Das RSA-Kryptosystem

«In this age of communications that span both distance and time, the only tool we have that approximates a ‹whisper› is encryption. When I cannot whisper in my wife's ear or the ears of my business partners, and have to communicate electronically, then encryption is our tool to keep our secrets secret.» [1]

Abb. 1: John McAfee (1945–2021) [3]

Inhalt

1 Bemerkungen für Lehrpersonen

Diese Lerneinheit hat, vor Allem in Abschnitt 8, einen hohen mathematischen Anspruch. Sie richtet sich an Klassen, deren Schülerinnen und Schüler mathematisch besonders interessiert sind; sie kann auch im Ergänzungsfach Informatik eingesetzt werden. In allen Klassen ist sie einsetzbar, sofern der Abschnitt 8 ausgelassen wird.

Voraussetzungen:

- Die Schülerinnen und Schüler sollten mit der modularen Rechnung vertraut sein.
  - Es sollte kein Problem sein, Modulo als Operation oder als Relation zu sehen.
  - Die grundlegenden Rechenregeln der modularen Rechnung sollten bekannt sein.
  - Es sollte bekannt sein, dass man beim modularen Rechnen mod n die Zahlen immer klein (unter n) halten kann.
- Wünschenswert sind Grundkenntnisse aus der Zahlentheorie wie:
  - Was ist eine Primzahl?
  - Primfaktorzerlegung
  - Grösster gemeinsamer Teiler
  - Euklidischer Algorithmus

  vorhanden sein.
- Die Schülerinnen und Schüler sollten in der Lage sein, einfache Programme für das Ausprobieren und Überprüfen zu schreiben. In dieser Lerneinheit wurde die Programmiersprache Python verwendet. Der Inhalt orientiert sich aber nicht an einer spezifischen Sprache.
- Die Schülerinnen und Schüler, die sich an Übung 11* versuchen, sollten in der Lage sein, rekursive Funktionen zu programmieren.
- Es sollte bekannt sein, wie ein asymmetrisches Verschlüsselungsverfahren (Public/Private key) grundsätzlich funktioniert.

Ziele: Die Schülerinnen und Schüler sollten nach dem Studium dieser Lerneinheit

- verstanden haben, wie das RSA-Kryptosystem funktioniert,
- ein (simples, im Sinne von kleinen Primzahlen) RSA-Schlüsselpaar generieren können,
- eine Nachricht mit dem Verfahren ver- und entschlüsseln können,
- einen Einblick in die darunterliegende Zahlentheorie erhalten haben und
- weitere Sicherheit im Umgang mit Restklassen erhalten haben.

Didaktische Überlegungen: Es wurde versucht, das Thema in die Richtung «selbst entdecken» zu bringen. Dies ist hier aber grundsätzlich nicht komplett möglich, da RSA dafür zu komplex ist. Trotzdem soll nicht einfach gleich das Verfahren vorgegeben werden. Mit einigen Übungen sollen die Schülerinnen und Schüler an das Verfahren herangeführt werden, so dass es nicht komplett «aus dem Himmel fällt». Dies hat zur Konsequenz, dass an einigen Stellen die Methode «Ausprobieren», zum Beispiel um einen privaten Schlüssel zu finden, angewandt wird. Es ist wichtig, den Schülerinnen und Schülern gegenüber zu betonen, dass dies nur funktioniert, weil die Zahlen in den Beispielen klein genug sind.

Lösungen liegen zusätzlich in Videoform vor — mit Ausnahme von Übung 6 —, die den Schülerinnen und Schülern nach und nach angeboten werden können.

Kommentare zu einzelnen Teilen:

Übung 1

- Hier soll das modulare Rechnen minimal aufgefrischt werden. Dies kann mit einem Computer geschehen.
- Die Übung zielt natürlich darauf ab, dass die Schülerinnen und Schüler erkennen, dass $(k^5)^{29} \equiv k \mod 91$.
- Aufgrund der Komplexität von RSA wird hier mit konkreten Zahlen gestartet, dies in der Hoffnung, dass sich die Schülerinnen und Schüler bereits jetzt an die Grundidee von RSA gewöhnen.

- Das Überprüfen der Vermutung mittels eines kleinen Programms soll insbesondere den Umgang mit Modulo in der entsprechenden Programmiersprache repetieren.
- Da das Resultat von allgemeinem Interesse ist, ist es wichtig, die Vermutung kurz zu besprechen und festzuhalten.

Übung 2

- Hier soll durch Ausprobieren ein zweites gültiges Schlüsselpaar gefunden werden.
- Um die Situation nicht zu komplex zu gestalten und um langsam auf die allgemeine Situation vorzubereiten, wird hier am Modulus $n = 91$ festgehalten.

Übung 3

- Diese Übung dient dazu, etwas Vertrautheit mit der Eulerschen φ-Funktion zu erhalten.
- Es ist zu erwarten, dass die Schülerinnen und Schüler hier die Lösungen durch Abzählen erhalten. Stärkeren Schülerinnen oder Schülern kann hier die Frage gestellt werden, wie man die einzelnen Resultate vielleicht ohne Abzählen hätte finden können.

Übung 4

- Dies ist eine Vorbereitung auf das allgemeine Verfahren, das dort natürlich $\varphi(p \cdot q)$ eine Rolle spielt.
- Auch $\varphi(p)$ findet in einem späteren Beweis Anwendung.
- Beide Resultate sollten im Unterricht festgehalten und betont werden.

Übung 5

- Hier wird nochmals versucht, eine wichtige Eigenschaft im allgemeinen RSA-Verfahren zu finden.
- Für die Übung kann das Resultat von Übung 4 benutzt werden.
- Im Unterricht sollte die zu entdeckende Vermutung, dass bei gültigen Tripel $e \cdot d \equiv 1 \mod n$ gilt, festgehalten und besprochen werden.

Übung 6

- Hier soll das RSA-Verfahren mit selbst ausgewählten Zahlen einmal durchgespielt werden.
- Dies soll dazu dienen, dass die Schülerinnen und Schüler mit dem Verfahren vertrauter werden.
- Da die Lösung hier stark von der Auswahl der Zahlen der Schülerinnen und Schüler abhängt, gibt es auch keine Musterlösung.

Übung 7

- Da hier n klein ist, lässt sich p und q leicht bestimmen. Damit lässt sich der private Schlüssel finden.
- Die Motivation besteht hier darin, eine Geheimschrift zu knacken.

Übung 8 Diese Übung dient dazu, dass die Schülerinnen und Schüler den erweiterten euklidischen Algorithmus in einem sehr einfachen Fall anwenden, ohne dass ihnen dieser schon gezeigt wurde.

Übung 9

- Hier sollen sich die Schülerinnen und Schüler mit dem erweiterten euklidischen Algorithmus vertraut machen.
- In der Theorie wird bewusst darauf verzichtet, den erweiterten euklidischen Algorithmus präzise oder gar iterativ anzugeben. Aus den Beispielen sollte klar werden, wie dieser funktioniert. So soll das Verständnis dafür, was dabei passiert, erhöht werden.
- In Teilaufgabe (c) ist es die Absicht erfassbar zu machen, dass diese Methode auch in anderen Fällen anwendbar sein kann. Dies ist nicht direkt relevant für das RSA-Verfahen. Sie soll zu dem Verständnis des erweiterten euklidschen Algorithmus beitragen.

