KURS KRYPTOGRAPHIE:
Xelcgbtencuvr zvg Wnin

Mathematische Verschlüsselungssysteme und ihre Algorithmen

Einleitung

Johanne Ebel, Johanna Löhlein

Wir, Schüler der 8. und 9. Klasse, aus ganz Baden–Württemberg, kamen voller Erwartungen und sehr gespannt zur Science Academy. Wir interessieren uns alle für Mathematik und wollten wissen was wir im Kryptographiekurs lernen würden.

Wir, das sind:

Alexander Bradl, 14 Jahre alt, kommt aus Schriesheim und hat den Berufswunsch Patentanwalt.

Johanne Ebel, 15 Jahre alt, kommt aus Widdern und hat den Berufswunsch Journalistin oder möchte etwas in Richtung Medizin machen.

Sandra Hofmann, 16 Jahre alt, kommt aus Schwaikheim und will etwas im Bereich Technik machen.

Lena Kliesch, 16 Jahre alt, kommt aus Ladenburg und will eventuell Apothekerin oder Hotelmanagerin werden.

Johanna Löhlein, 15 Jahre alt, kommt aus Gammelsbach und hat den Berufswunsch Physiotherapeutin oder Ingenieurin.

Aron Philipp, 14 Jahre alt, kommt aus Zell im Wiesental und will später eventuell als Steuerberater arbeiten.

Patricia Quellmalz, 16 Jahre alt, kommt aus Cleebronn und will eventuell Mathematik studieren und vielleicht im Bereich Bank- Versicherungswesen arbeiten.

Tomma Radlow, 15 Jahre alt, kommt aus Friedrichshafen und möchte vielleicht als Autorin arbeiten.

Jan Philipp Rehner, 14 Jahre alt, kommt aus Sindelfingen und würde gerne im EDV-Bereich oder im Ingenieurwesen arbeiten.

Jonathan Striebel, 16 Jahre alt, kommt aus Bühl und könnte sich vorstellen im Bereich Musik oder Naturwissenschaften in Verbindung mit Informatik zu arbeiten.

Saskia Voß, 15 Jahre alt, kommt aus Deizisau und würde gern Raumfahrtsingenieurin oder Mathematikerin werden.

Alexander Wenz, 16 Jahre alt, kommt aus Kehl und will Journalist werden.

Wir begannen unsere Kursarbeit mit einfachen Verschlüsselungsverfahren wie Cäsar- und Vigenère–Verschlüsselung und fuhren mit XOR- und RSA–Verschlüsselung fort. Wir lernten einiges über die Programmiersprache Java und arbeiteten auch damit. Außerdem beschäftigten wir uns mit schwierigen und komplexen Beweisen. Zum Beispiel haben wir alle mathematischen Sätze

bewiesen, die zum Verständnis der RSA–Verschlüsselung nötig sind.
Dies sind alles Untergebiete der Kryptographie (gr. krytos: „verstecken“, graphein: „schreiben“). Hierbei geht es darum geheime Nachrichten sicher und unentdeckt zu ver- und entschlüsseln.

Die Kursarbeit war zwar manchmal sehr anstrengend, hat aber immer viel Spaß gemacht, wir haben ausgiebig diskutiert und viel gelacht ;). Insgesamt war die Atmosphäre sehr schön. Bei Problemen halfen wir uns gegenseitig und konnten so relativ stressfrei arbeiten.

Grundbegriffe

Tomma Radlow, Aron Philipp

Es gibt immer einen Sender, der eine Nachricht weitergeben will, und einen Empfänger der Nachricht, welcher den Text lesen will. Der Sender will dem Empfänger eine Nachricht (Originaltext) zukommen lassen, ohne dass ein externer Dritter diese lesen kann. Dazu einigen sich Sender und Empfänger auf einen gemeinsamen Schlüssel und einen Verschlüsselungsalgorithmus. Diese sind in der Regel öffentlich bekannt. Trotzdem sollte im Idealfall ein Dritter, welcher die Information abfangen will, nach dem Verschlüsseln den dabei entstandenen Geheimtext (Chiffre) nicht lesen können. Nur der Empfänger kann mithilfe des richtigen Schlüssels den Originaltext zurückgewinnen.

Nutzen

Die Kryptographie ist aus dem modernen Leben nicht mehr wegzudenken.
Wie vielseitig und doch unbemerkt Kryptographie heutzutage eingesetzt wird, wurde auch uns als Kryptographiekurs erst richtig klar, als wir Besuch vom IT-Sicherheitschef der Deutschen Bank, Herrn Esslinger, bekamen. Er hielt uns einen hochinteressanten und äußerst informativen Vortrag. Dabei erfuhren wir unter anderem, dass Banken ohne Kyptographie nicht mehr auskommen, weshalb sie oft Angestellte beschäftigen, welche nur für die Datensicherheit zuständig sind. Den Banken liegt – sowohl in ihrem eigenen als auch im Interesse der Kunden – daran, ihre Daten geheim zu halten. Außerdem sind sie laut Gesetz dazu verpflichtet. Denn in der modernen Zeit wickeln Millionen Menschen auf der ganzen Welt ihre Geschäfte übers Internet, also via Online Banking, ab. Eine Bank oder ein anderes Institut muss also gewährleisten, dass die Daten sicher übertragen werden und kein Dritter an die Informationen gelangt.
Der Nutzer authentifiziert sich erstens mit seinem Namen und seinem Kennwort; außerdem erhält er auch noch eine TAN (Trans-Aktions-Nummer). Nur der Kunde und die Bank sind in Besitz dieser Nummern. Nun kann der Kunde mit Hilfe dieser TAN seine Buchung verschlüsseln und nur die Bank ist fähig diese wieder zu entschlüsseln. Die Trans-Aktions-Nummern werden dem Kunden meist per Post zugesandt. Die TAN verfällt nach einmaligem Gebrauch. (siehe One-time-pad)
Auch bei der alltäglichen Internetnutzung gibt es das so genannte SSL (Secret Sockets Layer), bei welchem der Browser sowie der Server einen Schlüssel aushandeln, den sie dann zur verschlüsselten Kommunikation benutzen – alles vollautomatisch, ohne, dass der Benutzer etwas davon mitbekommt.
Ähnlich funktioniert das Telefonieren mit dem Handy: Während die Kommunikation zwischen Handy und Basisstation unverschlüsselt geschieht, wird die zwischen den Basisstationen ausgetauschte Information verschlüsselt. Das Ver- und Entschlüsseln geht ohne merkliche Verzögerung vonstatten.

