Wie sicher sind heutige Verschlüsselungsmethoden am Beispiel der RSA-Verschlüsselung?

Jannis Hajda

Seminararbeit im Seminarkurs „Mathematik im Alltag“
betreut durch R. Prüter

Hedwig Bollhagen Gymnasium
Emma-Ihrer-Straße 7B, 16727 Velten

Schuljahr 2019/2020

Inhaltsverzeichnis

1 Vorwort

1.1 Motivation

Schon früh in der Geschichte der Menschheit spielte die Verschlüsselung von Informationen und Nachrichten eine entscheidende Rolle, doch besonders im Zeitalter des Internets ist sie von essenzieller Bedeutung für den sicheren Datenverkehr. Online Banking, E-Mail, Bezahl- und Handelssysteme – all dies wäre ohne die Kryptografie in ihrer heutigen Form undenkbar. Man stellt also fest, dass Verschlüsselung nicht nur für einige wenige, sondern vielmehr für jeden Einzelnen von uns einen unfassbaren Wert hat. Eines der bedeutendsten kryptologischen Verfahren unserer Zeit stellt hierbei das RSA-Verfahren dar. Aus diesem Grund habe ich mich dazu entschieden, mich in der vorliegenden Arbeit genauer mit diesem Verfahren auseinanderzusetzen. Ziel ist es, die mathematischen Grundlagen zu verstehen, eventuelle Sicherheitslücken zu erläutern und einen Blick in die Zukunft zu wagen, um am Ende die Frage nach der Sicherheit heutiger Verschlüsselungsmethoden beantworten zu können.

1.2 Das Problem symmetrischer Verschlüsselungsverfahren

Symmetrische Verschlüsselungsverfahren, auch Secret-Key-Verfahren, sind kryptologische Verfahren, die zur Ver- und Entschlüsselung von Daten denselben Schlüssel, bzw. einen leicht zu berechnenden Schlüssel verwenden und somit im direkten Kontrast zu den asymmetrischen Verschlüsselungs-, auch Public-Key-Verfahren, stehen. Die meisten Secret-Key-Verfahren sind sehr ressourcenschonend entwickelt und weisen somit einen geringen Energieverbrauch auf. Anders als die Public-Key-Verfahren sind sie jedoch in Zeiten des Internets nicht dazu in der Lage, eine sichere und verschlüsselte Kommunikation zwischen zwei beliebigen Gesprächspartnern zu ermöglichen, da eine sichere Übertragung des Schlüssels bei diesem Verfahren von essenzieller Bedeutung ist und nicht ohne einen physischen Kontakt der beiden Parteien gewährleistet werden kann. Somit ist ein verschlüsselter Datenaustausch zwischen zwei sich unbekannten Personen in der Realität durch symmetrische Verschlüsselungsverfahren nicht umsetzbar.

1.3 Asymmetrische Verschlüsselung als Lösung des Problems der Schlüsselübertragung

Anders als die symmetrischen Verschlüsselungsverfahren basieren die asymmetrischen Verschlüsselungsverfahren nicht auf dem Prinzip der Verwendung desselben Schlüssels zur Ver- und Entschlüsselung von Daten. Stattdessen besitzt jeder Kommunikationspartner zwei Schlüssel – einen Öffentlichen und einen Privaten (auch Public- und Private-Key genannt). Möchte nun Partner A eine verschlüsselte Nachricht an Partner B übertragen, so muss er lediglich über dessen öffentlichen Schlüssel verfügen, welcher zur Verschlüsselung der Nachricht u. o. der Überprüfung seiner digitalen Signatur dient. Möchte Partner B nun die übermittelte Nachricht entschlüsseln, so benötigt er seinen privaten Schlüssel. Dieser ermöglicht es ihm, die Nachricht zu entschlüsseln u. o. digitale Signaturen zu erzeugen. Durch mathematische Verfahren ist nur dieser private Schlüssel, welcher sich mit der aktuellen Rechenkapazität von Hochleistungscomputern nicht in absehbarer Zeit aus dem öffentlichen Schlüssel berechnen lässt, in der Lage, die verschlüsselte Nachricht in einen Klartext umzuwandeln. Folglich sind diese Public-Key-Verfahren für eine sichere Kommunikation im Internet von unfassbarem Wert.

2 Begriffsklärungen und mathematische Grundlagen

2.1 Kerckhoffs'sche Prinzip

Das Kerckhoffs'sche Prinzip ist ein 1883 vom niederländischen Linguist und Kryptologen Auguste Kerckhoffs aufgestellter Grundsatz der Kryptografie, welcher besagt, dass die Sicherheit eines kryptologischen Verfahrens nicht auf der Geheimhaltung des Algorithmus, sondern nur auf der Geheimhaltung der verwendeten Schlüssel beruhen sollte. Durch eine Veröffentlichung des Algorithmus wird die Sicherheit des Verschlüsselungsverfahrens garantiert, da eventuelle Sicherheitslücken durch eine ständige Weiterentwicklung und Kontrolle des Verfahrens seitens Dritter entdeckt und so behoben werden können.

2.2 Teilbarkeit

Definition 1. *Für ganze Zahlen $x, y \in \mathbb{Z}$ definiert man*

$$x \mid y \qquad \text{(gesprochen: } x \text{ teilt } y\text{)}$$

genau dann, wenn eine ganze Zahl $q \in \mathbb{Z}$ existiert mit $y = qx$

2.3 Euklidischer Algorithmus

Der Euklidische Algorithmus ist eine Methode, um den größten gemeinsamen Teiler zweier ganzen Zahlen x und y zu berechnen. Wir setzen $x_0 := x$ und $x_1 := y$ und führen nach folgendem Schema sukzessive Teilungen mit Rest durch, bis sich der Rest 0 ergibt

$$\begin{cases} x_0 = q_1 x_1 + x_2, & 0 < x_2 < x_1, \\ x_1 = q_2 x_2 + x_3, & 0 < x_3 < x_2, \\ \ldots & \\ x_{n-2} = q_{n-1} x_{n-1} + x, & 0 < x_n < x_{n-1}, \\ x_{n-1} = q_n x_n + 0 & \end{cases} \tag{1}$$

Die Reihe der Zahlen $x_1, x_2, \ldots$ nimmt streng monoton ab, folglich wird nach endlich vielen Schritten der Rest 0 erreicht.