Übung 10

- Diese Übung ist im Prinzip eine Wiederholung von Übung 7.
- Der Unterschied liegt darin, dass nun kein Rechner mehr zugelassen ist.
- Die Schülerinnen und Schüler sollen das RSA-Verfahren nochmals durchspielen und dieses Mal an keiner Stelle mehr auf die Methode «Ausprobieren» ausweichen.

- Insbesondere soll nochmals der erweiterte euklidische Algorithmus angewandt werden.

Übung 11*

- Diese Übung ist optional, da sie etwas grössere Programmierkenntnisse voraussetzt. Es ist denkbar, diese versierteren Schülerinnen oder Schülern zu überlassen.
- Es wird für diese Übung vorausgesetzt, dass rekursive Funktionen bekannt sind.
- Es ist sinnvoll, die Schülerinnen und Schüler den Algorithmus rekursiv implementieren zu lassen.
- Grundsätzlich ist damit zu rechnen, dass die Schülerinnen und Schüler hier Unterstützung benötigen.

Übung 12

- Die Idee ist es, hier etwas Vertrautheit mit dem Satz von Euler-Fermat zu erhalten.
- Da $\varphi(7) = 6$, kann hier einfach $5^2 \mod 7$ berechnet werden.

Übung 13 Dies ist eine anspruchsvollere, aber analoge Übung zu Übung 12.

Abschnitt 8

- Für diesen Abschnitt 8 ist eine umfangreiche Bearbeitungszeit einzuplanen.
- Gerade der Beweis des Satzes von Euler-Fermat dürfte die Schülerinnen und Schüler stark fordern.
- Man sollte hier situativ abwägen, ob es sich lohnt, den Beweis zu behandeln oder nicht.
- Wenn man sich für den Beweis entscheidet, sollte dieser von Hand neu aufgeschrieben werden. Mit entsprechenden Erklärungen dürfte es verständlicher sein.
- Die Beweise der beiden Sätze wurden bewusst auch in Videoform festgehalten, damit die Schülerinnen und Schüler der Argumentation in ihrem eigenen Tempo nochmals folgen können.
- Um den Beweis etwas leichter zugänglich zu machen, wird dieser erst mit Beispielzahlen durchgeführt. So soll die Idee des Beweises klargemacht werden. Eine Zwischenvariante wäre es, nur diese Beweisskizze zu verwenden.
- Investiert man genügend Zeit und erklärt den Beweis erst anhand von konkreten Zahlen, ist er erfahrungsgemäss machbar.

2 Modulares Rechnen: Operation oder Relation

Es gibt zwei unterschiedliche Arten mit modularem Rechnen umzugehen.

Die eine wird beim Programmieren häufiger verwendet und ist als *Operation* zu verstehen. Schreiben wir

$$17 \mod 4 = 1$$

meinen wir, dass beim Teilen von 17 durch 4 ein Rest von 1 übrig bleibt.

Die zweite Art, welche in der Mathematik üblicher ist, ist als *Relation* zwischen zwei ganzen Zahlen zu verstehen. Schreiben wir

$$17 \equiv 37 \mod 4$$

meinen wir, dass $37 - 17$ durch 4 teilbar ist. Gleichbedeutend ist dazu auch $17 \mod 4 = 37 \mod 4$.

Aus praktischen Gründen bevorzugen wir in diesem Kapitel die zweite Art. Kommt aber eine solche Modulo Rechnung in einem Algorithmus vor, nutzen wir die erste Art.

Grundsätzlich sind die beiden Arten austauschbar.

3 Einführung

Das *RSA-Kryptosystem* ist eines der weltweit meisten eingesetzen asymmetrischen Verschlüsselungsverfahren. Der Name «RSA» ist eine Abkürzung für die Nachnamen der drei Mathematiker: Ronald Rivest, Adi Shamir und Leonard Adleman. Dies sind die Erfinder dieses Systems.

Abb. 2: Ronald Rivest (∗1947) [7]

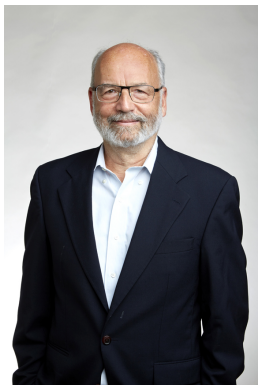

Abb. 3: Adi Shamir (∗1952) [2]

Abb. 4: Leonhard Adleman (∗1945) [4]

Dieses Kryptosystem ist komplex. Daher ist es schwierig nachzuvollziehen, wie man darauf kommt, warum es funktioniert und warum es als sicher gilt.

Wir werden aufgrund der Komplexität nicht alle diese Fragen im Detail klären können.

4 Ein Prototyp des Verfahrens

Übung 1 *(Lösung auf Seite 16)*

Berechnen Sie die folgenden Restklassen:

(a) $2^5 \mod 91$

(b) $32^{29} \mod 91$

(c) $12^5 \mod 91$

(d) $38^{29} \mod 91$

(e) $(33^5)^{29} \mod 91$

(f) $70^{5 \cdot 29} \mod 91$

Was fällt Ihnen auf? Stellen Sie eine Vermutung auf und überprüfen Sie diese mit einem kleinen Programm.

Wir können diese Beobachtung nun nutzen, um eine (Teil-)Nachricht (d.h. hier eine natürliche Zahl von 0 bis 90) asymmetrisch zu verschlüsseln. Wir benutzen als öffentlichen Schlüssel die Zahl $e = 5$ (e steht für «encrypt») und als privaten Schlüssel die Zahl $d = 29$ (d steht für «decrypt»). Aus einer Nachricht m kann nämlich ein Absender mit Hilfe des öffentlichen Schlüssels 5 (und dem *Modulus* $n = 91$) den Chiffre-Text

$$c = m^5 \mod 91$$

bestimmen. Der Empfänger kann dann mit Hilfe des privaten Schlüssels $d = 29$ (und dem Modulus $n = 91$) die Klartextnachricht wieder entschlüsseln, indem dieser

$$m = c^{29} \mod 91$$

berechnet.

Dieses Verfahren ist ein Spezialfall des *RSA-Verfahrens*.

Natürlich macht es wenig Sinn, immer die gleichen Zahlen $e = 5$, $d = 29$ und $n = 91$ zu verwenden.

Übung 2 *(Lösung auf Seite 16)*

Wir ändern jetzt e auf $e = 31$. Bestimmen Sie mit Hilfe eines kleinen Programms durch Ausprobieren eine natürliche Zahl d, so dass das oben beschriebene Verfahren mit $n = 91$ auch funktioniert.