Steganographie

Tomma Radlow, Johanne Ebel

Bei der Steganographie (griechisch: steganós = schützen und graphéin = schreiben) wird die Botschaft versteckt, so dass Unbefugte von ihrer Existenz nichts wissen sollten. Die Steganographie ist ein altbekanntes Verfahren, das bereits in der Antike angewendet wurde. So rettete die Kunst der Steganographie bereits Griechenland vor der Eroberung durch Xerxes, König der Perser. Xerxes plante einen Überraschungsangriff gegen die Griechen, doch Damartos, ein im persischen Exil lebender Grieche, erfuhr von der Aufrüstung der Perser und

warnte die Griechen vor der Invasion Xerxes. Er schaffte es mit einer List, seine Botschaft unentdeckt zu übermitteln, indem er das Wachs von einer Schreibtafel abkratzte, die Nachricht ins Holz hinein ritzte und das Wachs wieder darüber goss. Die unauffällige Tafel gelangte über die Grenze nach Griechenland. Die Nachricht wurde von den Griechen entdeckt und sie konnten sich auf den Krieg vorbereiten, somit schlug der Überraschungsangriff fehl.
Wir haben uns im Kurs ebenfalls mit Steganographie beschäftigt: Dazu verwendeten wir ein Programm, das die Buchstaben der Botschaft jeweils so codiert, dass sie einzelne Pixel eines Bildes verändern. Die Veränderungen stören den Gesamteindruck des Bildes nicht. Zum Entziffern vergleicht unser Programm nun das veränderte Bild mit dem Originalbild und kann so die versteckte Botschaft sichtbar machen.
Wir haben mit diesem Programm gearbeitet und uns gegenseitig Botschaften geschickt. Eine Botschaft ist jedoch nur so lange sicher, solange sie unentdeckt bleibt. Man könnte sie zur Sicherheit noch verschlüsseln... Steganographie und Kryptographie sollte man nicht verwechseln. Sie sind als eng verwandte Wissenschaften zu sehen, haben jedoch verschiedene Ansätze, dem Empfänger eine Information zu übermitteln.

Cäsar – Verschlüsselung

Saskia Voss, Lena Kliesch,
Alexander Wenz

Ein sehr einfaches Verfahren ist die Cäsar–Verschlüsselung, die von Gaius Julius Cäsar, z. B. im Gallischen Krieg, eingesetzt wurde und im Ersten Weltkrieg sogar noch von der russischen Armee verwendet wurde.

Um einen Klartext mit Cäsar zu verschlüsseln, in unserem Beispiel

„`Java ist ganz toll`“,

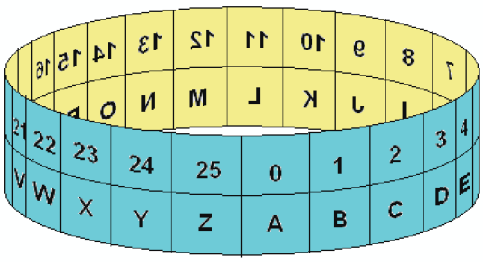
muss man jeden Buchstaben im Alphabet um z. B. 3 Stellen nach rechts verschieben, wobei diese Verschiebung den Schlüssel angibt.

Beispiel:

Originaltext:	`javaistganztoll`
Schlüssel:	3
Geheimtext:	`MDYDLVWJDQCWSOO`

Um es wieder zu entschlüsseln, verschiebt man jeden Buchstaben des Geheimtextes wieder, hier um 3 Stellen, zurück.
Um das ganze rechnerisch zu lösen, ordnen wir zuerst jedem Buchstaben des Klartextalphabets eine Zahl zu: `a` wird zu `0`, `b` zu `1`, ... und `z` zu `25`. In unserem Beispiel wird `j` zu `9`. Nun addieren wir dazu unseren Schlüssel `3` und erhalten die Zahl `12`, was den Buchstaben `M` entspricht.

Allerdings wird es beim Verschlüsseln des Buchstaben `z`(=25) problematisch, da wir als Ergebnis `28` herausbekommen, was keinem Buchstaben entspricht.



Wie man an der Grafik sieht funktioniert die Cäsar–Verschlüsselung wie eine Schleife, denn immer wenn man über `z` (25) hinauskommt, fängt man wieder von vorne im Alphabet an. Damit wir auch mehrmals `a` (0) überschreiten können, subtrahieren wir nicht die Alphabetlänge `26`, sondern betrachten den Rest bei Division durch `26`. Im obigen Beispiel rechnen wir $28 \div 26$ und erhalten `1 Rest 2`. Wichtig ist nur der Rest, weil er den Geheimtextbuchstaben angibt. Der andere Teil des Ergebnisses, die `1`, ist uninteressant, da es unwichtig ist, wie oft

man über das „a“ bzw. das „z“ hinausgeht. Statt $28 \div 26 = 1$ Rest 2 schreibt man $28 \bmod 26 = 2$. Die Abkürzung „mod“ steht für „Modulo“ und bedeutet, dass wir nur den Rest der Division betrachten. (siehe Rechenregel mit modulo) Somit können wir in unserem Beispiel z (=25) mit dem Buchstaben C (=2) verschlüsseln. Weil man für das Ver- sowie Entschlüsseln den gleichen Schlüssel verwendet, gehört die Cäsar–Verschlüsselung zu den symmetrischen Verschlüsselungsverfahren.

So einfach und praktisch das Verfahren ist, so einfach und schnell lässt es sich auch ohne Schlüssel wieder knacken:

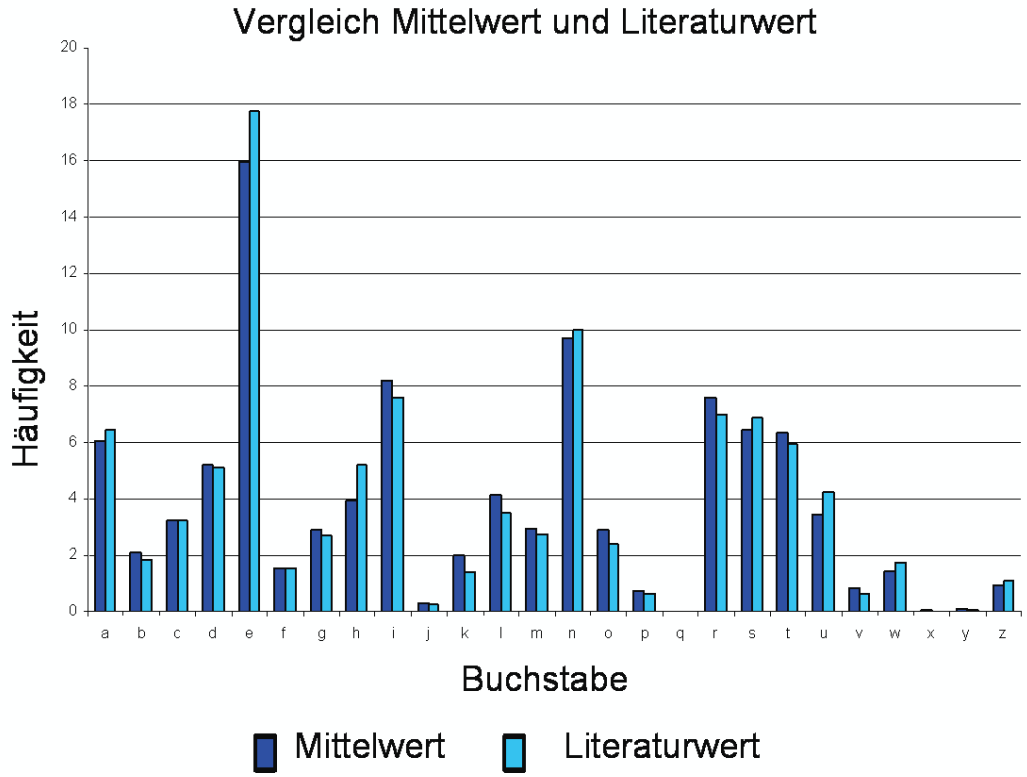
In jeder Sprache hat jeder Buchstabe des Alphabets eine charakteristische Häufigkeit: so kommt z. B. das „e“ in einem deutschen längeren Text am häufigsten vor, während das q, x oder y seltener vorkommt.

