Johann-Mathesius-Gymnasium
Rochlitz

Schuljahr 2017/2018

Komplexe Leistung

Thema : Darstellung der Funktionsweise von Verschlüsselungsverfahren

Fach : Mathematik

Verfasser : Moritz Enge

Betreuender Fachlehrer : Herr Fritz

Abgabetermin : 27.Oktober 2017

Inhaltsverzeichnis

Komplexe Leistung Moritz Enge 11CH1

1. Einleitung

Eine der wichtigsten Errungenschaft der Sicherheit heutzutage ist die Verschlüsselung.
Ohne sie wäre eine erfolgreiche digitale Existenz unmöglich. Ob bei dem Übertragen wichtiger Staatsgeheimnisse oder dem Sichern der Firmendaten gegen Unbefugte, überall wird sie angewandt.
Die ersten bekannten Verschlüsselungmethoden wurden ca 1900 vor Christus von den Ägyptern „erfunden“. Zu dieser Zeit wurden die Zeichen des Klartextes durch Hieroglyphen ersetzt.Ob vor den Ägyptern es schon andere Kryptoverfahren gab, ist nicht bekannt.
Im Laufe der Zeit vebesserten sich die Verfahren und die Sicherheit und somit wurde im 20. Jahrhundert von Arthur Scherbius die ENIGMA entwickelt. Sie wurde zur sicheren Datenübertragung im dritten Reich benutzt , zumindestens so lange bis der britische Logiker, Mathematiker, Kryptoanalytiker und Informatiker Alan Turing eine effizente Methode fand . Die mit ENIGMA verschlüsselten Narichten zu entschlüsseln
Die Verschlüsselung lässt sich in drei Teile unterteilen, erstens das Erzeugen eines Schlüssels, dies kann durch das „einfache“ auswählen einer beliebigen Zahl oder durch komplexe mathematische Verfahren geschehen. Danach folgt das Verschlüsseln (Encoding) mit dem davor generierten Schlüssel, bei diesem Schritt erhält man aus dem Klartext den Geheimtext, das sogenannte Chiffrat. Um aus dem Geheimtext wieder den Klartext zu erzeugen benötigt man den letzten Schritt, das Entschlüsseln(Decoding), dazu wird der Geheimtext und ein Entschlüsselungs-Schlüssel benötigt. Je nachdem welches Verfahren angewandt wird, ist der Schlüssel zum Verschlüsseln und Entschlüsseln gleich oder unterschiedlich.Anhand diesen letzten Schrittes kann man die Verschlüsselungsverfahren klassifizieren.Es gibt die symmetrische, die asymmetrische und die hybride Verschlüsselung, auf welche später eingegangen wird.

2. Mathematische Grundlagen

2.1 Modulo-Division

Die Modulo-Division ist wie das, schon aus der Grundschule bekannte Teilen mit Rest.
Es wird aber nur der Rest betrachtet. Man kann Modulo wie jede andere Rechenart verwenden. Als Zeichen verwendet man „mod“, „modulo“ oder „%“.

Beispiel :
20 / 3 = 6 Rest **2** , das heißt 20 mod 3 = **2**

2.2 Primzahlen

Primzahlen sind natürliche Zahlen größer 1, welche nur durch sich selbst und 1 teilbar sind. Die Menge aller Primzahlen trägt das Symbol $\mathbb{P}$. Sie spielen in der Verschlüsselung eine große Rolle, da die Primfaktorzerlegung des Produktes zweier großer Primzahlen, heutzutage immer noch ein großes mathematisches Problem ist. Dieses Produkt nimmt in der Realität bis zu mehrere tausend Stellen an. Damit lassen sich die beiden Primzahlen in Lebenszeit nicht bestimmen.

2.3 Eulersche φ-Funktion

Die Eulersche φ-Funktion φ(n) gibt an, wie viele teilerfremde Zahlen es zu n gibt.

Allgemein gilt : $\varphi(n * m) = \varphi(n) * \varphi(m)$

Für Primzahlen ($n \in \mathbb{P}$) gilt : $\varphi(n) = n - 1$

Beispiele : $\varphi(11) = 11-1 =$ **10** , 11 ist prim

$\varphi(77) = \varphi(7*11) = \varphi(7) * \varphi(11) = 6 * 10 =$ **60** , 7 und 11 sind prim

2.4 Teilerfremdheit zweier Zahlen

Zwei Zahlen sind teilerfremd, wenn sie keinen gemeinsamen Primteiler (Teiler ist eine Primzahl) haben .