Die Methode «Ausprobieren» von Übung 2 funktioniert nur, wenn der Modulus n klein ist, mit vertretbarem Rechenaufwand. Wird das RSA-Verfahren in der Praxis angewandt, ist n viel zu gross, als dass diese Methode erfolgversprechend wäre. Dies ist auch gut so, denn sonst wäre es möglich, aus dem öffentlichen Schlüssel (e, n) auf diese Weise den privaten Schlüssel (d, n) herausfinden.

Wir müssen also klären, welche mathematischen Eigenschaften so ein gültiges Schlüsselpaar, also ein Tripel (e, d, n) haben muss, damit es für das RSA-Verfahren funktioniert. Kennen wir diese Eigenschaften, können wir auch leichter eine Methode finden, um ein gültiges Schlüsselpaar zu erzeugen.

5 Die Eulersche φ-Funktion

Die sogenannte *Eulersche-φ-Funktion* ist für das RSA-Kryptosystem von grosser Bedeutung. Diese wird auf jeden Fall nötig sein, um zu verstehen, warum das Verfahren funktioniert.

Definition: Eulersche φ-Funktion

Für eine natürliche Zahl n definieren wir $\varphi(n)$ als die Anzahl natürlicher Zahlen $k \leqslant n$, die teilerfremd zu n sind.

Abb. 5: Leonhard Euler (1707–1783) [5]

Beispiele

(a) Es gilt $\varphi(5) = 4$, da alle Zahlen 1, 2, 3 und 4 teilerfremd zu und kleiner als 5 sind.

(b) Es gilt $\varphi(8) = 4$, da genau die Zahlen 1, 3, 5 und 7 teilerfremd zu und kleiner als 8 sind.

Übung 3 *(Lösung auf Seite 16)*
Berechnen Sie die folgenden Ausdrücke:

(a) $\varphi(1)$ (b) $\varphi(7)$ (c) $\varphi(10)$ (d) $\varphi(32)$

Übung 4 *(Lösung auf Seite 17)*
Es seien p und q zwei *verschiedene* Primzahlen.

(a) Begründen Sie, dass gilt: $\varphi(p) = p - 1$

(b) Begründen Sie, dass gilt: $\varphi(p \cdot q) = (p - 1) \cdot (q - 1)$

Übung 5 *(Lösung auf Seite 17)*
Berechnen Sie jeweils

$$e \cdot d \mod \varphi(n)$$

für die folgenden Werte von e, d und n.

(a) $e = 5$, $d = 29$, $n = 91$ (c) $e = 8$, $d = 37$, $n = 91$ (e) $e = 13$, $d = 37$, $n = 55$

(b) $e = 31$, $d = 7$, $n = 91$ (d) $e = 15$, $d = 22$, $n = 91$ (f) $e = 10$, $d = 19$, $n = 55$

Welche Tripel entsprechen gültigen Schlüsselpaaren? Überprüfen Sie dies jeweils mit einem kleinen Programm.
Fällt Ihnen etwas auf?

6 Wie funktioniert das Verfahren konkret?

Wir fassen nun die obigen Beobachtungen zusammen und formulieren ein Rezept, um festzuhalten, wie das RSA-Kryptosystem funktioniert.

Rezept: RSA-Verfahren

Schlüsselpaar generieren

1. Wählen Sie zwei verschiedene Primzahlen p und q.
2. Berechnen Sie den Modulus $n = p \cdot q$.
3. Berechnen Sie $\varphi(n) = (p-1) \cdot (q-1)$.
4. Wählen Sie eine natürliche Zahl e, die teilerfremd zu $\varphi(n)$ ist.
5. Der öffentliche Schlüssel ist nun das Paar (e, n).
6. Finden Sie eine natürliche Zahl d, so dass gilt:

$$d \cdot e \equiv 1 \mod \varphi(n)$$

7. Der private Schlüssel ist nun das Paar (d, n).

Verschlüsseln einer Nachricht Aus der Klartextnachricht m wird der Chiffre-Text

$$c = m^e \mod n$$

erzeugt.

Entschlüsseln einer Nachricht Aus dem Chiffre-Text c erhält man die Klartextnachricht m durch

$$m = c^d \mod n.$$

Bemerkung

- Wie bei jedem asymmetrischen Verschlüsselungsverfahren ist es absolut wichtig, dass der private Schlüssel nur dem Empfänger und niemand anderem zur Verfügung steht. Neben der Zahl d wäre es auch fatal, p, q oder $\varphi(n)$ zu veröffentlichen, denn damit könnte der private Schlüssel leicht berechnet werden.
- Da wir sowohl beim Verschlüsseln als auch beim Entschlüsseln «mod n» rechnen, muss die Klartextnachricht einer Zahl von 0 bis $n-1$ entsprechen. Ist die zu verschlüsselnde Nachricht länger, muss sie in entsprechende Teile gespalten werden.
- In der Praxis sind p und q sehr grosse Primzahlen. Dies macht es aufwändig p und q aus n zurückzugewinnen. Dies ist der Grund, warum das RSA-Verfahren aufwändig zu knacken ist.

  Für das Faktorisieren einer grossen Zahl (n) in Primfaktoren kein effizienter Algorithmus (für klassische Computer) bekannt. Es wird auch vermutet, dass es keinen solchen gibt.

  Für Quantencomputer gibt es jedoch theoretisch einen effizienten Algorithmus - den sogenannten Shor-Algorithmus. Wir sind aber bei Weitem nicht in der Lage einen genügend grossen Quantencomputer bauen zu können. Deswegen gilt das RSA-Verfahren (mit genügend grossen Primzahlen p und q) als sicher.
- In dieser Lerneinheit verwenden wir oft die Methode «Ausprobieren» um beispielsweise d zu bestimmen. Dies ist nur möglich, weil wir sehr kleine Werte für p und q verwenden. Mit solchen kleinen Zahlen ist das Verfahren nicht sicher. Es erlaubt uns jedoch besser «hinter die Kulissen» zu schauen. Sind die Primzahlen in einer praxisnahen Grössenordnung, hat $\varphi(n)$ mindestens ein paar tausend Stellen. Es müssten also extrem viele Zahlen durchprobiert werden. Dies ist bei so vielen Kandidaten hoffnungslos. Die Grösse der gewählten Primzahlen ist also ein entscheidender Faktor für die Sicherheit!
- Beim Verschlüsseln und Entschlüsseln entstehen vermeintlich riesige Potenzen (m^e bzw. c^d). Eine geschickte Implementation verhindert hier aber explodierende Zahlen, da bei jeder Multiplikation «mod n» gerechnet werden kann.
- Schritt 1 ist in der Praxis alles andere als trivial. Denn hier müssen zwei verschiedene (mindestens mehrere hundert Stellen) grosse Primzahlen zufällig gewählt werden. Es ist nicht klar, wie dass

effizient geht. Es gibt aber Algorithmen, die effizient testen können, ob eine Zahl eine Primzahl ist oder nicht. Genau solche werden hier verwendet, um auch effizient solche Primzahlen zu finden. Die Details würden anspruchsvolle Zahlentheorie und Algebra erfordern und den Rahmen sprengen.[1]

Beispiel

1. Wir wählen $p = 17$ und $q = 41$.
2. Der Modulus lautet $n = 17 \cdot 41 = 697$.
3. Es gilt $\varphi(n) = (17 - 1) \cdot (41 - 1) = 640$.
4. Wir wählen $e = 47$. Dies ist teilerfremd zu 640.
5. Der öffentliche Schlüssel ist nun das Paar (47, 697).
6. Die Zahl d können wir zum Beispiel durch Ausprobieren finden[2]:

```python
def find_d(e,phi):
    for d in range(phi):
        if d*e % phi == 1:
            return d
    return -1

print(find_d(47,640))
```

Quellcode 1: Naives Python-Programm für das Finden von d

Das Programm von Quellcode 1 liefert:

$$d = 463$$

7. Der private Schlüssel ist nun das Paar (463, 697).

Übung 6

Arbeiten Sie für diese Übung in Zweiergruppen.