Um diese unterschiedlichen Häufigkeiten zu untersuchen, haben wir in unserm Kurs 6 beliebige deutschsprachige Texte ausgewählt und eine Häufigkeitsanalyse mit Hilfe eines Computerprogramms angewendet.

Zur Veranschaulichung benutzen wir ein Balkendiagramm, das die unterschiedlichen Häufigkeiten der Buchstaben miteinander vergleicht. Bei diesem Diagramm wird ein Sollwert aus der Literatur mit unseren eigenen Ergebnissen aus dem Kurs verglichen. Die geringen Abweichungen entstehen, da wir für unseren Mittelwert nur 6 Texte untersucht haben, während dem Literaturwert viel mehr Texte zugrunde liegen.

Wenn wir einen mit Cäsar verschlüsselten Text entschlüsseln wollen, wenden wir die Häufigkeitsanalyse an und werden einen Buchstaben mit einer sehr hohen Häufigkeit herausbekommen.

Wenn z.B. H am häufigsten vorkommt, können wir davon ausgehen, dass H dem Klartextbuchstaben e entspricht, da dieser Buchstabe im Deutschen am meisten vorkommt. Weil wir nun wissen, dass **e** auf **H** abgebildet wird, kann man nun die Verschiebung, hier 3, herausfinden und den restlichen Text übersetzten.



Die Cäsar Verschlüsselung hat heute für die praktische Anwendung keine Bedeutung, da man die Botschaft genauso einfach ent- wie verschlüsseln kann. Deswegen wurden kompliziertere und sichere Verfahren entwickelt. Anhand der Cäsar–Verschlüsselung haben wir jedoch viel Prinzipielles über Ver- und Entschlüsseln gelernt.

Die Vigenère–Verschlüsselung

Johanna Löhlein und Saskia Voss

1460 wurde diese polyalphabetische Verschlüsselung, die auf mindestens zwei Geheimtextalphabeten beruht, von dem Mathematiker Leon Battista Alberti entdeckt. Von Johannes Trithemius, einem deutschen Abt, und dem, italienischen Wissenschaftler Giovanni Porta, wurde sie weiterentwickelt. 1562 wurde sie schließlich von Blaise de Vigenère vollendet.

Wir wollen den Originaltext:

„`Die Science Academy ist toll`“

mit Vigenère verschlüsseln. Dazu suchen wir uns ein Schlüsselwort, hier `KURS` und schreiben dieses so oft unter den Originaltext bis jeder Buchstabe einen zugehörigen Schlüsselbuchstaben hat.

Beispiel:

Originaltext:
`diescienceacademyisttoll`
Schlüsselwort:
`KURSKURSKURSKURSKURSKURS`
Geheimtext:
`??????????????????????????`

Um den Text zu verschlüsseln, verwendet man ein Vigenèrequadrat, in dem alle möglichen Cäsarverschiebungen aufgeführt sind. Nun sucht man im oben gezeigten Beispiel die Spalte mit dem Originaltextbuchstaben `h` und die Zeile mit der man `h` verschlüsseln möchte, also die Zeile des ersten Schlüsselwortbuchstaben `w`. Diese erste Verschlüsselung entspricht einer Cäsarverschiebung um 22. An der Schnittstelle ergibt sich der erste Buchstabe des Geheimtextes, in diesem Fall ein `D`.

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
1	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
2	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
3	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
4	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
5	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
6	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
7	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
8	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
9	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
10	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
11	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
12	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
13	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
14	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
15	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
16	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
17	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
18	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
19	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
20	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
21	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
22	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
23	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
24	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
25	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y
26	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z

So fährt man mit den folgenden Originalbuchstaben fort, d. h. man sucht immer die passenden Spalten und Zeilen und notiert sich den Buchstaben an der Schnittstelle. Dabei erhält man folgenden Geheimtext:

Originaltext:
`diescienceacademyisttoll`

Schlüsselwort:
`KURSKURSKURSKURSKURSKURS`

Geheimtext:
`NCVKMCVFMYRUKXVEICJLEICD`

Wie man sieht, wird jeder Buchstabe mit der Caesar–Verschlüsselung verschlüsselt.

Da derselbe Originaltextbuchstabe jetzt nicht mehr immer demselben Geheimtextbuchstaben entspricht, ist die Häufigkeitsanalyse unbrauchbar. Deswegen wurde das Vigenère–Verfahren auch „Le Chiffre indéchiffrable“, auf Deutsch „der unknackbare Code“, genannt.

Die Kasiski – Methode

Die Kasiski–Methode dient zum Harausfinden der Schlüssellänge. Diese Methode wurde als erstes vermutlich im Jahre 1854 von Charles Babbage entdeckt, der sie allerdings nicht veröffentlichte. Erst 10 Jahre später erfand Friedrich Wilhelm Kasiski unabhängig von Babbage dasselbe Verfahren erneut und veröffentlichte es.

Dieses Verfahren greift an den Wiederholungen an, die bei Vigenère auftauchen, da mehrere Buchstaben mit demselben Geheimtextalphabet verschlüsselt werden.

Beim oben genannten Beispiel wird jeder 4. Buchstabe mit demselben Geheimtext verschlüsselt.

Als erstes muss man bei einem mit Vigenère verschlüsselten Text nach sich wiederholenden Zeichenfolgen suchen, die mindestens 3 Zeichen lang sind.

Beispiel:

```
EYRYC FWLJH FHSIU BHMJO UCSEG
TNEER FLJLV SXMVY SSTKC MIKZS
JHZVB FXMXK PMMVW OZSIA FCRVF
TNERH MCGYS OVYVF PNEVH JAOVW
UUYJU FOISH XOVUS FMKRP TWLCI
FMWVZ TYOIS UUIIS ECIZV SVYVF
PCQUC HYRGO MUWKV BNXVB VHHWI
FLMYF FNEVH JAOVW ULYER AYLER
VEEKS OCQDC OUXSS LUQVB FMALF
EYHRT VYVXS TIVXH EUWJG JYARS
ILIER JBVVF BLFVW UHMTV UAIJH
PYVKK VLHVB TCIUI SZXVB JBVVP
VYVFG BVIIO VWLEW DBXMS SFEJG
FHFVJ PLWZS FCRVU FMXVZ MNIRI
GAESS HYPFS TNLRH UYR
```

(bei diesem Text wurden zur Übersichtlichkeit die Zeichen in 5er Päckchen unterteilt. Normalerweise würde alles ohne Leerzeichen stehen)

Im Beispiel finden wir, dass sich die Zeichenfolge `NEVHJAOVWU` nach `90` Zeichen wiederholt. Die Zeichenfolge `VYVFP` wiederholt sich nach `70` Zeichen usw.

Nun tragen wir alle sich wiederholende Zeichenfolgen mit ihren Abständen zueinander in eine Tabelle ein:
Weiterhin markieren wir alle Teiler dieser Abstände.

