheberrechtsverletzung erkennen, so bitten wir um Hinweis an den auf der Titelseite genannten Autor und werden entsprechende Inhalte sofort entfernen oder fehlende Rechtsnennungen nachholen. Bei Produkt- und Firmennamen können Markenrechte Dritter bestehen. Verweise und Verlinkungen wurden zum Zeitpunkt des Setzens der Verweise überprüft; sie dienen der Information des Lesers. Der Autor macht sich die Inhalte, auch in der Form, wie sie zum Zeitpunkt des Setzens des Verweises vorlagen, nicht zu eigen und kann diese nicht laufend auf Veränderungen überprüfen.
Alle sonstigen, hier nicht angeführten Inhalte unterliegen dem Copyright des Autors, Prof. Dr. Clemens Cap, ©2020. Wenn Sie diese Inhalte nützlich finden, können Sie darauf verlinken oder sie zitieren. Jede weitere Verbreitung, Speicherung, Vervielfältigung oder sonstige Verwertung außerhalb der Grenzen des Urheberrechts bedarf der schriftlichen Zustimmung des Rechteinhabers. Dieses dient der Sicherung der Aktualität der Inhalte und soll dem Autor auch die Einhaltung urheberrechtlicher Einschränkungen wie beispielsweise Par 60a UrhG ermöglichen.
Die Bereitstellung der Inhalte erfolgt hier zur persönlichen Information des Lesers. Eine Haftung für mittelbare oder unmittelbare Schäden wird im maximal rechtlich zulässigen Ausmaß ausgeschlossen, mit Ausnahme von Vorsatz und grober Fahrlässigkeit. Eine Garantie für den Fortbestand dieses Informationsangebots wird nicht gegeben.
Die Anfertigung einer persönlichen Sicherungskopie für die private, nicht gewerbliche und nicht öffentliche Nutzung ist zulässig, sofern sie nicht von einer offensichtlich rechtswidrig hergestellten oder zugänglich gemachten Vorlage stammt.
**Use of Logos and Trademark Symbols:** The logos and trademark symbols used here are the property of their respective owners. The YouTube logo is used according to brand request 2-9753000030769 granted on November 30, 2020. The GitHub logo is property of GitHub Inc. and is used in accordance to the GitHub logo usage conditions https://github.com/logos to link to a GitHub account. The Tweedback logo is property of Tweedback GmbH and here is used in accordance to a cooperation contract.

# Rechtliche Hinweise (2)

**Disclaimer:** Die sich immer wieder ändernde Rechtslage für digitale Urheberrechte erzeugt ein nicht unerhebliches Risiko bei der Einbindung von Materialien, deren Status nicht oder nur mit unverhältnismäßig hohem Aufwand abzuklären ist. Ebenso kann den Rechteinhabern nicht auf sinnvolle oder einfache Weise ein Honorar zukommen, obwohl deren Leistungen genutzt werden.
Daher binde ich gelegentlich Inhalte nur als Link und nicht durch Framing ein. Lt EuGH Urteil 13.02.2014, C-466/12 (Pressemitteilung, Blog-Beitrag, Urteilstext). ist das unbedenklich, da die benutzten Links ohne Umgehung technischer Sperren auf im Internet frei verfügbare Inhalte verweisen.
Wenn Sie diese Rechtslage stört, dann setzen Sie sich für eine Modernisierung des völlig veralteten Vergütungs- und Anreizsystems für urheberrechtliche Leistungen ein. Bis dahin klicken Sie bitte auf die angegebenen Links und denken Sie darüber nach, warum wir keine für das digitale Zeitalter sinnvoll angepaßte Vergütungs- und Anreizsysteme digital erbrachter Leistungen haben.
Zu Risiken und Nebenwirkungen fragen Sie Ihren Rechtsanwalt oder Gesetzgeber.
Weitere Hinweise finden Sie im Netz hier und hier oder hier.