(a) Generieren Sie beide jeweils ein gültiges Schlüsselpaar und tauschen Sie (nur) die öffentlichen Schlüssel in der Gruppe aus.

(b) Jeder von Ihnen verschlüsselt je eine Nachricht (eine Zahl) mit dem öffentlichen Schlüssel des anderen Gruppenmitglieds. Tauschen Sie die entsprechenden Chiffren-Texte aus.

(c) Entschlüsseln Sie die erhaltene Nachricht mit dem privaten Schlüssel.

Übung 7 *(Lösung auf Seite 17)*

Eine Nachricht wurde mit dem öffentlichen Schlüssel $(e, n) = (23, 55)$ verschlüsselt. Das Resultat war der Chiffre-Text

$$c = 4.$$

Bestimmen Sie die Klartextnachricht m.

7 Der erweiterte euklidische Algorithmus

Wir haben bisher bei Schritt 6 des RSA-Verfahrens (siehe Seite 8) auf die naive Methode «Ausprobieren» zurückgegriffen. Dies ist für solch kleine Werte von n bzw. $\varphi(n)$ gut möglich. Wird n grösser, so wird dies schnell zu aufwändig. Dies ist auch gut so, denn sonst könnte man die ganze Verschlüsselung durch Ausprobieren knacken.

[1] Genaueres findet man zum Beispiel in [9].
[2] Dies ist ineffizient und bei grossen Zahlen hoffnungslos. Mehr dazu in Abschnitt 7.

Folgerichtig ergibt sich an dieser Stelle, dass wir eine Methode finden müssen, um den Schritt anders und effizienter durchzuführen.

Übung 8 *(Lösung auf Seite 18)*
Teilen Sie 49 durch 8 mit Rest und finden Sie damit zwei ganze Zahlen u und v, so dass gilt:

$$u \cdot 49 + v \cdot 8 = 1$$

Mit dem Resultat von Übung 8 können wir schliessen, dass $v \cdot 8 \equiv 1 \mod 49$, und so das gesuchte d finden.

Um die Zahl d bei RSA-Verfahren im Allgemeinen zu finden, können wir ähnlich wie in Übung 8 vorgehen. Allerdings kann es sein, dass wir, wie im euklidischen Algorithmus, mehrere Schritte brauchen, bis wir bei dem grössten gemeinsamen Teiler 1 landen. Man nennt dies den *erweiterten euklidischen Algorithmus* und dieser ist viel effizienter als Ausprobieren. Das Ausprobieren ist für genügend grosse Zahlen hoffnungslos.

Wie dies konkret funktioniert, sollen die folgenden Beispiele zeigen. Versuchen Sie zu erkennen, wie die Methode im Allgemeinen funktioniert.

Beispiele

(a) Wir suchen eine natürliche Zahl d, so dass gilt:

$$d \cdot 7 \mod 5 = 1$$

Wenn wir den euklidischen Algorithmus auf 7 und 5 anwenden, erhalten wir die folgenden Gleichungen[3]

$$7 = 1 \cdot 5 + 2 \quad (1)$$
$$5 = 2 \cdot 2 + \mathbf{1} \quad (2)$$

und damit ggT(7, 5) = 1. Dies wussten wir natürlich schon. Wir können die obigen Gleichungen aber «rückwärts» nutzen. Konkret schliessen wir zunächst aus (2):

$$1 = 5 - 2 \cdot 2$$

Des Weiteren folgt aus (1)

$$2 = 7 - 1 \cdot 5$$

und damit

$$1 = 5 - 2 \cdot (7 - 1 \cdot 5) = 3 \cdot 5 - 2 \cdot 7 \Rightarrow -2 \cdot 7 \equiv 1 - 3 \cdot 5 \equiv 1 \mod 5.$$

Wir finden also $d \equiv -2 \mod 5$ und um eine positive Lösung zu haben, wählen wir

$$d = -2 + 5 = 3.$$

(b) Wir wollen eine natürliche Zahl d finden, so dass gilt:

$$d \cdot 17 \equiv 1 \mod 57$$

Der euklidische Algorithmus für 57 und 17 liefert:

$$57 = 3 \cdot 17 + 6 \quad (3)$$
$$17 = 2 \cdot 6 + 5 \quad (4)$$
$$6 = 1 \cdot 5 + \mathbf{1} \quad (5)$$

Wenn wir diese Gleichungen «rückwärts» nutzen, erhalten wir:

$$1 \overset{(5)}{=} 6 - 1 \cdot 5 \overset{(4)}{=} 6 - 1 \cdot (17 - 2 \cdot 6) = 3 \cdot 6 - 17 \overset{(3)}{=} 3 \cdot (57 - 3 \cdot 17) - 17 = 3 \cdot 57 - 10 \cdot 17$$

Es folgt $-10 \cdot 17 \equiv 1 \mod 57$. Also wählen wir:

$$d = -10 + 57 = 47$$

Sie finden dieses Beispiel auch in Videoform unter folgender Adresse:

[3] Wir lassen hier die letzte Division mit Rest, die offenbar Rest 0 liefert, aus Gründen der Übersichtlichkeit bewusst weg.

oli4math. *Beispiel zum erweiterten euklidischen Algorithmus*. YouTube. 4. Nov. 2022. URL: `https://youtu.be/gaHRgzvHGtY` (besucht am 04. 11. 2022)

Wir können also mit dem euklidischen Algorithmus für zwei natürliche Zahlen a, b zwei ganze Zahlen u, v finden, so dass gilt:

$$u \cdot a + v \cdot b = \text{ggT}(a, b) \tag{6}$$

Man nennt dies den *erweiterten euklidischen Algorithmus*.

Aus (6) folgt natürlich

$$u \cdot a \equiv \text{ggT}(a, b) \mod b \text{ und } v \cdot b \equiv \text{ggT}(a, b) \mod a$$

Übung 9 *(Lösung auf Seite 18)*
Bestimmen Sie ohne Rechner mit Hilfe des erweiterten euklidischen Algorithmus jeweils eine natürliche Zahl d, so dass gilt

(a) $d \cdot 29 \equiv 1 \mod 94$ (b) $d \cdot 12 \equiv 1 \mod 103$ (c) $d \cdot 105 \equiv 7 \mod 182$

Übung 10 *(Lösung auf Seite 19)*
Eine Nachricht wurde mit dem öffentlichen Schlüssel $(e, n) = (35, 221)$ verschlüsselt. Das Resultat war der Chiffre-Text:

$$c = 2$$

Bestimmen Sie die Klartextnachricht m ohne Rechner. Benutzen Sie auch den erweiterten euklidischen Algorithmus.