Tabelle für den Beispieltext:

wiederholte Folge	Zwischenraum	2	3	4	**5**	6	7	8	9	10
`NEVHJAOVWU`	90	x	x		**x**	x			x	x
`VYVFP`	70	x			**x**		x			x
`JBVV`	40	x		x	**x**			x		x
`XVB`	125				**x**					

Hier sieht man, dass nur der Teiler 5 in allen Abständen vorkommt.
Jetzt erstellen wir 5 neue Texte, wobei der erste Text nur aus den 1., 6., 11., 16., usw. Buchstaben des Geheimtextes besteht. Der zweite Text besteht aus den 2., 7., 12., 17., usw. Buchstaben. Der fünfte Text besteht schließlich aus den 5., 10., 15., 20., usw. Buchstaben. Jeder dieser Einzeltexte kann nun mit der Häufigkeitsanalyse untersucht werden. Dabei stellen wir fest, dass im ersten Text das „F“ als häufigster Buchstabe auftritt. Wenn wir weiter annehmen, dass das „e“ auf eben diesem Buchstaben verschlüsselt wurde, gelangen wir zum Caesarcode 1, aus„e“ wird „F“, bzw. aus „a“ wird „B“. Genauso verfahren wir mit allen anderen Texten und erhalten das Schlüsselwort BUERO, mit dem wir nun den restlichen Text entschlüsseln können.

Die Kasiski – Methode dient dazu, die Länge des Schlüsselwortes herauszufinden. Da die Kasiski – Methode auf Wiederholungen basiert, funktioniert sie nur mit längeren Texten.

One Time Pad

Beim One Time Pad wählen wir zufällig einen Buchstaben mit dem wir den ersten Originaltextbuchstaben verschlüsseln. Für den zweiten Originaltextbuchstaben wählen wir wieder zufällig einen neuen Buchstaben usw. Wie man sieht ist das nichts anderes als eine Vigenère–Verschlüsselung mit unendlich langem Schlüsselwort.

Die Kasiski Methode wird unbrauchbar, da beim One Time Pad keine Wiederholungen des Schlüsselwortes auftreten, weil der Schlüssel genauso lang ist wie der Text. Somit ist das One Time Pad ein absolut sicheres Verschlüsselungsverfahren.

Allerdings gibt es zwei Nachteile, denn man darf das One Time Pad nur einmal benutzen und man muss den Schlüssel jedes Mal neu und sicher überbringen.

Allgemeiner und erweiterter Euklidscher Algorithmus und dessen Anwendung

Patricia Quellmalz

Allgemeiner Euklidscher Algorithmus

Ich möchte jetzt eines unserer mathematischen Themen vorstellen, welches unserem Kryptografiekurs die Grundlagen fürs Verständnis von RSA lieferte. Zwei positive natürliche Zahlen a und b haben stets einen größten gemeinsamen Teiler, kurz $\mathrm{ggT}(a, b)$. Mit Hilfe des so genannten Euklidschen Algorithmus kann man diesen berechnen. Folgende Idee steckt dahinter:
Für zwei natürliche Zahlen a und b mit $a \geq b > 0$ gilt $a = q \cdot b + r$, wobei q und r eindeutige natürliche Zahlen sind mit $r < b$.
Für die Zahlen $a = 49$ und $b = 5$ wäre dies z. B. $49 = 9 \cdot 5 + 4$ (also $q = 9$ und $r = 4$).

Es gilt: $\mathrm{ggT}(a, b) = \mathrm{ggT}(b, r)$.
Beweis dieser Behauptung ist im Anhang nachzulesen.

Bevor der Euklidsche Algorithmus allgemein dargestellt wird, soll er zum besseren Verständnis an folgendem Beispiel illustriert werden:

Beispiel 1

Bestimme mit Hilfe des Euklidschen Algorithmus den größten gemeinsamen Teiler von 792 und 75.

$$\begin{aligned} 792 &= 10 \cdot 75 + 42 \\ 75 &= 1 \cdot 42 + 33 \\ 42 &= 1 \cdot 33 + 9 \\ 33 &= 3 \cdot 9 + 6 \\ 9 &= 1 \cdot 6 + 3 \\ 6 &= 2 \cdot 3 + 0 \end{aligned}$$

Da nun der Rest = 0 ist, wird der Algorithmus abgebrochen. Somit ist

$$\text{ggT}(792, 75) = 3.$$

Erweiterter Euklidscher Algorithmus

Beim erweiterten Euklidschen Algorithmus geht es nicht nur darum, den größten gemeinsamen Teiler zweier Zahlen zu bestimmen, sondern den größten gemeinsamen Teiler als Summe zweier ganzer Zahlen darzustellen.

Mathematisch exakt formuliert sieht das so aus:

Satz:

Es seien a, b und d natürliche Zahlen.
Es ist $d = \text{ggT}(a, b)$ genau dann, wenn es Zahlen s, $t \in \mathbb{Z}$ gibt mit $a \cdot s + b \cdot t = d$.

Da für teilerfremde Zahlen a und b der größte gemeinsame Teiler 1 ergibt, kann man den Satz noch folgendermaßen spezialisieren:

Zwei natürliche Zahlen a und b sind genau dann teilerfremd, wenn es Zahlen s, $t \in \mathbb{Z}$ gibt mit $a \cdot s + b \cdot t = 1$.

Mit Hilfe des erweiterten Euklidschen Algorithmus gelingt es nun, die Zahlen s und t aus obigem Satz zu ermitteln, was uns später zur Schlüsselerstellung für eines der sichersten Verschlüsselungssysteme dienen wird.

Der erweiterte Algorithmus wird zunächst an folgendem Beispiel dargestellt:

Beispiel 2

Gemäß obigem Beispiel gilt
$\text{ggT}(792, 75) = 3$.
Bestimme die ganzen Zahlen s und t so dass gilt:
$792 \cdot s + 75 \cdot t = 3$.

Aus dem vorletzten Schritt des Beispiels 1 folgt:

$$3 \;=\; 9 - 1 \cdot 6$$

Aus dem Schritt davor, bei dem die Zahl 6 als Rest aufgetreten ist, kann nun 6 entsprechend wieder als Differenz dargestellt werden.

$$\begin{aligned} \Rightarrow\ 3 &= 9 - 1 \cdot (33 - 3 \cdot 9) \\ &= -1 \cdot 33 + 4 \cdot 9 \end{aligned}$$

Analog geht es nun weiter:

$$\begin{aligned} \Rightarrow\ 3 &= -1 \cdot 33 + 4 \cdot (42 - 1 \cdot 33) \\ &= 4 \cdot 42 - 5 \cdot 33 \\ \Rightarrow\ 3 &= 4 \cdot 42 - 5 \cdot (75 - 1 \cdot 42) \\ &= -5 \cdot 75 + 9 \cdot 42 \\ \Rightarrow\ 3 &= -5 \cdot 75 + 9 \cdot (792 - 10 \cdot 75) \\ &= -95 \cdot 75 + 9 \cdot 792 \end{aligned}$$

Damit gilt $s = 9$ und $t = -95$.