# Zitierweise dieses Dokuments

Wenn Sie Inhalte aus diesem Werk nutzen oder darauf verweisen wollen,
zitieren Sie es bitte wie folgt:

Clemens H. Cap: Verschlüsselung. Electronic document, 17. 11. 2026 https://iuk.one/DS-ENC.pdf

**Bibtex Information:** https://iuk.one/DS-ENC.bib

```
@misc{doc:DS-ENC,
   author        = {Clemens H. Cap},
   title         = {Verschlüsselung},
   year          = {2026},
   month         = {11},
   howpublished  = {Electronic document},
   url           = {https://iuk.one/DS-ENC.pdf }
}
```

**Typographic Information:**

Typeset on 2026-11-17 at 18:17
This is pdfTeX, Version 3.141592653-2.6-1.40.29 (TeX Live 2026) kpathsea version 6.4.2
This is pgf in version 3.1.11a
This is preamble-slides.tex ©C.H.Cap
Jobtype flag=foreground
Active modes (if any): slides, biblio
Keywords: keywords
Subject: subject

# Verzeichnis aller Folien


Titelseite . . . 1
Übersicht . . . 2

**1. Grundbegriffe . . . 3**

Motivation: Einweg-Funktion . . . 4
Beispiel: Einweg-Funktion . . . 5
Begriff der Typikalität . . . 6
Testfrage . . . 7
Wichtige Punkte . . . 8
Was bedeutet schwierig genau? . . . 9
Motivation: Einweg-Funktion mit Falltüre . . . 10
Anwendung . . . 11

**2. Voraussetzungen . . . 12**

**2.1. Modulo-Rechnung**
Modulo Rechnung . . . 13
**2.2. Euklidischer Algorithmus**
Euklidischer Algorithmus: Eine Wiederholung an einem Beispiel . . . 14
Stimmt der Algorithmus? . . . 15
Euklidischer Algorithmus: Termination . . . 16
Euklidischer Algorithmus: Teiler-Eigenschaft . . . 17
Euklidischer Algorithmus: Größte-Eigenschaft . . . 18
**2.3. Bezoutsche Identität**
Bezoutsche Identität: Aussage und Algorithmus . . . 19
Bezoutsche Identität: Beispiel . . . 20
**2.4. Modulo-Division**
Modulo-Division . . . 21
**2.5. Eulersche Phi-Funktion**
Eulersche Phi-Funktion . . . 22
**2.6. Theorem von Euler**
Theorem von Euler . . . 23

**3. RSA-Algorithmus . . . 24**

Historie und Namensgebung . . . 25
RSA-Verfahren . . . 26
Sicherheit des RSA-Verfahrens: (1) Faktorisierungs-Problem . . . 27
Sicherheit des RSA-Verfahrens: (2) RSA-Problem . . . 28
RSA-Operation als Einweg-Funktion mit Falltüre . . . 29
Sicherheit des RSA-Verfahrens: Praktische Bewertung . . . 30
Erzeugung der Schlüssel: Mathematische Probleme . . . 31
Erzeugen der Schlüssel: Organisatorische Probleme . . . 32
Nutzung als Verschlüsselung . . . 33
Nutzung als Unterschrift . . . 34
Testfrage . . . 35
Testfrage . . . 36

**4. Facetten und Grenzen ..... 37**
Was ist eine große ganze Zahl? ..... 38
Beweiskraft einer Unterschrift ..... 39
Beweiskraft der Existenz einer großen Zahl (1) ..... 40
Beweiskraft der Existenz einer großen Zahl (2) ..... 41
Ein weiteres Problem ..... 42
Homomorphie-Eigenschaft ..... 43
Voraussetzungen für einen Homomorphie-Angriff ..... 44
Wir wollen genauer wissen! ..... 45
**4.1. Homomorphie und RSA-Praxis**
Schutz vor Homomorphie-Eigenschaft ..... 46
Textbook-RSA ..... 47
Seitenkanal-Angriffe ..... 48
Power Attacke als Beispiel für Seitenkanal-Angriff ..... 49
Praktische Nutzung von RSA ..... 50
**4.2. Blinde Unterschrift**
Blinde Unterschrift ..... 51
Blinde Signatur nach David Chaum ..... 52
Korrektheit der Blinden Signatur ..... 53
Digitales Geld: Modell eCash von David Chaum ..... 54

**Legende:**
Fortsetzungsseite
Seite ohne Überschrift
Bildseite