Behauptung 1. *Der letzte Divisor $d := x_n$ ist der größte gemeinsame Teiler von $x = x_0$ und $y = x_1$*

Beweis. (1) d ist ein gemeinsamer Teiler von x und y
Wir betrachten die Gleichungen von (1) der Reihe nach von der letzten zur ersten. Aus der letzten Gleichung folgt, dass $d = x_n$ ein Teiler von x_{n-1} ist. Da also d ein Teiler von x_n und x_{n-1} ist, folgt aus der vorletzten Gleichung, dass d auch x_{n-2} teilt. So fortfahrend erhält man, dass d alle $x_k, k = n, n-1, \ldots, 1, 0$ teilt. Insbesondere ist d ein gemeinsamer Teiler von $x_0 = x$ und $x_1 = y$.

(2) Jeder Teiler d' von $x = x_0$ und $y = x_1$ ist auch ein Teiler von $d = x_n$
Hierzu Betrachten wir die Gleichungen von (1) von der ersten bis zur letzten. Aus der ersten Gleichung folgt per Definition der Teilbarkeit zunächst, dass d' auch ein Teiler von x_2 ist. Da also d' ein gemeinsamer Teiler von x_1 und x_2 ist, folgt aus der zweiten Gleichung, dass d' auch ein Teiler von x_3 ist, usw. Schließlich erhält man, dass d' auch ein Teiler von x_n ist. □

Beispiel 1. *Bestimmung des ggT von* 768 *und* 564

$$\begin{aligned} 768/564 &= 1 && \textit{Rest } 204, \\ 564/204 &= 2 && \textit{Rest } 156, \\ 204/156 &= 1 && \textit{Rest } 48, \\ 156/48 &= 3 && \textit{Rest } 12, \\ 48/12 &= 4 && \textit{Rest } 0 \end{aligned}$$

Ergebnis: Der ggT von 768 *und* 564 *ist* 12.

2.4 Modulo-Funktion

Definition 2. *Will man eine natürliche Zahl a durch eine natürliche Zahl m teilen, so erhält man einen Rest r. Für diesen Rest gilt $0 \leq r \leq m-1$. Die Modulo-Funktion liefert zu gegebenen Zahlen a und m gerade diesen Rest r. Man schreibt auch*

$$a \mod m = r$$

2.5 Das multiplikative Inverse modulo einer Zahl m

Definition 3. *Sind a und m zwei teilerfremde positive ganze Zahlen, so ist die multiplikative Inverse von a modulo m diejenige (eindeutig bestimmte) positive Zahl $b < m$, welche die Gleichung $1 \equiv ab \mod m$ erfüllt. Man schreibt auch $b = a^{-1} \mod m$.*

2.6 Lemma von Bézout

Satz 1. *Seien $x, y \in \mathbb{Z}$ und $d = ggT(x, y)$ ihr größter gemeinsamer Teiler. Dann gibt es Zahlen $\lambda, \mu \in \mathbb{Z}$, so dass sich der ggT als ganzzahlige Linearkombination aus λ und μ darstellen lassen kann*

$$d = \lambda x + \mu y.$$

Diese Gleichung ist nicht eindeutig, d.h., dass sie durch verschiedene Zahlenpaare λ, μ erfüllt wird.

2.7 Bestimmung vom multiplikativen Inversen a modulo m durch den erweiterten euklidischen Algorithmus

Zur Bestimmung des multiplikativen Inversen a modulo m mit $ggT(a, m) = 1$ benutzt man häufig den erweiterten euklidischen Algorithmus. Er liefert als Ergebnis jene eindeutig bestimmte positive Zahl $b < m$, welche die Gleichung

$$ab \mod m = 1$$

erfüllt. Seine Funktionsweise wird gut anhand eines Beispiels erkennbar.

Beispiel 2. *Bestimmung des multiplikativen Inversen von* 5 *modulo* 48[1]

$$\begin{aligned} 48 &= 9 * 5 + 3 \\ 5 &= 1 * 3 + 2 \\ 3 &= 1 * 2 + 1 \\ 2 &= 2 * 1 + 0 \end{aligned}$$

$$\begin{aligned} 1 &= 3 - 1 * 2 \\ &= 3 - 1 * (5 - 1 * 3) = 2 * 3 - 1 * 5 \\ &= 2 * (48 - 9 * 5) - 1 * 5 \\ &= 2 * 48 - 19 * 5 \end{aligned}$$

$$1 = -19 * 5 \mod 48 \qquad \textit{Addition der linken Seite mit 48*5}$$

$$1 = 29 * 5 \mod 48$$

Das multiplikative Inverse von 5 modulo 48 lautet 29. Man arbeitet also zunächst den euklidischen Algorithmus ab, um anschließend durch Substitution den Satz 1 zu lösen.

2.8 Satz von Euler-Fermat

Satz 2. *Sind a und n teilerfremde Zahlen mit 1 < a < n, so gilt*

$$a^{\phi(n)} \equiv 1 \mod n.$$

Hierbei ist $\phi(n)$ die Anzahl der zu n teilerfremden Zahlen.

[1] Beispiel entnommen aus 7.

2.9 Einwegfunktionen/Falltürfunktionen

Definition 4. *Eine Einwegfunktion ist eine Abbildung f einer Menge X in eine Menge Y, so dass $f(x)$ für jedes Element von X leicht zu berechnen ist, während es für (fast) jedes y aus Y extrem schwer ist, ein Urbild x (d.h. ein x mit $f(x) = y$) zu finden. Eine spezielle Variante der Einwegfunktionen ist die Falltürfunktion, welche sich nur leicht umkehren lässt, wenn man im Besitz einer Zusatzinformation ist.*

3 Das RSA-Kryptosystem

3.1 Geschichtlicher Hintergrund

Die Idee der asymmetrischen Public-Key-Verfahren geht auf eine 1976 veröffentlichte Theorie der beiden US-Amerikanischen Kryptologen Diffie Whitefield und Martin Hellmann zurück, welche auf der Suche nach einer geeigneten Einwegfunktion scheiterten. Die beiden Informatiker Ron Rivest und Adi Shamir, sowie der Mathematiker Leonard Adleman, allesamt Studierende am MIT, nahmen sich der Aufgabe an und suchten über ein Jahr lang nach einer Einwegfunktion, welchen den Anforderungen Whitefileds und Hellmanns entsprach. Rivist und Shamir überlegten sich mögliche Funktionen, während Adleman damit beschäftigt war, diese mathematisch zu widerlegen. Zunächst glaubten sie, dass eine solche Einwegfunktion aufgrund der widersprüchlichen Anforderungen unmöglich umzusetzen sei, allerdings stieß Rivest schließlich 1977 auf ein Verfahren, das keine Schwächen aufwies. Noch im selben Jahr veröffentlichen sie das RSA-Verfahren, benannt nach ihren Initialen, und meldeten es 1983 zum Patent an, das am 21. September 2000 auslief.[2] Das RSA-Verfahren richtet sich also nach dem wie in 2.1 beschriebenen Kerckhoffs'sche Prinzip, da die Sicherheit des Verfahrens nur auf der Geheimhaltung der Schlüssel und nicht auf der Geheimhaltung des Algorithmus beruht.