Primzahlen:

Alexander Bradl

Ein weiteres mathematisches Phänomen, das uns beschäftigte sind die Primzahlen.
Primzahlen sind Zahlen, welche nur durch 1 und sich selbst teilbar sind.
Bis heute lässt sich keine Regelmäßigkeit in ihrem Auftreten feststellen, mit der man etwa voraussagen könnte, wie man zu einer bekannten Primzahl die nächst größere finden kann.
Doch weiß man, dass es unendlich viele solcher Primzahlen gibt.
Der Beweis dieser Behauptung ist im Anhang zu finden.

Boole'sche Funktion/ XOR–Verschlüsselung

Lena Kliesch, Alexander Wenz

Eine weitere Verschlüsselungstechnik ist die XOR–Verschlüsselung, die auf der Boole'schen Binärfunktion XOR basiert. Diese Funktion verknüpft zwei Boole'sche Werte `a` und `b`, die nur mit `0` und `1` belegt sein können, mit der XOR–Verknüpfung zu einem dritten Wert `c`. `0` und `1` lassen sich als falsch und wahr interpretieren. Das Kürzel „XOR" steht für „e**x**clusive **or**", was auf Deutsch übersetzt „entweder... oder" bedeutet. **Entweder `a` oder `b`** muss wahr sein, damit `c` wahr ist. Die Tabelle zeigt alle Fälle.

a	b	c
0	0	0
0	1	1
1	0	1
1	1	0

Wollen wir einen Klartext, der mit einer Folge von Nullen und Einsen ausgedrückt wird, mit der XOR–Verschlüsselung verschlüsseln, so verknüpft man ihn mit einem Schlüssel, der aus einer gleich langen Folge zufällig gewählter Nullen und Einsen besteht, mit XOR zu einem Geheimtext. Somit ist die XOR–Verschlüsselung ein One–Time–Pad.
Will man z. B. `1100` mit dem Schlüssel `0101` verschlüsseln, so verknüpft man jeden einzelnen Wert dieser Zahlenfolge per XOR mit dem entsprechenden Wert des Schlüssels und erhält den Wert des „Geheimtexts".

Klartext	1	1	0	0
Schlüssel	0	1	0	1
Geheimtext	1	0	0	1

(1 XOR 1 = 0)

Um den Geheimtext wieder zu entschlüsseln, verknüpfen wir diesen mit dem gleichen Schlüssel per XOR–Verknüpfung, denn es gilt:
$(a \,\mathrm{XOR}\, b)\ \mathrm{XOR} b = a$.

Die XOR–Verschlüsselung ist ebenfalls ein symmetrisches Verschlüsselungsverfahren, da man zum Ver- und Entschlüsseln denselben Schlüssel verwendet. Die XOR–Verschlüsselung wurde z. B. zur Zeit des Kalten Krieges verwendet, um Botschaften zwischen dem amerikanischen Präsidenten und dem sowjetischen Generalsekretär zu schützen (= „heißer Draht").

Die Sicherheit der XOR–Verschlüsselung ist aber nur dann gewährleistet, wenn der Schlüssel rein zufällig erstellt wurde. Die Schwachstelle ist der Schlüsselaustausch: Bevor man verschlüsseln kann, muss man den langen Schlüssel zunächst auf einer sicheren Leitung, die nicht abgehört werden kann, austauschen. Damit ist das Problem nur verschoben. Aufgrund dieses Problems ist die XOR–Verschlüsselung eine sehr unpraktische Verschlüsselung und wird deshalb selten angewendet.

RSA – Allgemein

Aron Philipp

RSA ist die derzeit modernste Verschlüsselung. Sie wird von vielen Unternehmen, Privatnutzern und anderen angewandt.
Die Grundidee stammt von Whitfield Diffie und Martin Hellmann. Sie kamen unter anderem auf die Idee mit dem Vergleich mit einem Briefkasten. Jemand hat einen Briefkasten, in den auch jeder etwas hereinwerfen kann, aber nur der Inhaber hat den Schlüssel und kann den Briefkasten öffnen. Diesem System wollten sie ein Verschlüsselungssystem nachempfinden. Das nennt man asymmetrische Kryptographie. Sie veröffentlichten ihre Entdeckungen 1976.
Auf diesem Prinzip beruht auch RSA, hier gibt es den Public Key, der jedem zugänglich ist, aber nur der Adressat hat den Private Key.

1970 hatten Ellis, Cocks und Williamson schon im Government Communication Headquarters eine ähnliche Idee wie Hellmann und Diffie. Allerdings wurde diese nicht weiter verfolgt und aus Geheimhaltungsgründen auch nicht weiterverarbeitet. Rivest, Shamir und Adleman gelang dann die endgültige Ausarbeitung von RSA. Sie konnten dieses Verfahren auch mathematisch beweisen.
Warum bietet RSA eine so große Sicherheit?
Die komplette RSA–Verschlüsselung basiert auf Primzahlen. Diese sind ohne System auf dem Zahlenstrahl verteilt und es gibt auch keine Formel um eine solche Primzahl zu ermitteln.
Zur Schlüsselerzeugung wählt man zwei möglichst große Primzahlen, deren Produkt man bildet. Um aus dem öffentlichen Schlüssel den privaten zu erhalten, müsste man das riesige Produkt in Primfaktoren zerlegen.
Leider – oder zum Glück – ist dies mit Computern heutiger Bauart nicht machbar. Sie würden für eine Lösung viele Jahre brauchen. RSA kann nicht nur zum Verschlüsseln, sondern auch zum Signieren von Botschaften verwendet werden. Durch eine digitale Signatur wird sichergestellt, dass eine Botschaft tatsächlich von dem angegebenen Absender stammt. Zum Verschlüsseln und Signieren von E-Mails gibt es ein von Phil Zimmermann erfundenes Programm, Pretty Good Privacy genannt.

Der RSA–Algorithmus

Aron Philipp

Ein Algorithmus ist eine Abfolge von Anweisungen, vergleichbar mit einem Kochrezept.
Es gibt einige Regeln:
Der Algorithmus muss eindeutig, endlich und ausführbar sein.

Im Folgenden wird der Algorithmus zur Erzeugung eines RSA – Schlüssels anhand eines Beispiels erklärt. Der Einfachheit halber sind die Zahlen klein gewählt, normalerweise befinden sich diese in einer Größenordnung von 10^{130}.

1. Man wähle zwei Primzahlen p und q, die möglichst groß sein sollen.

   $$p = 5; \quad q = 11$$

2. Das Produkt n aus p und q berechnen.

   $$n = 5 \cdot 11 = 55$$

3. Nun bilde man die Eulersche Funktion φ von n:

   $$\varphi(n) = (p-1) \cdot (q-1).$$

   Also:

   $$\varphi(55) = 4 \cdot 10 = 40.$$

4. Wähle ein e, sodass e teilerfremd zu $\varphi(n)$ ist, das heißt:

   $$\mathrm{ggT}(e, \varphi(n)) = 1.$$

   In unserem Fall wäre z. B. $e = 3$, denn

   $$\mathrm{ggT}(40, 3) = 1.$$

5. Berechne d, sodass gilt:

   $$(e \cdot d) \bmod \varphi(n) = 1$$

   $$(3 \cdot d) \bmod 40 = 1$$

   $$d = 27$$

Nun haben wir einen sogenannten „Public Key“, nämlich e und n. Das Gegenstück dazu bildet der „Private Key“: d und n.