Die Zahlen 12 und 77 sind teilerfremd, denn ihre Primfaktorzerlegungen $12= 2\cdot2\cdot3$ und

$77= 7\cdot11$ enthalten keine gemeinsamen Primfaktoren.(1)

2.5 Multiplikatives Inverses bezüglich mod n

Das multiplikative Inverse bezüglich mod n, lässt sie durch den erweiterten euklidischer Algorithmus berechnen . Dies ist ein Algorithmus, der neben dem größten gemeinsamen Teiler von den natürlichen Zahlen a und b auch noch zwei natürliche Zahlen m und n berechnet, für die gilt:

$ggT(a,b) = a * m + b * n$

$ggT(a,n) = 1$

Wobei die natürliche Zahl m das multiplikative Inverse bezüglich mod n darstellt.

3. Arten der Verschlüsselung

3.1 Symmetrische Verschlüsselung

Die symmetrische Verschlüsselung ist ein Teilgebiet der Verschlüsselung, bei welchem beide Teilnehmer meistens denselben Schlüssel verwenden, d. h. $Schlüssel_{Klartext} = Schlüssel_{Geheimtext}$, oder sich die beiden Schlüssel leicht auseinander berechnen lassen.Man teilt die symmetrische Verschlüsselung nochmal in Block- und Stromchiffren auf.Bei Stromchiffren wird der Klar - bzw. Geheimtext Zeichen für Zeichen ver- bzw. entschlüsselt (2). Ein Beispiel für diese Verschlüsselung ist die Vigenère – Chiffre.Eine Blockchiffre arbeitet mit einer festen Blockgröße und ver- bzw. entschlüsselt mehrere Zeichen auf einmal. Die Größe der Blockgröße wird hinter den Namen des Algorithmus geschrieben, zum Beispiel AES 256 (Advanced Encryption Standard) .Der Nachteil der symmetrischen Verschlüsselung ist der Transport des Schlüssels zum Gegenüber. Da dieser ja im Klartext transportiert werden müsste und jeder „Angreifer“ diesen Schlüssel abfangen könnte und so der Geheimtext nicht mehr gegenüber dritten geheim ist . Man könnte den Schlüssel auch wieder symmetrisch verschlüsseln, aber da müsste ja der Schlüssel zum Entziffern des Schlüssels irgendwie übertragen werden. Dieses Problem kann man mit Hilfe der hybriden Verschlüsselung lösen (siehe 3.3).

Vorteil	Nachteil
• Einfaches Schlüsselmanagement, da nur ein Schlüssel • Relativ schnelle Ver- und Entschlüsselung	• Schlüssel darf nicht zu Unbefugten gelangen • Sicherer Übertragungsweg

3.2 Asymmetrische Verschlüsselung

Die asymmetrische Verschlüsselung ist ein Verschlüsselungsverfahren, bei dem es einen öffentlichen und einen privaten Schlüssel gibt . Mit dem öffentlichen Schlüssel verschlüsselt man den Klartext zum Geheimtext. Aus dem Geheimtext und dem öffentlichen Schlüssel kann man nicht den Klartext entschlüsseln. Mit dem privaten Schlüssel kann man aus dem Geheimtext den Klartext berechnen. Die asymmetrische Verschlüsselung ist erst seit Beginn der 1970er bekannt (3).

Die asymmetrische Verschlüsselung besteht aus mindestens drei Algorithmen :