3.2 Generierung der Schlüssel

Die Generierung der Schlüssel des RSA-Verfahrens läuft in fünf Schritten ab. Diese müssen systematisch abgearbeitet werden.

[2] Vgl. 6, S. 2.

1. Man wähle zwei sich in der gleichen Größenordnung befindende Primzahlen p und q mit $p \neq q$
2. Berechnung des RSA-Moduls $N = p * q$
3. Berechnung von $\phi(N) = (p-1) * (q-1)$
4. Wahl eines $e \in \mathbb{N}$ mit $1 < e < \phi(N)$ und $ggT(e, \phi(N)) = 1$
5. Berechnung von d als multiplikatives Inverses von e bezüglich des Moduls $\phi(N)$[3]

Das Tupel (e, N) bildet nun den öffentlichen Schlüssel, der mit jedem Kommunikationspartner problemlos ausgetauscht werden kann, während die Zahl d als privater Schlüssel fungiert. Er wird zusammen mit den beiden Primzahlen p und q sicher aufbewahrt. Diese können wahlweise auch vernichtet werden, dürfen jedoch wie auch d nicht an die Öffentlichkeit gelangen.

3.3 Ver- und Entschlüsselung von Nachrichten

Möchte man nun eine Nachricht $m \in \mathbb{N}$ für die Gegenseite verschlüsseln, so benötigt man wie in 3.2 erwähnt das öffentliche Tupel. (e, N). Mit Hilfe der relativ einfachen Formel

$$c = m^e \mod N, \tag{2}$$

wobei die Beziehung $m < N$ gelten muss, ergibt sich die verschlüsselte Nachricht c, welche nun übermittelt werden kann. Der Empfänger der verschlüsselten Nachricht kann über die Formel

$$m = c^d \mod N \tag{3}$$

die Nachricht wieder entschlüsseln.

3.4 Beweis der Gültigkeit

Die mathematische Gültigkeit des RSA-Verfahrens zu beweisen, bedeutet die Gleichung 3 zu belegen, sprich zu zeigen, dass für m folgender Zusammenhang

[3] Vgl. 22, S. 3.

gilt

$$\begin{aligned} m &= c^d \mod N \\ &= (m^e \mod N)^d \mod N \end{aligned}$$

Hierfür macht man sich den Satz von Euler-Fermat (Satz 2) zunutze.

$$\begin{aligned} m &= (m^e \mod N)^d \mod N \\ &= (m^e)^d \mod N \\ &= m^{ed} \mod N \end{aligned}$$

nach 3.2 ist d das multiplikative Inverse von e bezüglich des Moduls $\phi(N)$. Es existiert also ein $k \in \mathbb{Z}$, für welches $e * d = k * \phi(N) + 1$.

$$\begin{aligned} m &= m^{k*\phi(N)+1} \mod N \\ &= m^{k*\phi(N)} * m \mod N \\ &= (m^{\phi(N)})^k * m \mod N \\ &= ((m^{\phi(N)})^k \mod N) * (m \mod N) \mod N \\ &= ((m^{\phi(N)}) \mod N)^k * (m \mod N) \mod N \\ &= (1^k \mod N) * (m \mod N) \mod N \\ &= 1 * m \mod N \\ &= m \qquad \text{wegen } m < N \end{aligned}$$

3.5 Das RSA-Verfahren am Beispiel erklärt

Wie in der Fachliteratur üblich, werden im folgenden Beispiel für Sender, Empfänger und Angreifer die Synonyme Alice, Bob und Eve verwendet. „Alice und Bob kamen erstmals 1978 in der Schrift über Digitale Signaturen und Public-Key-Kryptosysteme von Ronald L. Rivest, Adi Shamir und Leonard M. Adleman vor.“[4] Ziel ist es, dass Alice Bob eine verschlüsselte Nachricht, in diesem Beispiel die Zahl 80, über einen offenen Kommunikationsweg übermitteln kann, ohne dass Eve in der Lage ist, diese zu entschlüsseln.

Zunächst muss Bob, wie in Abschnitt 3.2 beschrieben, seinen öffentlichen und privaten Schlüssel generieren. Hierzu wählt er die beide Primzahlen $p = 7$

[4] 21.

und $q = 13$ und berechnet aus diesen beiden Primzahlen sein RSA-Modul und $\phi(N)$

$$N = p * q = 7 * 13 = 91$$
$$\phi(N) = (p-1) * (q-1) = (7-1) * (13-1) = 72$$

Nun wählt sich Bob ein $e \in \mathbb{N}$ mit $1 < e < \phi(N)$, welches teilerfremd zu $\phi(N)$ ist, so beispielsweise $e = 5$. Als letzten Schritt muss Bob nun die Zahl d als multiplikatives Inverses von e bezüglich des Moduls $\phi(N)$ berechnen. Es muss also gelten

$$d * e \mod \phi(N) = 1.$$

Daraus folgt, dass ein k existiert, das folgende Gleichung erfüllt

$$1 = d * e + k * \phi(N).$$

Die beiden Faktoren k und d kann man durch den erweiterten euklidischen Algorithmus bestimmen.

$$72 = 14 * 5 + 2$$
$$5 = 2 * 2 + 1$$

$$\begin{aligned} 1 &= 5 - 2 * 2 \\ &= 5 - 2 * (72 - 14 * 5) \\ &= 29 * 5 - 2 * 72 \end{aligned}$$

Bobs öffentlicher Schlüssel setzt sich aus e und N zusammen und lautet $(5, 91)$. Diesen öffentlichen Schlüssel kann er nun über den ungesicherten Kommunikationsweg mit Alice teilen und auch Eve darf Zugriff auf dieses Tupel haben. Seinen privaten Schlüssel $d = 29$ muss er jedoch sicher vor Eve verwahren, da dieser das Entschlüssel der geheimen Nachricht ermöglicht.