Die Verschlüsselung mit RSA

Zuerst hat man den Originaltext. Dieser besteht aus Buchstaben, Zeichen und Zahlen. Allerdings braucht man für eine RSA–Verschlüsselung Zahlen.
Deshalb behilft man sich des ASCII–Codes. Dieser ordnet jedem Symbol eine Zahl zwischen 0 und 255 zu. Anhand eines Beispiels wird nun das Verfahren erklärt:

1. Man zerteilt den Originaltext in einzelne Pakete.
   Eine Verschlüsselung einzelner Buchstaben wäre sinnlos, da dann die Häufigkeitsanalyse greift.
2. Jedes Zeichen wird mit Hilfe des ASCII–Codes in eine Zahl umgewandelt.
3. Man reiht die Zahlen eines jeden einzelnen Päckchens aneinander. Bei dem letzten werden Nullen angefügt, damit es auch dieselbe Größe wie die anderen hat.
4. Nun kommt die eigentliche Verschlüsselung mit RSA:
   c ist der verschlüsselte Text.

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

Jedes Paket wird einzeln verschlüsselt.

Die Entschlüsselung mit RSA verläuft eigentlich genau gleich.
Die Geheimbotschaft wird wieder in Pakete, die gleich groß sind wie bei der Verschlüsselung, eingeteilt. Diese werden einzeln entschlüsselt.

$$m = c^d \bmod n$$

m ist wieder der Originaltext.

Die Eulersche Funktion

Jan Philipp Rehner

Im Folgenden werden einige Sätze dargestellt, die wir im Kurs bewiesen haben. Diese Beweise sind für Interessierte im Anhang zu finden.

Definition

Sei n eine natürliche Zahl: φ heißt Eulersche Funktion, wenn $\varphi(n)$ die Anzahl aller zu n teilerfremden natürlichen Zahlen $\leq n$ ist.
Kurz:

$$\begin{aligned}
\varphi(n) &= |\{k \in \mathbb{N} \mid \mathrm{ggT}(k, n) = 1 \wedge k \leq n\}| \\
\varphi(1) &= |\{1\}| = 1 \\
\varphi(2) &= |\{1\}| = 1 \\
\varphi(3) &= |\{1, 2\}| = 2 \\
\varphi(4) &= |\{1, 3\}| = 2 \\
\varphi(5) &= |\{1, 2, 3, 4\}| = 4 \\
\varphi(6) &= |\{1, 5\}| = 2 \\
\varphi(7) &= |\{1, 2, 3, 4, 5, 6\}| = 6 \\
\varphi(8) &= |\{1, 3, 5, 7\}| = 4 \\
\varphi(9) &= |\{1, 2, 4, 5, 7, 8\}| = 6
\end{aligned}$$

$|M|$ ist die Mächtigkeit der Menge M. Sie gibt an, wie viele Elemente M hat.

Satz 1.1

Ist p eine Primzahl, dann ist: $\varphi(p) = p - 1$.

Satz 1.2

Seien p, q zwei verschiedene Primzahlen, dann gilt:
$\varphi(p \cdot q) = (p - 1)(q - 1) = \varphi(p) \cdot \varphi(q)$.

Rechenregeln mit Modulo

Alexander Bradl

$17 = 5 \cdot 3 + 2$
Das bedeutet, dass man bei der Division von 17 durch 3 den Rest 2 erhält. Dafür schreibt man: $17 \bmod 3 = 2$ Die Zahlen die diese „Reste“ annehmen können sind 0 , 1 oder 2. Allgemein gilt: Die „Reste“ bei einer Division durch n können die Werte 0 bis $n - 1$ annehmen.

Rechenregeln:

Regel 1:

Sei $a_1 \bmod n = b_1$ und sei $a_2 \bmod n = b_2$
Dann gilt $a_1 \cdot a_2 \bmod n = b_1 \cdot b_2 \bmod n$.

Regel 2:

Aus $a \bmod n = b \bmod n$ folgt
$a^k \bmod n = b^k \bmod n$

Ein Spezialfall dazu ist:
$a \bmod n = 1$ $a^k \bmod n = 1$
(vergleiche Satz oben)

Der Satz von Euler und der Beweis des RSA–Algorithmus

Patricia Quellmalz

Der Satz von Euler

Es sei n eine natürliche Zahl mit $n > 1$. Weiterhin sei m eine zu n teilerfremde natürliche Zahl. Dann gilt

$$m\varphi(n) \bmod n = 1$$

Hierzu ein Beispiel:
Es sei $n = 14$, $m = 3$ und

$$\varphi(n) = \varphi(14) = 6.$$

Da $3^6 = 729$ und $729 = 52 \cdot 14 + 1$ gilt $3^6 \bmod 14 = 1$, was den Satz von Euler für diesen speziellen Fall bestätigt.

Kleiner Fermat

> „Dieser Satz, welcher sowohl wegen seiner Eleganz als wegen seines hervorragenden Nutzens höchst bemerkenswert ist, wird nach seinem Erfinder Fermatsches Theorem genannt.“
> C.F.Gauß

Pierre de Fermat (1607 - 1665) war ein französischer Mathematiker, seine berühmteste Entdeckung war der Satz, der heute Fermat´s kleiner Satz genannt wird.

Kleiner Fermat:

Es sei p eine Primzahl. Dann gilt:

$$m^{p-1} \bmod p = 1.$$

Die Aussage dieses Satzes erhält man, wenn man im Satz von Euler $n = p$ damit $\varphi(n) = p - 1$ setzt.
Historisch stand am Anfang der Fermatsche Satz. Leonhard Euler (1707 - 1783) verallgemeinerte ihn in die nach ihm benannte Form.

Der Beweis des RSA–Algorithmus

Das Beweisen des RSA–Algorithmus hat unseren Kurs fast einen kompletten Tag beschäftigt und setzt einige tief greifende mathematische Grundlagen voraus.
Zu Beweisen ist, dass beim Entschlüsseln des Geheimtextes auch tatsächlich der vom Sender verschlüsselte Klartext gefunden wird.

Wer sich für den Beweis interessiert, findet diesen im Anhang.

Quellenangabe:

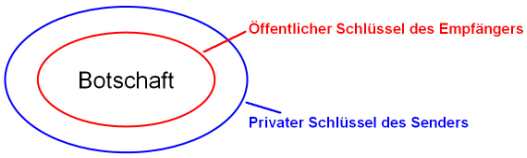
„Zahlentheorie für Einsteiger“ von Andreas Bartholomé, Josef Rung, Hans Kern, 3.Auflage, Vieweg-Verlag

Ausblick

Sandra Hofmann, Jonathan Striebel

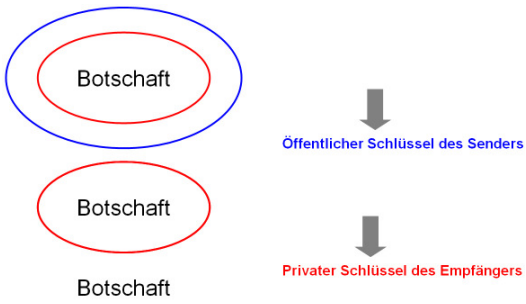
Digitale Signatur

Wer garantiert, dass eine zugeschickte verschlüsselte Nachricht auch wirklich von dem stammt, der vorgibt Sender zu sein? Dieses Problem wird durch die digitale Signatur gelöst:
Als erstes verschlüsselt der Sender seinen Text mit dem öffentlichen Schlüssel des Empfängers. Danach verschlüsselt der Sender den Text ein zweites Mal mit seinem eigenen privaten Schlüssel. Das funktioniert, da Verschlüsselungs- und Entschlüsselungsalgorithmus durch die gleichen mathematischen Funktionen dargestellt sind.
Diese zweite Verschlüsselung trägt nichts zur Sicherheit des Textes bei, sondern zeigt nur eindeutig, wer der Sender ist. Das kann man so einsehen:



Bekommt der Empfänger die Botschaft, entschlüsselt er als erstes mit dem öffentlichen Schlüssel des Senders. Dies kann jeder machen, aber es gibt auch noch die zweite Verschlüsselungsebene. Diese kann nur der Empfänger mit seinem privaten Schlüssel entschlüsseln und so als einziger die Nachricht lesen.