Hinweis: $221 = 13 \cdot 17$

Übung 11* *(Lösung auf Seite 19)*
Implementieren Sie den erweiterten euklidischen Algorithmus. Schreiben Sie also eine Funktion

`ext_eucl(a,b).`

Diese soll aus zwei natürlichen Zahlen `a,b` eine Liste `[u,v]` liefern, so dass gilt

$$u \cdot a + v \cdot b = \text{ggT}(a, b)$$

Testen Sie ihr Programm mit den Beispielen aus Übung 9.

Hinweis: Implementieren Sie die Funktion rekursiv und nutzen Sie die für den euklidischen Algorithmus zentrale Eigenschaft, dass gilt

$$\text{ggT}(a, b) = \text{ggT}(b, r),$$

wobei r der Rest beim Teilen von a durch b ist.

8 Warum funktioniert das RSA-Verfahren?*

Wir müssen nun noch etwas rigoroser einsehen, dass das Verfahren funktioniert. Um dies zu tun, werden wir tiefer in die Zahlentheorie eintauchen müssen. Dieser Abschnitt ist mathematisch anspruchsvoll.

Konkret müssen wir den folgenden Satz 1 einsehen.

Satz 1: Korrektheit des RSA-Verfahrens

Gegeben seien zwei verschiedene Primzahlen p und q. Es sei $n = p \cdot q$ und e eine zu $\varphi(n)$ teilerfremde Zahl. Des Weiteren sei d eine natürliche Zahl, so dass $e \cdot d \equiv 1 \mod \varphi(n)$. Dann gilt für alle ganzen Zahlen m:

$$(m^e)^d \equiv m \mod n$$

Der Satz besagt im Wesentlichen, dass eine mit dem RSA-Verfahren verschlüsselte Nachricht korrekt wieder entschlüsselt wird.

Das wichtigste Werkzeug, um diesen Satz einzusehen, ist der sogenannte Satz von Euler-Fermat (Satz 2).

Abb. 6: Pierre de Fermat (1607–1665) [6]

Satz 2: Euler-Fermat

Es sei n eine natürliche und m eine zu n teilerfremde ganze Zahl, dann gilt:

$$m^{\varphi(n)} \equiv 1 \mod n$$

Der Beweis dieses Satzes ist der mathematisch anspruchsvollste Teil dieser Lerneinheit und wir werden diesen zunächst als gegeben annehmen.

Übung 12 *(Lösung auf Seite 19)*
Berechnen Sie mit Hilfe des Satzes von Euler-Fermat (Satz 2) und ohne Rechner:

$$5^{6'000'002} \mod 7$$

Übung 13 *(Lösung auf Seite 19)*
Bestimmen Sie mit Hilfe des Satzes von Euler-Fermat (Satz 2) und ohne Rechner die letzte Stelle der Zahl

$$7^{2022}.$$

Beweis von Satz 1. Da $e \cdot d \mod \varphi(n) = 1$, gibt es eine ganze Zahl k, so dass

$$e \cdot d = 1 + k \cdot \varphi(n).$$

Es folgt

$$(m^e)^d = m^{e \cdot d} = m^{1+k \cdot \varphi(n)} = m \cdot m^{k \cdot \varphi(n)}. \tag{7}$$

Falls m und n teilerfremd sind, folgt mit Hilfe von Euler-Fermat (Satz 2) direkt

$$(m^e)^d \equiv m \cdot (m^{\varphi(n)})^k \equiv m \cdot 1^k \equiv m \mod m.$$

Ist m durch q aber nicht durch p teilbar, so sind m und p teilerfremd. Mit Euler-Fermat für p und m

folgt

$$m^{p-1} \equiv m^{\varphi(n)} \equiv 1 \mod p.$$

Es gibt also eine ganze Zahl ℓ, so dass

$$m^{p-1} = 1 + \ell \cdot p.$$

Aus (7) und $\varphi(n) = (p-1) \cdot (q-1)$ folgt damit:

$$(m^e)^d = m \cdot m^{k \cdot (p-1) \cdot (q-1)} = m \cdot (m^{p-1})^{k \cdot (q-1)} = m \cdot (1 + \ell \cdot p)^{k \cdot (q-1)}$$

Wenn wir den Ausdruck $(1 + \ell \cdot p)^{k \cdot (q-1)}$ ausmultiplizieren, erhalten wir

$$1 + p \cdot s$$

für eine ganze Zahl s. Beachten Sie dazu, dass in jedem Summanden, ausser in dem ersten Summanden, der Faktor $\ell \cdot p$ vorkommen muss.

Wir schliessen

$$(m^e)^d = m \cdot (1 + p \cdot s) = m + m \cdot p \cdot s.$$

Da nun m durch q und p offenbar durch p teilbar sind, ist $m \cdot p \cdot s$ durch $n = p \cdot q$ teilbar und somit $m \cdot p \cdot s \equiv 0 \mod n$.

Dies liefert

$$(m^e)^d \equiv m + 0 \equiv m \mod n.$$

Aus Symmetriegründen folgt auch der Fall, wenn m durch q, aber nicht durch p teilbar ist.

Ist m durch n, also durch p und durch q, teilbar, so folgt direkt

$$(m^e)^d \equiv m \equiv 0 \mod n.$$

□

Bemerkung Sie finden diesen Beweis auch in Videoform unter

oli4math. *Korrektheit des RSA-Verfahrens.* YouTube. 28. Okt. 2022. URL: `https://youtu.be/IwoFqnVo6fg` (besucht am 28. 10. 2022)

Wir müssen jetzt noch den Satz von Euler-Fermat (Satz 2) einsehen.

Bevor wir das tun, betrachten wir erstmals den folgenden Hilfssatz. Dieser wird an vielen Stellen im Beweis nützlich sein.

Hilfssatz 1

Es seien drei ganze Zahlen a, b, c und eine natürliche Zahl n gegeben, so dass gilt:

$$a \cdot b \equiv a \cdot c \mod n \text{ und } \mathrm{ggT}(a, n) = 1$$

Dann gilt:

$$b \equiv c \mod n \text{ und } \mathrm{ggT}(a, n) = 1$$

Wir können in diesem Fall also quasi «durch a dividieren.»

Beweis. Aus $a \cdot b \equiv a \cdot c \mod n$ folgt, dass es eine ganze Zahl k gibt, so dass

$$a \cdot b = a \cdot c + k \cdot n \;\Rightarrow\; k \cdot n = a \cdot b - a \cdot c = a \cdot (b - c). \tag{8}$$

Wir schliessen, dass $k \cdot n$ durch a teilbar sein muss. Da $\mathrm{ggT}(a, n) = 1$ folgt, dass a ein Teiler von k ist.

Wir können also

$$k = k' \cdot a$$

für eine ganze Zahl k' schreiben.