Möchte Alice Bob nun die Zahl 80 verschlüsselt übermitteln, so muss sie diese nach Abschnitt 3.3 durch die Formel

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

verschlüsseln. Bezogen auf unser Beispiel ergibt sich

$$c = 80^5 \mod 91 = 19.$$

Nun übermittelt Alice die verschlüsselte Nachricht $c = 19$ an Bob. Dieser kann sie über die Formel

$$m = c^d \mod N$$

wieder entschlüsseln und erhält als Resultat

$$m = 19^{29} \mod 91 = 80$$

die zuvor von Alice verschlüsselte Nachricht $m = 80$. Die Angreiferin Eve hat nicht die Möglichkeit die Nachricht c zu entschlüssen, ohne den Wert von d zu kennen. Natürlich ist dieses Beispiel in der Praxis sehr unsicher, da die Sicherheit des RSA-Verfahrens auf dem Problem der Primfaktorzerlegung beruht, welches jedoch für kleine Primzahlen p und q auch mit aktueller Rechentechnik sehr schnell lösbar ist. Allerdings zeigt dieses Beispiel gut die Grundprinzipien von RSA.

3.6 Signieren von Nachrichten

Das RSA-Verfahren kann jedoch nicht nur zur Verschlüsselung von Daten eingesetzt werden. Es bietet zusätzlich auch noch die Möglichkeit digitale Signaturen (eine „kryptografische Realisierung einer Unterschrift unter ein Dokument“[5]) zu erstellen. Durch diese Signatur ist es Alice möglich Bob zu beweisen, dass die verschlüsselte Nachricht von ihr und nicht etwa von der Angreiferin Eve stammt. „Das RSA-Signaturverfahren ist die Umkehrung des RSA-verschlüsselungsverfahrens.“[6]. Möchte Alice Bob die Nachricht m signiert übermitteln, so erzeugt sie die Signatur s über die Formel

$$s = m^d \mod N.$$

Diese Signatur übermittelt sie anschließend an Bob, welcher sie nun mit Alice öffentlichen Schlüssel e entschlüsseln kann.

$$m = s^e \mod N.$$

Nur Alice privater Schlüssel ist also in der Lage, ihre digitale Signatur zu erstellen. Dadurch ist es Eve nicht möglich, Nachrichten in Alice Namen zu versenden. Da eine Verschlüsselung der gesamten Nachricht mit zunehmender Länge dieser

[5] 9.
[6] 9.

sehr aufwendig wird, benutzt man die RSA-Signatur häufig in Kombination mit einer Kryptografischen Hash-Funktion. Diese Funktionen „generieren aus beliebig langen Datensätzen eine Zeichenkette mit einer festen Länge“[7]. Alice würde also zuerst den Hash-Wert von m signieren und diesen zusammen mit der unverschlüsselten Nachricht m an Bob übermitteln. Dieser bildet nun seinerseits erneut den Hash-Wert der unverschlüsselten Nachricht m und entschlüsselt die Signatur s. Anschließend vergleicht er die beiden Werte miteinander. Stimmen sie überein, so stammt die Nachricht definitiv von Alice, da nur sie im Besitz ihres privaten Schlüssels d ist.

3.7 Übertragung von ganzen Texten

Für die Übertragung von ganzen Texten ist es notwendig, die vorkommenden Zeichen zunächst in Zahlen zu überführen. Dafür kann man beispielsweise die UTF-8-Zeichentabelle (8-Bit Universal Character Set Transformation Format) verwenden, welche jedem Unicode-Zeichen eine bestimmte Zahl zuordnet.[8] Diese Zahlen setzt man anschließend zu einzelnen Blöcken zusammen, welche nach 3.4 nicht größer als das RSA-Modul N sein dürfen, und überträgt diese verschlüsselt zum Empfänger. Er entschlüsselt die einzelnen Blöcke und decodiert sie anschließend, unter Verwendung der bereits zuvor erwähnten UTF-8-Zeichentabelle, zu einem Klartext. Jedoch ist bei der Übertragung von ganzen Texten zu beachten, dass es sich bei dem RSA-Verfahren um ein deterministisches Verfahren handelt. Folglich wird es in der Regel wie in 3.8 beschrieben nicht für die Übertragung von ganzen Texten verwendet.

3.8 RSA in der Praxis

Mit zunehmender Schlüssellänge, in der Praxis werden inzwischen 2048 Bit[9] Schlüssel empfohlen, und der zunehmenden Länge von zu verschlüsselnden Daten, erweist sich das RSA-Verfahren in der Praxis häufig als zu zeitaufwändig, um überall eingesetzt zu werden. Zudem ist es für die Übertragung von ganzen Texten aufgrund seiner deterministischen Arbeitsweise ungeeignet. Aufgrund dessen arbeitet man häufig mit Hybriden Verschlüsselungsverfahren. Diese kombinieren die Vorzüge der Symmetrischen- und der Asymmetrischen Verschlüs-

[7] 20.
[8] 1.
[9] vgl. S. 6 4.

selungsverfahren, in dem zuerst ein symmetrischer Schlüssel asymmetrisch verschlüsselt übertragen wird und die folgenden Nachrichten durch diesen nur noch symmetrisch verschlüsselt werden.[10] Somit spart man enorme Rechenkapazität, während die Sicherheit weiterhin gewährleistet ist. So arbeitet beispielsweise das Verschlüsselungsprotokoll SSL, welches unter anderem beim Online-Banking oder Online-Shopping eingesetzt wird.

4 Angriffspunkte

4.1 Faktorisierung von N

Wie man in Abschnitt 3.2 leicht sehen kann, wäre ein Angreifer schnell in der Lage die RSA-Verschlüsselung zu knacken, wenn er im Besitz der beiden Primzahlen p und q wäre. Die Sicherheit des RSA-Verfahrens beruht also auf der Annahme, dass die Faktorisierung von N eine Einwegfunktion mit Falltür (siehe 2.9) ist, wobei die Zusatzinformation im Falle von RSA der private Schlüssel d darstellt. Während diese Annahme nach dem heutigen Stand für sehr große N stimmt, lassen sich kleine N mit Hilfe aktueller Rechentechnik leicht faktorisieren. Da Computer ständig weiterentwickelt, somit leistungsfähiger werden und nicht ausgeschlossen werden kann, dass in Zukunft eine effiziente Methode für die Faktorisierung von großen Zahlen gefunden wird, ist die Sicherheit des RSA-Verfahrens nicht dauerhaft garantiert.