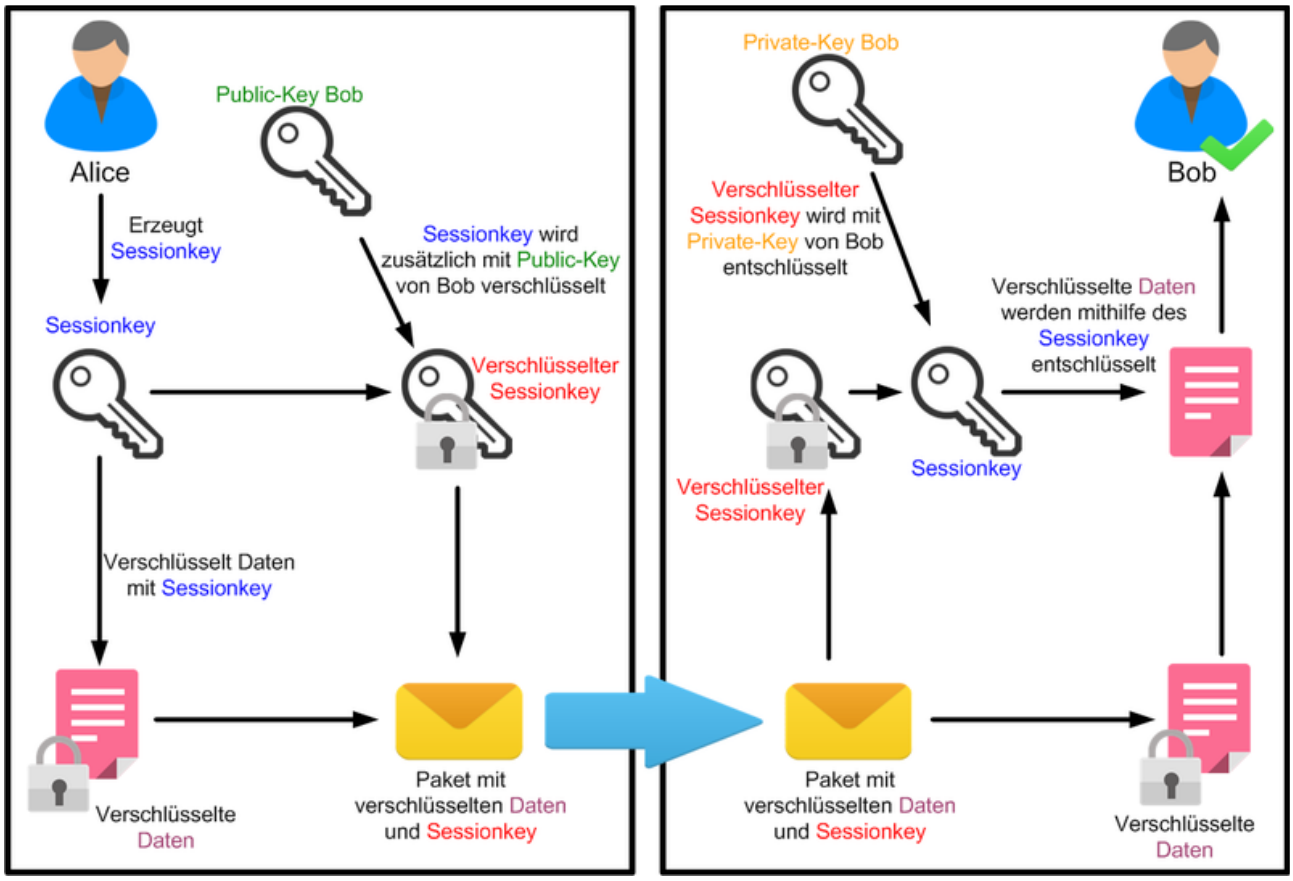
1. Algorithmus zur Schlüsselerzeugung
2. Verschlüsselungsalgorithmus
3. Entschlüsselungsalgorithmus

Dadurch, dass es einen öffentlichen Schlüssel gibt , existiert kein Übertragungsproblem wie bei der symmetrischen Verschlüsselung. Da jeder den öffentlichen Schlüssel wissen kann, da er damit aus dem Geheimtext keinen Klartext entschlüsseln kann.

Vorteil	Nachteil
• Hohe Sicherheit • kein Übertragungsproblem	• Relativ Langsame Ver- und Entschlüsselung • Für Sicherheit werden große Schlüssel benötigt

3.3 Hybride Verschlüsselungsverfahren

Das hybride Verschlüsselungsverfahren ist eine Kombination von der symmetrischen Verschlüsselung und der asymmetrischen Verschlüsselung. Bei dieser Verschlüsselung wird der Klartext symmetrisch verschlüsselt , aber der Schlüssel für die symmetrische Verschlüsselung asymmetrisch verschlüsselt übertragen. Man nimmt symmetrische Verschlüsselung zur Verschlüsselung der Daten, da diese auch bei großen Datenmengen sehr schnell sind (Diese sind meistens der Klartext). Asymmetrische Verschlüsselungsverfahren sind sehr langsam und daher nur geeignet für kleine Datenmengen (Der Schlüssel für den Klartext).