Aus (8) folgt jetzt

$$a \cdot b = a \cdot c + k' \cdot a \cdot n.$$

Dividieren wir noch durch a, liefert dies $b = c + k' \cdot n$ und damit

$$b \equiv c \mod n.$$

□

Um eine Idee darüber zu gewinnen, wie der Beweis von Satz 2 funktioniert, führen wir diesen erst an einem Beispiel durch.

Beweisidee von Satz 2: Wir schauen uns die Beweisidee anhand des Beispiels $n = 8$ und $m = 5$ an.

Zunächst einmal sind

$$1, 3, 5, 7 \tag{9}$$

alle natürlichen Zahlen, die kleiner als und teilerfremd zu 8 sind. Dies liefert uns: $\varphi(8) = 4$.

Nun multiplizieren wir alle Zahlen von (9) mit $m = 5$. Dies liefert uns

$$m \cdot 1 = 5,\ m \cdot 3 = 15,\ m \cdot 5 = 25,\ m \cdot 7 = 35. \tag{10}$$

Modulo $n = 8$ erhalten wir

$$5, 7, 1, 3,$$

also genau die *gleichen* Zahlen wie in (9). Diese erscheinen nur in einer anderen Reihenfolge.

Multiplizieren wir alle Zahlen von (9) bzw. alle Zahlen von (10), müssen wir mod n das Gleiche erhalten. Wir schliessen:

$$1 \cdot 3 \cdot 5 \cdot 7 \equiv (m \cdot 1) \cdot (m \cdot 3) \cdot (m \cdot 5) \cdot (m \cdot 7) \equiv m^{\varphi(n)} \cdot 1 \cdot 3 \cdot 5 \cdot 7 \mod n$$

Da $1 \cdot 3 \cdot 5 \cdot 7$ und $n = 8$ sicher teilerfremd sind, erhalten wir mit Hilfssatz 1

$$1 \equiv m^{\varphi(n)}.$$

□

Wir folgen nun der obigen groben Beweisidee, um einen vollständigen Beweis zu erhalten.

Beweis von Satz 2: Seien

$$a_1, a_2, \ldots, a_{\varphi(n)} \tag{11}$$

alle natürlichen Zahlen, die kleiner als und teilerfremd zu n sind.

Multiplizieren wir alle Zahlen von (11) mit m, erhalten wir:

$$m \cdot a_1, m \cdot a_2, \ldots, m \cdot a_{\varphi(n)} \tag{12}$$

Diese Zahlen sind alle teilerfremd zu n, da $\mathrm{ggT}(m, n) = \mathrm{ggT}(a_i, n) = 1$.

Es sind sogar alle Zahlen in der gleichen Restklasse mod n teilerfremd zu n, da sich diese nur um Vielfache von n unterscheiden.

Die entsprechenden Restklassen mod n sind auch alle verschieden, denn wäre

$$m \cdot a_i \equiv m \cdot a_j \mod n,$$

so würde mit Hilfssatz 1 folgen (beachten Sie, dass $\mathrm{ggT}(n, m) = 1$): $a_i \equiv a_j \mod n$ und damit $a_i = a_j$. Beachten Sie dazu, dass $0 \leqslant a_i, a_j < n$.

Es folgt, dass die Restklassen mod n der Zahlen von (11) und die Restklassen mod n der Zahlen von (12) die gleichen sein müssen. Die Reihenfolge kann dabei möglicherweise unterschiedlich sein.

Nun multiplizieren wir alle Restklassen mod n der Zahlen von (11) bzw. von (12). Dies liefert uns:

$$a_1 \cdot a_2 \cdots a_{\varphi(n)} \equiv m^{\varphi(n)} \cdot a_1 \cdot a_2 \cdots a_{\varphi(n)} \mod n$$

Da $a_1 \cdot a_2 \cdots a_{\varphi(n)}$ teilerfremd zu n sein muss, schliessen wir somit mit Hilfssatz 1:

$$1 \equiv m^{\varphi(n)} \mod n$$

□

Bemerkung Sie finden obigen Beweis auch in Videoform unter

oli4math. *Euler-Fermat: Beweis*. YouTube. 29. Okt. 2022. URL: `https://youtu.be/3YiTduH71bQ` (besucht am 29. 10. 2022)

9 Lösungen zu den Übungen

Lösung von Übung 1:

(a) 32 (c) 38 (e) 33

(b) 2 (d) 12 (f) 70

Wir vermuten, dass für alle Zahlen $k = 0, \ldots, 90$ gilt:

$$(k^5)^{29} \equiv k \mod 91$$

Diese Vermutung ist richtig, was das Programm von Quellcode 2 bestätigen kann.

```
for k in range(91):
    if (k**5)**29 % 91 != k:
        print("Vermutung ist für k="+str(k)+" falsch")
        quit()
print("Vermutung für alle k=0,...,90 korrekt")
```

Quellcode 2: Python-Programm für die Kontrolle

oli4math. *Modulares Rechnen: Vermutung über gewisse Potenzen*. YouTube. 30. Okt. 2022. URL: https://youtu.be/WFXWNH46_5Y (besucht am 30. 10. 2022)

Lösung von Übung 2:

```
def check_d(d):
    for m in range(91):
        if (m**31)**d % 91 !=m:
            return False
    return True

for d in range(2,91):
    if check_d(d):
        print("d="+str(d)+" funktioniert")
        quit()

print("Keine Lösung gefunden")
```

Quellcode 3: Python-Programm für das Finden von d

Das Programm von Quellcode 3 liefert uns: $d = 7$

oli4math. *RSA Verfahren: Exponent d durch Ausprobieren finden*. YouTube. 30. Okt. 2022. URL: https://youtu.be/fHU7n8CVJyI (besucht am 30. 10. 2022)

Lösung von Übung 3:

(a) $\varphi(1) = 1$ (b) $\varphi(7) = 6$ (c) $\varphi(10) = 4$ (d) $\varphi(32) = 16$

oli4math. *Zahlentheorie: Beispiele zur Eulerschen phi-Funktion*. YouTube. 30. Okt. 2022. URL: https://youtu.be/nbU_hw_7ZQw (besucht am 30. 10. 2022)

Lösung von Übung 4:

(a) Da p eine Primzahl ist, sind die Zahlen 1, 2, 3, ..., $p-1$ alle teilerfremd zu p. Wir erhalten

$$\varphi(p) = p - 1.$$

(b) Von den Zahlen 1, 2, ..., $p \cdot q$ sind die q Zahlen

$$p,\ 2 \cdot p,\ 3 \cdot p,\ \ldots, q \cdot p$$

durch p und die p Zahlen

$$q,\ 2 \cdot q,\ 3 \cdot q,\ \ldots, p \cdot q$$

durch q teilbar.