4.2 Schadsoftware und physischer Zugriff

Doch die größte Sicherheitslücke stellt in der Regel nicht das RSA-Verfahren an sich, sondern vielmehr der einzelne Nutzer dar. Würde der private Schlüssel d stets sicher verwahrt werden, so wäre RSA in Kombination mit der hybriden Verschlüsselung ein sehr sicheres System. Allerdings spielt auch weiterhin der Aspekt Mensch eine entscheidende Rolle, wenn wir über Sicherheit reden. Dieser private Schlüssel, welcher häufig auf der Festplatte des Nutzers gespeichert ist, kann nämlich durch den einfachen Einsatz von Schadsoftware in fremde Hände gelangen, wodurch die verschlüsselte Kommunikation kompromitiert ist. Nicht zu vernachlässigen ist auch die Möglichkeit des physischen Zugriffs auf einen

[10] vgl. 13.

Rechner oder das sogenannte Social Engineering, „ein Verfahren, um sicherheitstechnisch relevante Daten durch Ausnutzung menschlicher Komponenten in Erfahrung zu bringen“.[11]

4.3 Quantencomputer als Gefahr für asymmetrische Verschlüsselungsverfahren

Während die Faktorisierung des RSA-Moduls N ab einer gewissen Schlüssellänge für klassische Rechner ein nahezu unlösbares Problem darstellt, sind Quantencomputer mit Hilfe des Shor-Algorithmus, welcher „zu einer natürlichen Zahl N einen nichttrivialien Faktor“[12] liefert, dazu durchaus in annehmbarer Zeit fähig.

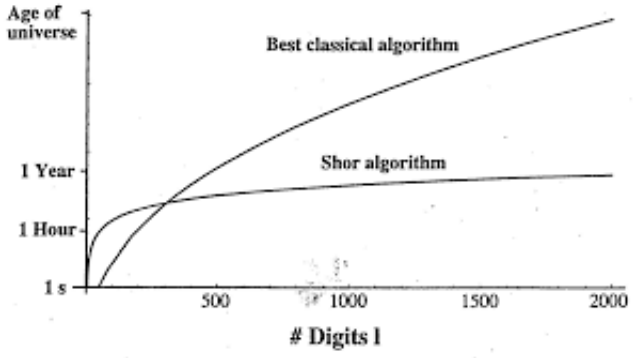


Abbildung 1: Laufzeit der Lösung des Faktorisierungsproblemes

https://www.tfp.kit.edu/downloads/lehre_2013_ss/aaa_HSS13_Shor.pdf

Dies liegt daran, dass die Laufzeit der Lösung des Faktorisierungsproblemes bei klassischen Algorithmen exponentiell mit der Größe von N zunimmt, während die Laufzeit des Shor-Algorithmus polynomiell zunimmt. Allerdings benötigt man für die Faktorisierung einer Zahl N mindestens $\log N$ Qubits, die kleinstmöglichen Speichereinheiten eines Quantencomputers. Die Entwicklung von Quantencomputern macht jedoch rasende Fortschritte, wodurch mittlerweile Quantencomputer existieren, welche 72 Qubits besitzen.[13] In Zukunft

[11] 19.
[12] 15.
[13] vgl. 12.

dürften unsere klassischen Verschlüsselungsverfahren also durch die Verbreitung von Quantencomputern gefährdet sein.

5 Quantenkryptografie am Beispiel des BB84-Protokolls

5.1 Schlüsselgenerierung

Das 1984 von Charles H. Bennett und Gilles Brassard vorgeschlagene BB84-Protokoll ermöglicht die Generierung eines Schlüssels für die Verschlüsselung von Nachrichten und zählt zu den bedeutendsten Verfahren der Quantenkrypotgrafie.[14] Zur Übertragung der Informationen werden Photonen benutzt. Diese Photonen können auf vier verscheidene Art und Weisen polarisiert sein. Man unterscheidet zwischen der horizontalen und der vertikalen Polarisation, welche im Folgenden zur Veranschaulichung durch die Zeichen - und | dargestellt werden, und zusätzlich noch zwischen der linksdiagonalen und der rechtsdiagonalen Polarisation (im Folgenden als \ und / bezeichnet). Die Polarisation eines Photons kann mit Hilfe eines Polarisationsfilters gemessen werden, wobei zwischen den horizontal-vertikalen (+) und den links-rechts-diagonalen (x) Polarisaitionsfiltern unterschieden wird. Wird ein horizontal polarisiertes Photon (-) mit einem horizontal-vertikalen Polarisationsfilter gemessen, so erhält man als Ergebnis, dass es sich um ein horizontal polarisiertes Photon handelt. Wiederholt man diese Messung des gleichen Photons mit einem links-rechts-diagonalen Polarisationsfilter, so erhält man aufgrund der Superposition der Photonen zu einer Wahrscheinlichkeit von 50% ein links-diagonales Photon (\) und zu einer Wahrscheinlichkeit von 50% ein rechts-diagonales Photon (/) als Messergebnis.