Komplexe Leistung Moritz Enge 11CH1

4. Vorstellung spezieller Verschlüsselungsverfahren

4.1 Vigenère-Chriffe

4.1.1 Allgemeines

Die Vigenère-Chriffe ist eine Stromchiffre aus dem 16. Jahrhundert. Der Klartext wird in Einzelzeichen zerlegt und durch Geheimzeichen substituiert , die Mithilfe eines Schlüssels aus dem Vigenère-Quadrat ausgewählt werden. Bei diesem Vigenère-Qudarat handelt es sich um eine quadratische Anordnung von untereinander stehenden verschobenen Alphabeten (4)

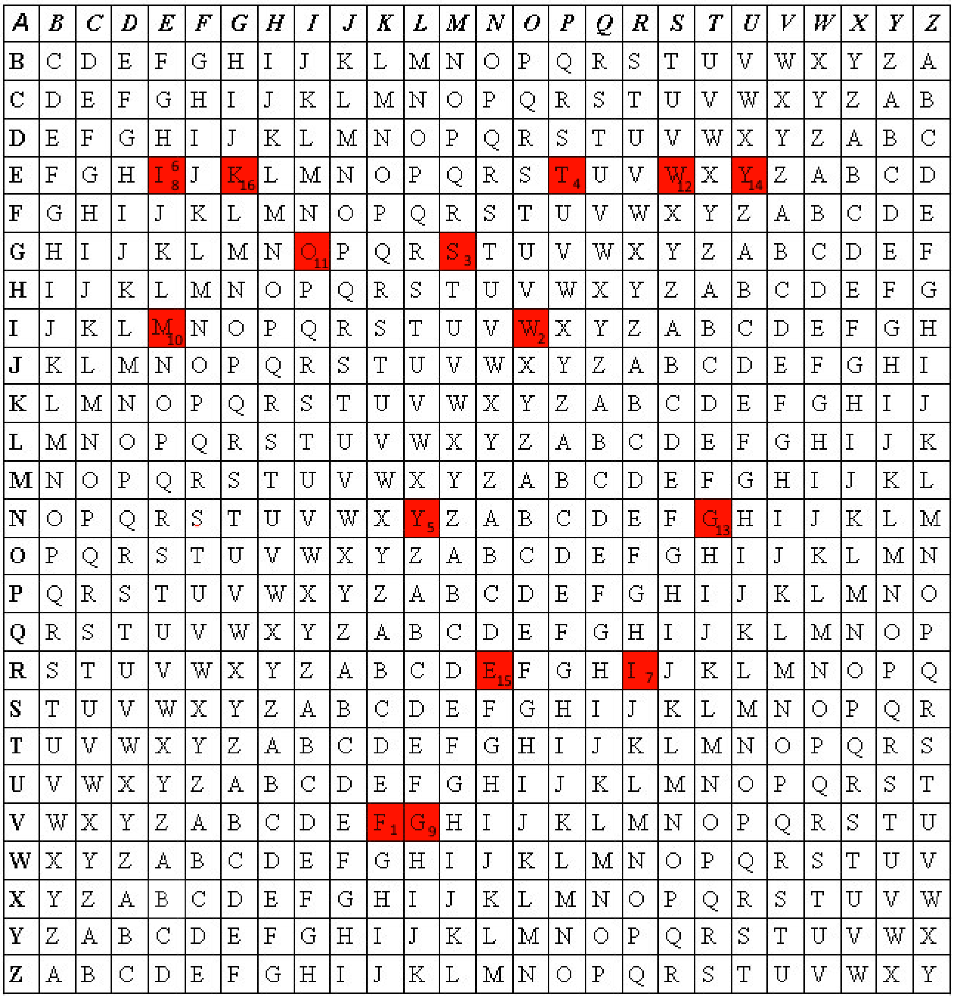
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
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
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
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
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
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
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
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
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
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
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
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
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
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
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
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
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
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
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
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
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