Da in diesen Listen die Zahl $p \cdot q$ zweimal vorkommt, erhalten wir:

$$\varphi(p \cdot q) = p \cdot q - q - p + 1 = p \cdot (q-1) - q + 1 = (p-1) \cdot (q-1)$$

oli4math. *Zahlentheorie: Eulersche Phi-Funktion: phi(p) und phi(p*q).* YouTube. 30. Okt. 2022. URL: `https://youtu.be/n18Z1EDU7NA` (besucht am 30. 10. 2022)

Lösung von Übung 5:

(a) $5 \cdot 29 \mod 72 = 1$

(b) $31 \cdot 7 \mod 72 = 1$

(c) $8 \cdot 37 \mod 72 = 8$

(d) $15 \cdot 22 \mod 72 = 42$

(e) $13 \cdot 37 \mod 40 = 1$

(f) $10 \cdot 19 \mod 40 = 30$

```python
def check_triple(e,d,n):
    for m in range(n):
        if (m**e)**d % n !=m:
            print("Tripel e="+str(e)+", d="+str(d)+", n="+str(n)+"
            ↪ ist ungültig")
            return False
    print("Tripel e="+str(e)+", d="+str(d)+", n="+str(n)+" ist
    ↪ gültig")
    return True

check_triple(5,29,91)
check_triple(31,7,91)
check_triple(8,37,91)
check_triple(15,22,91)
check_triple(13,37,55)
check_triple(10,19,55)
```

Quellcode 4: Python-Programm für das Überprüfen der Tripel

Das Programm von Quellcode 4 zeigt, dass die Tripel von den Teilen (a), (b) und (e) gültige Tripel sind.

Es fällt auf, dass genau in diesen Teilen gilt, $e \cdot d \mod \varphi(n) = 1$.

oli4math. *RSA: Was ist e*d mod phi(n)?* YouTube. 30. Okt. 2022. URL: `https://youtu.be/WkdSa93nweg` (besucht am 30. 10. 2022)

Lösung von Übung 7: Wir faktorisieren erstmals den Modulus $n = 5 \cdot 11$. Es folgt $\varphi(n) = (5-1) \cdot (11-1) = 40$.

Durch Ausprobieren finden wir $d = 7$, da $23 \cdot 7 \mod 40 = 1$. Jetzt berechnen wir

$$m = 5^7 \mod 55 = 25.$$

oli4math. *RSA Verfahren: Klartext zurückgewinnen*. YouTube. 30. Okt. 2022. URL: https://youtu.be/c_7bwBuGRFI (besucht am 30. 10. 2022)

Lösung von Übung 8: Es gilt:

$$49 = 6 \cdot 8 + 1 \;\Rightarrow\; 1 = 1 \cdot 49 - 6 \cdot 8$$

Wir erhalten so: $u = 1$ und $v = -6$

oli4math. *Teilen mit Rest: Beispiel*. YouTube. 4. Nov. 2022. URL: https://youtu.be/zmUdhEgNcjo (besucht am 04. 11. 2022)

Lösung von Übung 9:

(a) Der euklidische Algorithmus für 94 und 29 liefert:

$$94 = 3 \cdot 29 + 7 \tag{13}$$
$$29 = 4 \cdot 7 + 1 \tag{14}$$

Es folgt:

$$1 \overset{(14)}{=} 29 - 4 \cdot 7 \overset{(13)}{=} 29 - 4 \cdot (94 - 3 \cdot 29) = 13 \cdot \cdot 29 - 4 \cdot 94 \;\Rightarrow\; 13 \cdot 29 \equiv 1 \mod 94$$

Wir wählen also $d = 13$.

(b) Der euklidische Algorithmus für 103 und 12 liefert:

$$103 = 8 \cdot 12 + 7 \tag{15}$$
$$12 = 1 \cdot 7 + 5 \tag{16}$$
$$7 = 1 \cdot 5 + 2 \tag{17}$$
$$5 = 2 \cdot 2 + 1 \tag{18}$$

Es folgt:

$$1 \overset{(18)}{=} 5 - 2 \cdot 2 \overset{(17)}{=} 5 - 2 \cdot (7 - 1 \cdot 5) = 3 \cdot 5 - 2 \cdot 7 \overset{(16)}{=} 3 \cdot (12 - 1 \cdot 7) - 2 \cdot 7 = 3 \cdot 12 - 5 \cdot 7$$
$$\overset{(15)}{=} 3 \cdot 12 - 5 \cdot (103 - 8 \cdot 12) = 43 \cdot 12 - 5 \cdot 103 \;\Rightarrow\; 43 \cdot 12 \equiv 1 \mod 103$$

Wir wählen also $d = 43$.

(c) Der euklidische Algorithmus für 182 und 105 liefert:

$$182 = 1 \cdot 105 + 77 \tag{19}$$
$$105 = 1 \cdot 77 + 28 \tag{20}$$
$$77 = 2 \cdot 28 + 21 \tag{21}$$
$$28 = 1 \cdot 21 + 7 \tag{22}$$

Es folgt:

$$7 \overset{(22)}{=} 28 - 1 \cdot 21 \overset{(21)}{=} 28 - 1 \cdot (77 - 2 \cdot 28) = 3 \cdot 28 - 77 \overset{(20)}{=} 3 \cdot (105 - 1 \cdot 77) - 77$$
$$= 3 \cdot 105 - 4 \cdot 77 \overset{(19)}{=} 3 \cdot 105 - 4 \cdot (182 - 1 \cdot 105) = 7 \cdot 105 - 4 \cdot 182 \;\Rightarrow\; 7 \cdot 105 \equiv 7 \mod 182$$

Wir wählen also $d = 7$.

oli4math. *Erweiterter euklidischer Algorithmus: Beispiele*. YouTube. 4. Nov. 2022. URL: https://youtu.be/tNJDwZOh_ws (besucht am 04. 11. 2022)

Lösung von Übung 10: Aus der Faktorisierung des Hinweises folgt $\varphi(n) = (13-1)\cdot(17-1) = 192$.

Jetzt müssen wir eine natürliche Zahl d finden, so dass gilt $d \cdot 35 \equiv 1 \mod 192$. Wir benutzen dazu den erweiterten euklidischen Algorithmus für 35 und 192:

$$192 = 5 \cdot 35 + 17 \tag{23}$$
$$35 = 2 \cdot 17 + 1 \tag{24}$$

Es folgt:

$$1 \overset{(24)}{=} 35 - 2\cdot 17 \overset{(23)}{=} 35 - 2\cdot(192 - 5\cdot 35) = 11\cdot 35 - 2\cdot 192 \;\Rightarrow\; 11\cdot 35 \equiv 1 \mod 192$$

Wir finden also $d = 11$. Die Klartextnachricht m lautet nun wie folgt:

$$m \equiv 2^{11} \equiv 2048 \equiv 59 \mod 221$$

Es gilt also $m = 59$.

oli4math. *RSA-Verfahren: Knacken ohne Rechner*. YouTube. 4. Nov. 2022. URL: https://youtu.be/WcsdCPI5AbM (besucht am 04. 11. 2022)

Lösung von Übung 11:*

```python
def ext_eucl(a,b):
    r = a%b
    if r==0:
        return [0,1]
    q=a//b
    [u,v]=ext_eucl(b,r)
    return [v,u-v*q]

def test_eucl(a,b):
    [u,v]=ext_eucl(a,b)
    print("u="+str(u)+", v="+str(v))
    print("ggT("+str(a)+","+str(b)+")="+str(u*a+v*b))

test_eucl(29,94)
test_eucl(12,103)
test_eucl(105,182)
```