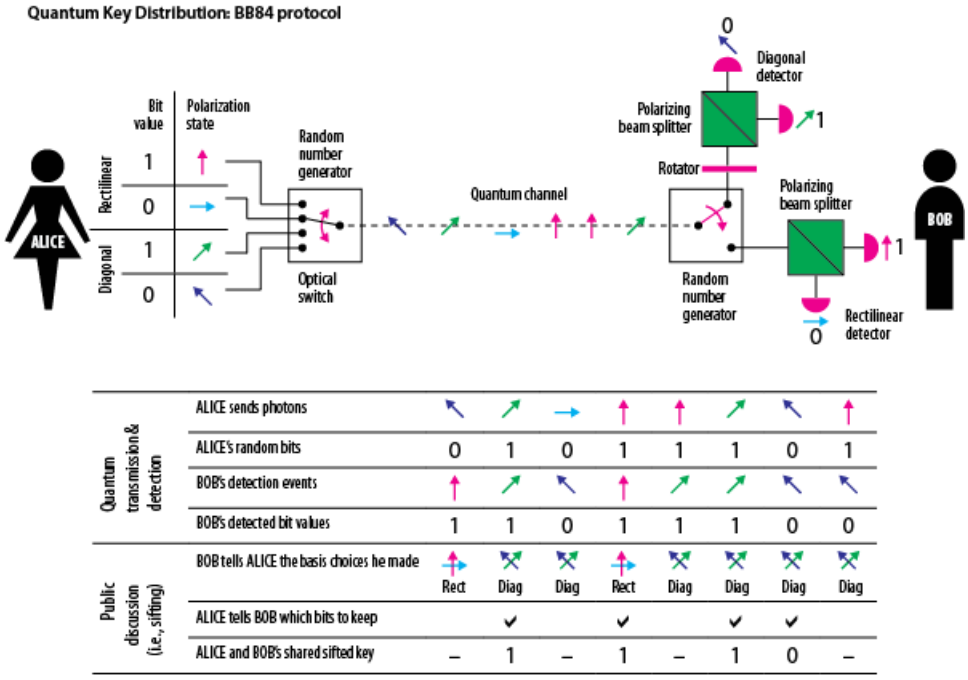
Alice legt zunächst fest, das ein mit einem horizontal-vertikalen Polarisationsfilter gemessenes horizontales Photon (-) den Bitwert 1 und ein vertikales Photon (|) den Bitwert 0 repräsentiert. Analog geht sie für die Photonen vor, die durch einen links-rechts-diagonalen Polarisationsfilter gemessen worden sind. Dadurch ergibt sich, folgende Beziehung:

$$-\widehat{=}1 \qquad |=0$$
$$\backslash\widehat{=}0 \qquad /=1$$

Diese Beziehung dient nur beispielhaft und kann getauscht werden. Wichtig ist

[14] vgl. 16.

nur, dass sich Alice und Bob über die Bitinterpretation der Polarisation der Photonen einig sind.



Quantum transmission & detection	ALICE sends photons	↖	↗	→	↑	↑	↗	↖	↑
	ALICE's random bits	0	1	0	1	1	1	0	1
	BOB's detection events	↑	↗	↖	↑	↗	↗	↖	↖
	BOB's detected bit values	1	1	0	1	1	1	0	0
Public discussion (i.e., sifting)	BOB tells ALICE the basis choices he made	+ Rect	× Diag	× Diag	+ Rect	× Diag	× Diag	× Diag	× Diag
	ALICE tells BOB which bits to keep		✔		✔		✔	✔	
	ALICE and BOB's shared sifted key	–	1	–	1	–	1	0	–

Abbildung 2: Beispiel des BB84-Protokolls in der Praxis

http://physique.unice.fr/sem6/2014-2015/PagesWeb/PT/Tomographie/?page=bb84

Nun erzeugt Alice eine Anzahl an Photonen. Diese werden mit einem Quantenzufallsgenerator auf die beiden Polarisationsfilter verteilt, so das Alice die gemessene Polarisation der Photonen als Bitfolge darstellen kann. Diese polarisierten Photonen leitet sie an Bob weiter. Bob besitzt so wie Alice zwei verschiedene Polarisationsfilter und ermittelte nun ebenfalls mit einem zufällig ausgewählten Polarisationsfilter die Polarisation der Photonen und stellt sie als Bitfolge dar. Nachdem dieser Austausch stattgefunden hat, informiert Alice Bob nun über einen öffentlichen Kanal, welche Polarisationsfilter sie für die einzelnen Photonen verwendet hat. Bob vergleicht seine verwendeten Polarisationsfilter mit denen von Alice. Für den Fall, dass sie beide den gleichen Polarisationsfilter verwendet haben, kann sich Bob sicher sein, dass Alice und er auch den selben Bitwert als Repräsentanten haben. Die Wahrscheinlichkeit, dass Alice und

Bob für ein Photon den gleichen Polarisationsfilter verwendet haben, liegt bei 50%. Sollten sie nicht den gleichen Filter verwendet haben, so kann Bob nicht eindeutig sagen, ob Alice und er den gleichen Bitwert verwendet haben.[15]

Die Bitwerte, für welche sie den selben Polarisationsfilter verwendet haben, bilden nun den geheimen Schlüssel. Diesen können sie für eine verschlüsselte Kommunikation verwenden.

5.2 Abhörsicherheit des Verfahrens

Für die Angreiferin Eve gibt es nun keine Möglichkeit den geheimen Schlüssel abzufangen. Aufgrund des No-Cloning-Theorem ist es nicht möglich, eine Kopie eines Quantenobjekts anzufertigen, ohne dessen Zustand zu verändern.[16] Eve kann also keine Kopie anfertigen, ohne das Alice und Bob etwas davon mitbekommen. Auch kann sie die Photonen nicht einfach abfangen und anschließend an Bob weiterleiten, denn auch sie müsste die Polarisation der Photonen durch einen Polarisationsfilter bestimmen. Folglich hätte sie eine 50:50 Chance den selben Filter wie Alice zu verwenden. Verwendet sie den richtigen Filter, so fällt dies weder Alice noch Bob auf. Verwendet sie jedoch den falschen Filter, so würde die Polarisation des Photons dauerhaft verändert werden, womit Alice und Bob durch einen Vergleich ihrer gemeinsamen Bitsequenz Fehler feststellen würden. Mit zunehmender Anzahl der Photonen wird die Chance nicht aufzufallen immer geringer und geht gegen null. Schlussendlich hilft Eve auch die Information über die verwendeten Polarisationsfilter im Nachhinein nicht weiter, da sie nun zwar weiß, welche Filter benutzt worden sind, aber trotzdem nicht in der Lage ist, Rückschlüsse auf die Polarisation der Photonen zu ziehen. Somit ist es Eve in der Praxis unmöglich die Bitfolge, die zur Verschlüsselung von geheimen Nachrichten verwendet wird, herauszufinden. Folglich stellt das BB84-Protokoll eine sichere Alternative zu aktuellen Verschlüsselungsverfahren dar.