4.1.2 Funktionsweise

Als Erstes wird das Vigenère-Quadrat (siehe 4.1.1) und ein Schlüssel benötigt, der Schlüssel soll möglichst lang und zufällig sein. Ist die Länge des Schlüssels = Länge des Klartextes, dann wird der Schlüssel nicht mehrfach verwendet und man erhält eine besondere Version der Vigenère-Chriffe, welche „One-Time-Pad“ genannt wird. Die entsprechenden Geheimtextbuchstaben kann man nun leicht mithilfe des Vigenère-Quadrats ermitteln. Dazu wird der Kreuzungspunkt gesucht, der durch den jeweiligen Schlüsselbuchstaben gekennzeichneten Zeile und der Spalte des Quadrats, die oben durch den Klartextbuchstaben gekennzeichnet ist (5).

4.1.3 Beispiel

Schlüssel	V	I	G	E	N	E	R	E	V	I	G	E	N	E	R	E
Klartext	K	O	M	P	L	E	X	E	L	E	I	S	T	U	N	G
Geheimtext	F	W	S	T	Y	I	I	I	G	M	O	W	G	Y	E	K



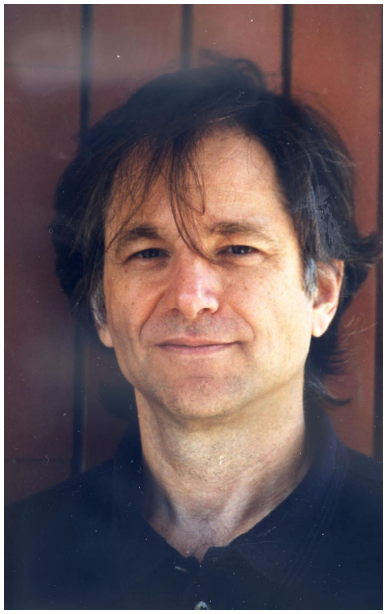
A	***B***	***C***	***D***	***E***	***F***	***G***	***H***	***I***	***J***	***K***	***L***	***M***	***N***	***O***	***P***	***Q***	***R***	***S***	***T***	***U***	***V***	***W***	***X***	***Y***	***Z***
B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
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
E	F	G	H	I$^{6}_{8}$	J	K$_{16}$	L	M	N	O	P	Q	R	S	T$_{4}$	U	V	W$_{12}$	X	Y$_{14}$	Z	A	B	C	D
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
G	H	I	J	K	L	M	N	O$_{11}$	P	Q	R	S$_{3}$	T	U	V	W	X	Y	Z	A	B	C	D	E	F
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
I	J	K	L	M$_{10}$	N	O	P	Q	R	S	T	U	V	W$_{2}$	X	Y	Z	A	B	C	D	E	F	G	H
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
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
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
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
N	O	P	Q	R	S	T	U	V	W	X	Y$_{5}$	Z	A	B	C	D	E	F	G$_{13}$	H	I	J	K	L	M
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
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
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
R	S	T	U	V	W	X	Y	Z	A	B	C	D	E$_{15}$	F	G	H	I$_{7}$	J	K	L	M	N	O	P	Q
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
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
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
V	W	X	Y	Z	A	B	C	D	E	F$_{1}$	G$_{9}$	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
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
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
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
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

4.2. RSA (Rivest – Shamir – Aldeman)

4.2.1 Allgemeines

Das RSA- Verfahren gilt heutzutage als ein der sichersten Verschlüsselungsalgorithmen der Welt , obwohl es das erste veröffentlichte assymetrische Verschlüsselungsverfahren ist. Der Name „RSA“ geht auf die drei Erfinder Ronald Linn **R**ivest , Adi **S**hamir und Leonard Max **A**dleman zurück. Dieses Verfahren wird den asymetrischen Verschlüsselungverfahren zugeordnet . Der private Schlüssel kann in lebenszeit nicht aus dem öffentlichen Schlüssel berechnet werden.