Für sehr lange Texte ist RSA zu aufwändig. Daher verwendet man bei der elektronischen Datenübermittlung einen anderen Algorithmus, lediglich der Schlüssel wird mit RSA signiert, ver- und entschlüsselt.



Quantencomputer

Wie schon vorher erwähnt, ist RSA ohne privaten Schlüssel nicht zu entschlüsseln. Prinzipiell gibt es die Möglichkeit aus dem öffentlichen Schlüssel den privaten zu errechnen, aber dazu benötigen 100 Mio. PCs ca. 1000 Jahre.
Aus der Sicht der Kryptologen besteht in Zukunft die „Gefahr“, RSA mit Quantencomputern zu entschlüsseln. Das Verfahren basiert auf Quantenphysik und ist sehr viel schneller als Verfahren auf herkömmlichen Rechnern. Sie sind allerdings bisher nur Theorie und noch nicht einsatzbereit. Wenn es sie aber irgendwann geben sollte, wird RSA nutzlos. Deshalb überlegen sich Kryptographen schon jetzt neuere, noch sicherere Verschlüsselungsmethoden, die selbst wieder auf Phänomenen der Quantenphysik basieren: die Quantenkryptographie.

Java Programmierung

Jonathan Striebel, Sandra Hofmann, Jan Philipp Rehner

Java allgemein

Java ist eine rein objektorientierte Programmiersprache, das macht sie zu einer der modernsten Sprachen. Die einzelnen Projekte sind in Pakete und Klassen gegliedert.
Eine weitere Stärke ist die Plattformunabhängigkeit von Java, d.h. die Programme laufen auf allen Betriebssystemen.

Programmierung in Java

Hier folgt der wesentliche Teil der Klasse Caesar. Sie enthält die beiden Methoden `enKrypt` und `deKrypt`:

Die Methode `enKrypt`, enthält als Parameter den Originaltext als String, das ist der zu verschlüsselnde Text, und den Schlüssel als Integer, eine ganze Zahl. Die Methode verschlüsselt den übergebenen Text und gibt ihn als String zurück.

```
public class Caesar {

      public static String enKrypt (String originalText, int schluessel){

            […]

            geheimText  +=  (char)((originalText.charAt(i)  -  (byte)`A`  +
            schluessel) % 26 + (byte)`A`);
      }
```

Letzteres ist der eigentliche Caesar–Algorithmus, durch den jeder einzelne Buchstabe durch den Geheimtextbuchstaben ersetzt wird. Die Operation mod26 garantiert, dass der resultierende Wert kleiner als 26 ist (% ist das Symbol für mod in Java)

Das Entschlüsseln ist im Wesentlichen der gleiche Algorithmus wie das Verschlüsseln. Die Verschiebung erfolgt lediglich in umgekehrter Richtung. Die Java Methode `deKrypt` ruft die Methode `enKrypt` mit veränderten Parametern auf:

```
      public static String deKrypt (String geheimText, int schluessel) {
            return enKrypt(geheimText, 26 - schluessel);
      }
}
```

GUI

Der Begriff GUI steht für Graphical User Interface (grafische Benutzeroberfläche) und bezeichnet eine Softwarekomponente, die einem Computerbenutzer die Interaktion mit der Maschine über grafische Elemente (Buttons, Symbole, Textfelder, etc.) unter Verwendung eines Zeigegerätes (wie einer Maus) erlaubt. GUIs sind von den Fachklassen getrennt. Konkret sah das dann in unserem Beispiel der Caesar–Verschlüsselung so aus:

Der Originaltext und der Schlüssel werden in die jeweiligen Fenster eingegeben. Durch Anklicken des „Verschlüsseln“–Buttons werden sie an die Methode `enKrypt` übergeben, die den Geheimtext berechnet und ihn an die GUI zurückgibt. Der Geheimtext wird nun im Fenster Geheimtext ausgegeben. Umgekehrt wird beim Klicken auf „Entschlüsseln“ die Methode `deKrypt` aufgerufen.

Anhang (Beweise)

Kapitel Mathematische Hintergründe:

Zu Euklidscher Algorithmus:

Sei $a = q \cdot b + r$ mit $a \geq b > 0$.
q und r eindeutige natürliche Zahlen mit $r < b$. Es gilt:

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

Beweis dieser Behauptung:

Gilt $d = \text{ggT}(a, b)$, dann teilt d neben den Zahlen a und b auch $r = a - q \cdot b$ (die Zahl d kann bei dem Ausdruck $a - q \cdot b$ ausgeklammert werden).
Damit ist d auch ein Teiler von b und r. Deshalb ist jeder Teiler von a und b auch $\leq d$. Also ist d der größte gemeinsame Teiler von a und b und es gilt

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

Zu Primzahlen:

Satz

Es gibt unendlich viele Primzahlen.

Beweis

Annahme: Es gibt endlich viele Primzahlen $p_1, p_2, \ldots, p_n$ wobei p_n die größte ist. Wir konstruieren eine Zahl P mit:

$$P = p_1 \cdot p_2 \cdot \cdots \cdot p_n + 1$$

P ist durch keine der Primzahlen p_1 bis p_n ohne Rest teilbar, damit ist P nur durch sich selbst und durch 1 teilbar. P ist damit eine Primzahl und nach Konstruktion größer als p_n. Damit ist die Annahme p_n sei größte Primzahl falsch. Also gibt es unendlich viele Primzahlen. q.e.d

Kapitel RSA – Beweis:

Zu Eulersche Funktion:

Beweis Satz 1.1.:

Sei p eine Primzahl.
Die einzige Zahl $\leq p$, die mit p nicht teilerfremd ist, ist p selbst, alle anderen Zahlen sind teilerfremd zu p.
Deshalb gibt es $p - 1$ Zahlen $\leq p$, die zu p teilerfremd sind.

Beweis Satz 1.2:

Es gibt $p \cdot q - 1$ natürliche Zahlen, die kleiner sind als $p \cdot q$.

Nicht teilerfremd zu $p \cdot q$ sind alle Zahlen, die q oder p als Teiler haben und kleiner als $p \cdot q$sind:

p; 2p; 3p; ...; (q - 1)p (q-1 Zahlen)
q; 2q; 3q; ...; (p - 1)q (p-1 Zahlen)

Die Zahlen $i \cdot p$, $j \cdot q$ sind für $i \leq q - 1$ und $j \leq p - 1$ paarweise verschieden, da das kleinste gemeinsame Vielfache von p und q $p \cdot q$ ist, da p und q Primzahlen sind.