5.3 Nachteile der Quantenkryptografie

Den größten Nachteil der Quantenkryptografie ist stand heute der fehlende Ausbau von Quantenkanälen. Diese werden zur Übertragung der im BB84-Protokoll beschriebenen Photonen benötigt und an sie werden ungeheure Ansprüche ge-

[15] vgl. 8.
[16] vgl. 17.

stellt, da der Zustand der Photonen schnell durch die Wechselwirkung mit anderen Teilchen beeinflusst werden kann. Glasfaserkabel, die für die Übertragung von Bits verwendet werden, eigenen sich auch nur bedingt für die Übertragung von Photonen, da klassische Signalverstärker nicht einsetzbar sind. Die bisher größte durch Glasfaserkabel realisierte Entfernung zum Austausch eines Schlüssels beträgt 184,6km. 2017 gelang es Forschern einen Quantenschlüssel mit einem Satellieten auf die Erdoberfläche zu übertragen. Es wurde eine Distanz von etwa 1200km überbrückt.[17]

Ein weiterer Nachteil der Quantenkrypotgrafie ist die Tatsache, dass sowohl Sender als auch Empfänger einen Quantencomputer für die Generierung des Schlüssels besitzen müssen. Zwar kann man solche Quantencomputer bereits kaufen, doch sind sie bisher keinesfalls massentauglich. Bis wir Quantenkryptografie im Alltag einsetzen können, wird also wohl noch einiges an Zeit vergehen, allerdings ist durch sie eine sichere Alternative zu den klassischen Verschlüsselungsverfahren bereits geschaffen, womit wir also auch in Zukunft unsere Informationen sicher und geheim übertragen können.

6 RSA-Toolkit als praktische Umsetzung

Um die mathematischen Grundlagen in die Praxis umzusetzen habe ich ein Programm entwickelt, das die Schlüsselgenerierung, die Ver- und Entschlüsselung von ASCII kodierten Klartextnachrichten und die Faktorisierung von N und somit die Berechnung des privaten Schlüssels d aus dem öffentlichen Tupel (e, N) ermöglicht. Das gesamte Programm ist in der Programmiersprache Python entwickelt worden. Bei der Schlüsselgenerierung kann der Nutzer die Schlüssellänge beliebig wählen, sie muss jedoch mindestens 8 Bit betragen. Ausgehend von dieser Schlüssellänge werden zwei zufällige Zahlen in der gewählten Größenordnung generiert. Die beiden Zahlen werden durch einen probabilistischen Primzahltest, in diesem Fall den Miller-Rabin-Test[18], überprüft. Es werden so lange neue Zufallszahlen p und q generiert, bis beide Zahlen prim sind. Für die Berechnung von e wird die gleiche Methode verwendet mit der zusätzlichen Bedingung, dass $1 < e < \phi(N)$ gilt. Für die Berechnung des privaten Schlüssels habe ich das Modul „Sympy“[19] verwendet. Die Anwendung bietet die Möglichkeit berechnete

[17] vgl. 25.
[18] vgl. 24.
[19] vgl. 23.

Schlüssel in einer Datei zu speichern und sie aus dieser zu laden.

Da das RSA-Verfahren nur mit Zahlen und nicht mit Klartext arbeitet, muss der zu verschlüsselnde Klartext zunächst in eine Zahl umgewandelt werden. Analog muss bei der Entschlüsselung die entschlüsselte Zahl in einen Klartext umgewandelt werden. Das Programm verwendet für die Kodierung von Zeichen den „Amerikanischen Standard-Code für den Informationsaustausch“ (kurz ASCII)[20].

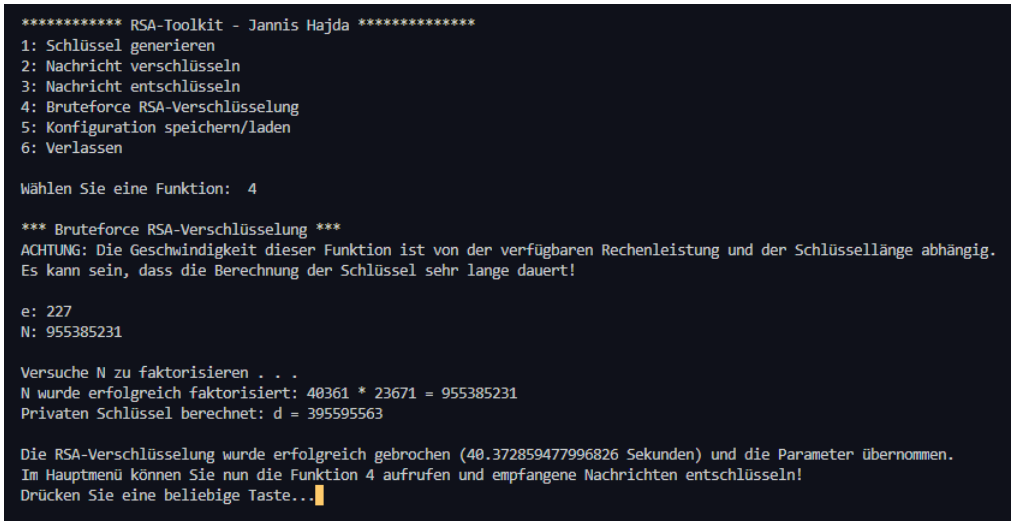


Abbildung 3: RSA-Toolkit

Die eingebaute Funktion die RSA-Verschlüsselung zu brechen, liefert für kleine Schlüssellängen ein schnelles Ergebnis. Die Laufzeit der Funktion richtet sich hierbei nach der verfügbaren Rechenleistung und der Länge des Schlüssels. So war ich beispielsweise in der Lage, mit der mir zur Verfügung stehenden Rechenleistung, eine 32 Bit Verschlüsselung in etwa 40 Sekunden zu knacken. Starte ich den gleichen Versuch jedoch mit einer 1024 Bit Zahl, so ist eine Lösung des Faktorisierungsproblems in absehbarer Zeit nicht möglich. Somit wurde veranschaulicht, wie sehr sich die Schlüssellänge auf die Sicherheit des RSA-Verfahrens auswirkt. Der Quelltext des Programms kann unter https://github.com/jannishajda/RSA-Toolkit abgerufen werden.

[20] vgl. 2.