Ronald L.. Rivest



Leonard M. Adleman

Adi Shamir

4.2.2 Funktionsweise

Erzeugung der öffentlichen und privaten Schlüssel

Als erstes werden zwei unterschiedliche unabhängige Primzahlen **p** und **q** gesucht, welche ungefähr die gleiche Größenordnung haben, aber nicht zu nah aneinander liegen.In der Praxis werden große Zahlen generiert und durch einen Primzahltest so lange geprüft und neu generiert, bis man zwei Primzahlen hat.

Danach wird das RSA-Modul N berechnet.

Dabei gilt : **N = p * q**

Als nächstes wird das Ergebnis der eulerschen φ-Funktion von Berechnent. Da N ein Produkt von zwei Primzahlen ist gilt:

φ(N) = (p - 1) * (q -1)

Als vorletzes wird eine zu φ(N) teilerfremde Zahl **e** gesucht, es gilt $1 < e < \varphi(N)$, meistens wird aus Effizientgründen eine relativ kleine Zahl gewählt,

üblich ist die 5.Fermat-Zahl : 2^16 +1 (65537).

Als letzes wird der Entschlüsselungsexponent **d** berechnet, dieser ist das Multiplikative Inverse von e bezüglich des Moduls φ(N), das heist :

$e * d \equiv 1 \bmod \varphi(N)$

Sind diese ganzen Zahlen berechnet werden p, q und φ(N) gelöscht, da diese nicht mehr benötigt werden und aus Sicherheitsgründen vernichtet werden. Die Wahl zu kleiner Zahlen bei p, q und e führt dazu, dass das Verfahren relativ schnell und effizient geknackt werden kann.

Nach der Berechnung der Zahlen gibt es den öffentlichen Schlüssel **öffentlich_key**, welcher sich aus **e** und **N** zusammensetzt : öffentlich_key(e,N)

Außerdem existiert nun der private Schlüssel **privat_key**, welcher sich aus d und N zusammensetzt : privat_key(d,N)

Verschlüsseln von Narichten

Um die Naricht m zu verschlüsseln, wird nach einer Formel der Geheimtext c berechnet, dazu wird der öffentliche Schlüssel benötigt, somit kann jeder eine Naricht verschlüsseln

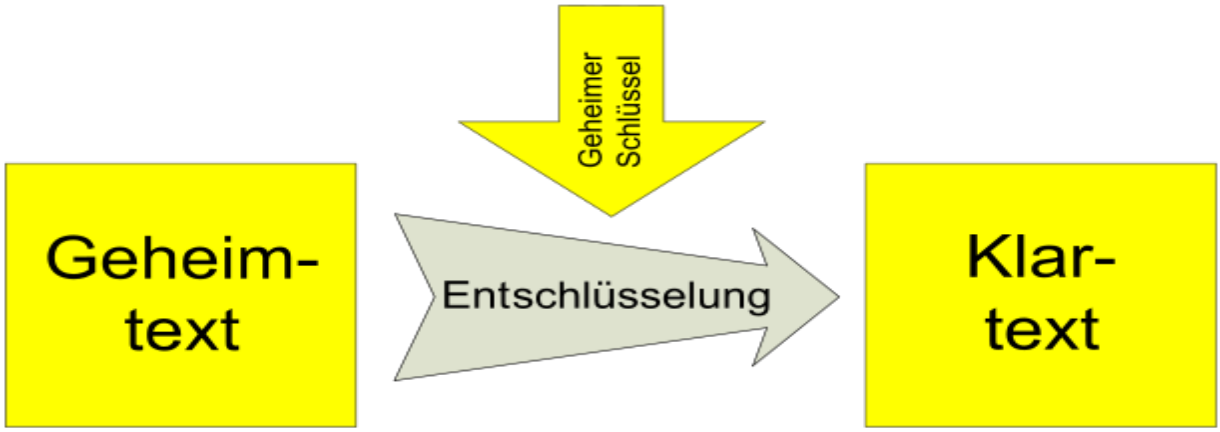
$c \equiv m^{e} \bmod N$

Die Zahl m muss außerdem kleiner sein als RSA-Modul N.