Quellcode 5: Rekursive Implementation des erweiterten euklidischen Algorithmus

oli4math. *Implementation des erweiterten euklidischen Algorithmus*. YouTube. 4. Nov. 2022. URL: https://youtu.be/Ly1e41MVGXk (besucht am 04. 11. 2022)

Lösung von Übung 12: Es gilt $\varphi(7) = 6$ und damit folgt mit Euler-Fermat

$$5^{6'000'002} \equiv 5^2 \cdot (5^6)^{1'000'000} \equiv 25 \cdot 1 \equiv 4 \mod 7.$$

oli4math. *Zahlentheorie: Modulo Rechnung mit dem Satz von Euler-Fermat*. YouTube. 30. Okt. 2022. URL: https://youtu.be/cZEXCSDMqkY (besucht am 30. 10. 2022)

Lösung von Übung 13: Es gilt $\varphi(10) = (2-1)\cdot(5-1) = 4$. Damit folgt mit Hilfe von Euler-Fermat:

$$7^{2022} \equiv 7^2 \cdot 7^{4\cdot 505} \equiv 49 \cdot (7^4)^{505} \equiv 9 \cdot 1 \equiv 9 \mod 10$$

Die letzte Ziffer lautet also 9.

oli4math. *Zahlentheorie: Letzte Stelle mit Euler-Fermat bestimmen.* YouTube. 30. Okt. 2022. URL: https://youtu.be/O-JfmkMT2bg (besucht am 30. 10. 2022)

Quellenverzeichnis

[1] BrainyQuote. *John McAfee Quotes*. URL: https://www.brainyquote.com/quotes/john_mcafee_755503 (besucht am 27. 10. 2022).

[2] Wikimedia Commons. *Adi Shamir*. Creative Commons Attribution-Share Alike 3.0 Unported. URL: https://commons.wikimedia.org/wiki/File:Adi_Shamir_Royal_Society.jpg (besucht am 25. 10. 2022).

[3] Wikimedia Commons. *John McAfee*. Creative Commons Attribution-Share Alike 3.0 Unported. URL: https://commons.wikimedia.org/wiki/File:John_McAfee_by_Gage_Skidmore.jpg (besucht am 27. 10. 2022).

[4] Wikimedia Commons. *Leonhard Adleman*. Creative Commons Attribution-Share Alike 3.0 Unported. URL: https://en.wikipedia.org/wiki/File:Len-mankin-pic.jpg (besucht am 25. 10. 2022).

[5] Wikimedia Commons. *Leonhard Euler*. public domain. URL: https://commons.wikimedia.org/wiki/File:Leonhard_Euler.jpg (besucht am 07. 07. 2020).

[6] Wikimedia Commons. *Pierre de Fermat*. public domain. URL: https://en.wikipedia.org/wiki/File:Pierre_de_Fermat.jpg (besucht am 10. 08. 2021).

[7] Wikimedia Commons. *Ronald Rivest*. Creative Commons Attribution-Share Alike 4.0 International. URL: https://en.wikipedia.org/wiki/File:Ronald_L_Rivest_photo.jpg (besucht am 25. 10. 2022).

[8] Karin Freiermuth u. a. *Einführung in die Kryptologie. Lehrbuch für Unterricht und Selbsstudium*. Vieweg+Teubner Verlag | Springer Fachmedien Wiesbaden GmbH 2010, 2010.

[9] Juraj Hromkovič. *Randomisierte Algorithmen. Methoden zum Entwurf von zufallsgesteuerten Systemen für Einsteiger*. B.G. Teubner Verlag, 2004.

[10] oli4math. *Beispiel zum erweiterten euklidischen Algorithmus*. YouTube. 4. Nov. 2022. URL: https://youtu.be/gaHRgzvHGtY (besucht am 04. 11. 2022).

[11] oli4math. *Erweiterter euklidischer Algorithmus: Beispiele*. YouTube. 4. Nov. 2022. URL: https://youtu.be/tNJDwZOh_ws (besucht am 04. 11. 2022).

[12] oli4math. *Euler-Fermat: Beweis*. YouTube. 29. Okt. 2022. URL: https://youtu.be/3YiTduH71bQ (besucht am 29. 10. 2022).

[13] oli4math. *Implementation des erweiterten euklidischen Algorithmus*. YouTube. 4. Nov. 2022. URL: https://youtu.be/Ly1e41MVGXk (besucht am 04. 11. 2022).

[14] oli4math. *Korrektheit des RSA-Verfahrens*. YouTube. 28. Okt. 2022. URL: https://youtu.be/IwoFqnVo6fg (besucht am 28. 10. 2022).

[15] oli4math. *Modulares Rechnen: Vermutung über gewisse Potenzen*. YouTube. 30. Okt. 2022. URL: https://youtu.be/WFXWNH46_5Y (besucht am 30. 10. 2022).

[16] oli4math. *RSA Verfahren: Exponent d durch Ausprobieren finden*. YouTube. 30. Okt. 2022. URL: https://youtu.be/fHU7n8CVJyI (besucht am 30. 10. 2022).

[17] oli4math. *RSA Verfahren: Klartext zurückgewinnen*. YouTube. 30. Okt. 2022. URL: https://youtu.be/c_7bwBuGRFI (besucht am 30. 10. 2022).

[18] oli4math. *RSA-Verfahren: Knacken ohne Rechner*. YouTube. 4. Nov. 2022. URL: https://youtu.be/WcsdCPI5AbM (besucht am 04. 11. 2022).

[19] oli4math. *RSA: Was ist e*d mod phi(n)?* YouTube. 30. Okt. 2022. URL: https://youtu.be/WkdSa93nweg (besucht am 30. 10. 2022).

[20] oli4math. *Teilen mit Rest: Beispiel*. YouTube. 4. Nov. 2022. URL: https://youtu.be/zmUdhEgNcjo (besucht am 04. 11. 2022).

[21] oli4math. *Zahlentheorie: Beispiele zur Eulerschen phi-Funktion*. YouTube. 30. Okt. 2022. URL: https://youtu.be/nbU_hw_7ZQw (besucht am 30. 10. 2022).

[22] oli4math. *Zahlentheorie: Eulersche Phi-Funktion: phi(p) und phi(p*q)*. YouTube. 30. Okt. 2022. URL: https://youtu.be/n18Z1EDU7NA (besucht am 30. 10. 2022).

[23] oli4math. *Zahlentheorie: Letzte Stelle mit Euler-Fermat bestimmen*. YouTube. 30. Okt. 2022. URL: https://youtu.be/O-JfmkMT2bg (besucht am 30. 10. 2022).

[24] oli4math. *Zahlentheorie: Modulo Rechnung mit dem Satz von Euler-Fermat*. YouTube. 30. Okt. 2022. URL: https://youtu.be/cZEXCSDMqkY (besucht am 30. 10. 2022).

Abbildungsverzeichnis