Somit gilt:

$$\begin{aligned} \varphi(p \cdot q) &= (p \cdot q - 1) - (p - 1) - (q - 1) \\ &= p \cdot q - p - q + 1 \\ &= (p - 1) \cdot (q - 1) \end{aligned}$$

Zu Rechenregeln mit Modulo:

Beweis zu Regel 1

$a_1 \bmod n = b_1$ bedeutet $a_1 = k_1 \cdot n + b_1$ mit $0 \leq b_1 < n$.
Analog gilt für $a_2 : a_2 = k_2 \cdot n + b_2$.
Wenn man a_1 und a_2 multipliziert erhält man:

$$\begin{aligned} a_1 \cdot a_2 &= (k_1 \cdot n + b_1) \cdot (k_2 \cdot n + b_2) \\ a_1 \cdot a_2 &= k_1 k_2 n^2 + k_1 b_2 n + b_1 k_2 n + b_1 b_2 \\ a_1 \cdot a_2 &= n \cdot (k_1 k_2 n + k_1 b_2 + b_1 k_2) + b_1 b_2 \end{aligned}$$

Definiert man nun $k' = k_1 k_2 n \cdot k_1 b_2 + b_1 k_2$ erhält man:

$$a_1 \cdot a_2 = k' \cdot n + b_1 b_2$$

Also ist:

$$a_1 \cdot a_2 \bmod n = b_1 \cdot b_2 \bmod n$$

Da $b_1 \cdot b_2$ größer als n sein könnte muss man $b_1 \cdot b_2$ noch modulo n nehmen. q.e.d.

Beweis zu Regel 2

Aus $a \bmod n = b \bmod n$ folgt

$$a^k \bmod n = b^k \bmod n.$$

Diese Regel kann man leicht einsehen, sie ergibt sich durch mehrfaches hintereinander ausführen der Regel 1.

Zu Beweis des Satzes von Euler:

Es seien m_1; m_2; m_3; ... ; $m_{\varphi(n)}$ die $\varphi(n)$ verschiedenen Zahlen, die zu n teilerfremd sind. Weiterhin seien die Zahlen r_1; r_2; r_3; ...; $r_{\varphi(n)}$ die Reste bei Division von m_1; m_2; m_3; ...; $m_{\varphi(n)}$ durch n. Das heißt es gilt $r_i = m \cdot m_i \bmod n$ für $i = 1, 2, \ldots, \varphi(n)$.

Die Zahlen r_i sind paarweise verschieden: $r_i \neq r_j$ für $i \neq j$.

Es ist $r_i < n$.

Angenommen, es gibt ein $i \neq j$ mit $r_i = r_j$.

Dann gilt: $m \cdot m_i \bmod n = m \cdot m_j \bmod n$ und daraus würde $m_i = m_j$ folgen. Dies steht allerdings im Widerspruch zur Voraussetzung des Satzes von Euler, dass die Zahlen m_1; m_2; m_3; ...; $m_{\varphi(n)}$, also auch m_i und $m_j$verschieden sind.

Außerdem gilt für alle i, dass die Zahlen r_i teilerfremd zu n sind (d.h. ggT(r_i, n) = 1). Denn, würde es einen gemeinsamen Teiler > 1 von r_i und n geben, dann würde dieser Teiler auch m_i teilen, was aber im Widerspruch zur Voraussetzung steht, dass n eine Primzahl ist.

Daraus folgt weiter, dass es sich bei der Zahlenreihe r_1; r_2; r_3; ...; $r_{\varphi(n)}$ um eine Permutation der Zahlen m_1; m_2; m_3; ...; $m_{\varphi(n)}$ - also um dieselben Zahlen, gegebenenfalls in einer anderen Reihenfolge - handelt.

Daraus folgt, dass

$$m_1 \cdot m_2 \cdot \ldots \cdot m_{\varphi(n)} = r_1 \cdot r_2 \cdot \ldots \cdot r_{\varphi(n)} \qquad (*)$$

Aus der Definition der r_i und den Rechenregeln im Modulrechnen folgt

$$\begin{aligned} & r_1 \cdot r_2 \cdot \ldots \cdot r_{\varphi(n)} \bmod n \\ = \; & (mm_1) \cdot (mm_2) \cdot \ldots \cdot (mm_{\varphi(n)}) \bmod n \\ = \; & m\varphi(n) \cdot m_1 \cdot m_2 \cdot \ldots \cdot m_{\varphi(n)} \bmod n \end{aligned}$$

Mit $(*)$ folgt daraus $1 = m^{\varphi(n)} \bmod n$
q.e.d.

Zu Beweis des RSA – Algorithmus:

(e, n) sei das öffentliche,
(d, n) das private Schlüsselpaar.
m sei die zu verschlüsselnde Nachricht.

Es muss $m \leq n$ gelten.

Zu zeigen ist, dass beim Entschlüsseln wieder der Klartext erzeugt wird:

$$m = m'$$

Für den Beweis wird folgender Satz benötigt:

Satz

Es seien p, $q \in \mathbb{N}$ zwei Primzahlen. Weiterhin sei $k \in \mathbb{N}$.
Für die natürliche Zahl m gelte $m \leq p \cdot q$.
Dann folgt:

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

Beweis:

Nach dem Satz von Euler gilt

$$m^{\varphi(n)} \bmod n = 1$$

Mit $n = p \cdot q$ ist

$$\begin{aligned} & m^{\varphi(p\cdot q} \bmod (p \cdot q) = 1. \\ \Rightarrow\ & m^{(p-1)\cdot(q-1)} \bmod (p \cdot q) = 1 \\ \Rightarrow\ & m^{k\cdot(p-1)\cdot(q-1)} \bmod (p \cdot q) = 1 \qquad (**) \end{aligned}$$

Der letzte Schritt ergibt sich direkt aus den Rechenregeln im Modulrechnen.
Nach Multiplikation von $(**)$ mit m folgt

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

q.e.d.

Beim RSA–Algorithmus gilt

$$\begin{aligned} m' &= c^d \bmod n \\ \text{und} \quad c &= m^e \bmod n \end{aligned}$$

$$\begin{aligned} \Rightarrow m' &= (m^e \bmod n)^d \bmod n \\ &= (me \cdot d \bmod n) \end{aligned}$$

Des Weiteren gilt im RSA–Algorithmus $(d \cdot e) \bmod \varphi(n) = 1$, daraus folgt

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

$$\Rightarrow m' = m^{k\cdot\varphi(n)+1} \bmod n$$

Für$\varphi(n)$ gilt $\varphi(p \cdot q) = (p-1) \cdot (q-1)$, da $n = p \cdot q$ und p und q zwei Primzahlen sind:

$$\Rightarrow m' = m^{k\cdot(p-1)\cdot(q-1)+1} \bmod p \cdot q$$

Nach Satz 3.1 gilt

$$m^{k\cdot(p-1)\cdot(q-1)+1} \bmod (p \cdot q) = m.$$

$\Rightarrow m' = m$

q.e.d.