Entschlüsseln von Narichten

Um aus dem Geheimtext c wieder die Naricht m zu berechen wendet man folgende Formel an, außerdem wird dazu noch der private Schlüssel benötigt, so kann nur der „Ersteller der Schlüssel“ die Naricht auch wieder entschlüsseln

$m \equiv c^{d} \bmod N$



4.2.3 Beispiel

„zufälliges“ p : 11 (aus rechnerischen Gründen klein gehalten)

„zufälliges“ q : 17 (aus rechnerischen Gründen klein gehalten)

N = p * q = 11 * 17 = 187

ϕ(N) = 11-1 * 17-1 = 10 * 16 = 160

e = 23 (aus rechnerischen Gründen klein gehalten)

e * d ≡ 1 mod ϕ(N) => d = 7

==> öffentlicher Schlüssel (23 , 187) und privater Schlüssel (7 ,187)

Klartext : RSA

Klartext in ASCII-Zahlen : 82 83 65

Verschlüsseln : 82^23 mod 187 = 125 ==> }

83^23 mod 187 = 161 ==> – nicht darstellbar –

65^23 mod 187 = 142 ==> – nicht darstellbar –

Geheimtext in Zahlen : 125 161 142

Entschlüsseln : 125^7 mod 187 = 82 ==> R

161^7 mod 187 = 83 ==> S

142^7 mod 187 = 65 ==> A

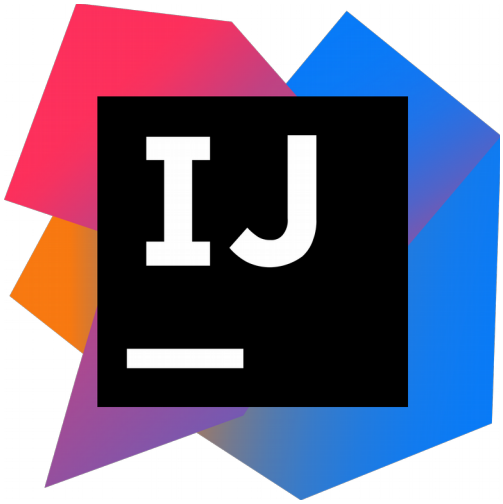
5. Eigenleistung

In meiner Eigenleistung hab eich eine Bibliothek für drei ausgewählte Verschlüsselungsalgorithmen programmiert. Passend dazu wurde auch eine graphische Visualisierung erstellt, dass sogenannte GUI (Graphical User Interface) . Zu den drei Algorithmen zählt die Cäsar Verschlüsselung, die Vigenère Chiffre und der RSA Algorithmus.

Die Bibliothek und das GUI wurden in der imperativ, objektorientierten High Level Sprache Java geschrieben. Ich hab Java als Programmiersprache gewählt, da sie Plattform unabhängig ist, das heißt man kann das Programm auf jedem Betriebssystem ausführen, vorausgesetzt die JVM ist installiert. Die JVM ist die „Java Virtual Machine“ welche zur Ausführung von Java-Programmen benötigt wird. Diese JVM kann man für jedes Betriebssystem auf der Seite von „Oracle“ unter dem Name „JRE“ herrunterladen und installieren . JRE steht für „Java Runtime Enviroment“, dieses beinhaltet unteranderem die JVM.

Zur Realisierung des GUI´s wurde das Java interne Toolkit „JavaFX“ verwendent.

Zur Besseren Programmierung wurde die Entwicklungsumgebung „Intelli J“ von JetBrains verwendet



Komplexe Leistung Moritz Enge 11CH1

Die Bibliothek besteht aus fünf Dateien :

- Caesar.java
- Vigenere.java
- RSA.java
- RSAKey.java
- Algorithmen.java

In den Dateien Caesar.java, Vigenere.java und RSA.java wurden die Ver- und Entschlüsselungsmethoden so wie gegebenenfalls benötigte Hilfsmethoden implementiert

Die Datei RSAKey.java beschreibt eine Klasse, mit welcher der Umgang mit den den oben gennanten RSA-Schlüssel einer zu realisieren ist.

Die GUI mit Hilfe von zwei Dateien erstellt :

- GUI.java
- GUI.fxml

Die Datei GUI.fxml beschreibt anhand einer auf XML-basierender Sprache das Aussehen der GUI. Die Funktionalität der GUI wurde innerhalb der Datei GUI.java unter Hilfe der Bibliothek implementiert.

Der Quelltext(Sourcecode) befindet sich in den Anlagen.

6.Fazit

7.Anhang