7 Fazit

Das Ziel der vorliegenden Arbeit war es, die mathematischen Grundlagen hinter dem RSA-Verfahren zu verstehen, eventuelle Sicherheitslücken aufzudecken und durch einen Blick in die Zukunft die Frage nach der Sicherheit des Verschlüsselungsverfahrens beantworten zu können. Im Verlaufe der Arbeit ist deutlich geworden, auf welchen mathematischen Grundlagen RSA basiert und wie man diese in der Praxis anwendet. Dadurch war ich in der Lage, eine eigene Implementierung des RSA-Verfahrens in meiner entwickelten Anwendung einzubauen. Auch wurden mögliche Sicherheitslücken aufgedeckt, welche primär eine geringe Schlüssellänge, sowie Schadsoftware sind. Bezüglich der Sicherheit lässt sich zusammenfassend jedoch sagen, dass RSA zum momentanen Zeitpunkt richtig angewendet, eine sehr sichere, jedoch in Zukunft durch die ständige Weiterentwicklung von Quantencomputern gefährdete Verschlüsselungsmethode ist. Allerdings stellt die in Abschnitt 5 beschriebene Quantenkryptografie eine zukunftssichere Alternative dar. Sie gewährleistet uns auch in Zukunft, sicher und geheim über ungeschützte Kanäle miteinander zu kommunizieren.

Literatur

[1] 1 und 1 Ionos SE. *UTF-8: Der Standard im Netz*. URL: https://www.ionos.de/digitalguide/websites/webseiten-erstellen/utf-8-codierung-globaler-digitaler-kommunikation/ (besucht am 28. 08. 2019).

[2] *ASCII Table*. URL: http://www.asciitable.com/ (besucht am 14. 10. 2019).

[3] Hochschule Augsburg. *Kryptographische Grundbegriffe*. URL: http://www.hs-augsburg.de/~meixner/saj/skript/sicherheit/KryptographieGrundbegriffe.html (besucht am 06. 08. 2019).

[4] Elaine Barker und Allen Roginsky. *Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths*. URL: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-131Ar1.pdf (besucht am 27. 09. 2019).

[5] Albrecht Beutelspacher. *Eine kurze Geschichte der Kryptografie*. URL: http://www.bpb.de/apuz/259145/eine-kurze-geschichte-der-kryptografie?p=all (besucht am 06. 08. 2019).

[6] Michael Calderbank. *The RSA Cryptosystem: History, Algorithm, Primes*. URL: http://math.uchicago.edu/~may/VIGRE/VIGRE2007/REUPapers/INCOMING/REU%20paper.pdf (besucht am 06. 08. 2019).

[7] Franz Embacher. *Der erweiterte euklidische Algorithmus*. URL: https://www.mathe-online.at/materialien/Franz.Embacher/files/RSA/Euklid.html (besucht am 24. 09. 2019).

[8] Cedric Engels. *Physikalisch unhackbar: Quantenkryptografie | Quanten Teil 4*. URL: https://www.youtube.com/watch?v=wks0F_h2FFQ (besucht am 29. 09. 2019).

[9] Fh-Flensburg. *RSA-Signatur*. URL: http://www.inf.fh-flensburg.de/lang/krypto/protokolle/rsa-signatur.htm (besucht am 12. 09. 2019).

[10] O. Forster. *Einfuhrung in die Zahlentheorie*. 24. Feb. 2017. URL: http://www.mathematik.uni-muenchen.de/~forster/v/zth/inzth_02.pdf (besucht am 06. 08. 2019).

[11] W. Gellert u. a. *Kleine Enzyklopädie Mathematik*. 4. Aufl. VEB Bibliographisches Institut Leipzig, 1969.

[12] Martin Giles und Will Knight. *Quantencomputer: Er fährt hoch.* URL: https://www.heise.de/tr/artikel/Quantencomputer-Er-faehrt-hoch-4132785.html (besucht am 29. 09. 2019).

[13] GNU. *Hybride Verschlüsselungsverfahren.* URL: https://www.gnupg.org/gph/de/manual/x112.html (besucht am 24. 09. 2019).

[14] Herbert Göthner und Peter Kästner. *Algebra — aller Anfang ist leicht.* Springer, 2013.

[15] Tanja Kempfert. *Shor-Algorithmus.* URL: https://www.tfp.kit.edu/downloads/lehre_2013_ss/aaa_HSS13_Shor.pdf (besucht am 29. 09. 2019).

[16] Dr. Björn Lippold. *Quantenkryptografie.* URL: https://www.chemie.de/lexikon/Quantenkryptografie.html (besucht am 29. 09. 2019).

[17] Quantiki. *The no-cloning theorem.* URL: https://www.quantiki.org/wiki/no-cloning-theorem (besucht am 29. 09. 2019).

[18] *RSA – Primzahlen zur Verschlusselung von Nachrichten.* URL: https://www.scai.fraunhofer.de/content/dam/scai/de/documents/Mediathek/Mathematik%20f%C3%BCr%20die%20Praxis/rsa_skript_und_arbeitsblaetter.pdf (besucht am 06. 08. 2019).

[19] Peter Schmitz. *Was ist Social Engineering?* URL: https://www.security-insider.de/was-ist-social-engineering-a-633582/ (besucht am 27. 09. 2019).

[20] Patrick Schnabel. *Kryptografische Hash-Funktionen.* URL: https://www.elektronik-kompendium.de/sites/net/1909041.htm (besucht am 12. 09. 2019).

[21] Patrick Schnabel. *Wer sind Alice, Bob und Mallory?* URL: https://www.elektronik-kompendium.de/sites/net/1909021.htm (besucht am 05. 09. 2019).

[22] Nora Schweppe. *Das Verschlüsselungsverfahren RSA.* URL: https://www.humboldtschule-berlin.de/images/pdf/informatik/rsa.pdf (besucht am 06. 08. 2019).

[23] SymPy Development Team. *SymPy.* URL: https://www.sympy.org/ (besucht am 14. 10. 2019).

[24] Prof. Dr. Edmund Weitz. *Der Miller-Rabin-Test.* URL: https://www.youtube.com/watch?v=Tqq6hxxnhEs (besucht am 14. 10. 2019).

[25] Wikipedia. *Quantenschlüsselaustausch.* URL: https://de.wikipedia.org/wiki/Quantenschl%C3%BCsselaustausch (besucht am 29. 09. 2019).

[26] Wikipedia. *RSA (cryptosystem).* URL: https://en.wikipedia.org/wiki/RSA_(cryptosystem) (besucht am 06. 08. 2019).

Selbstständigkeitserklärung

Hiermit versichere ich Jannis Hajda, dass ich die Seminararbeit mit dem Thema: „Wie sicher sind heutige Verschlüsselungsmethoden am Beispiel der RSA-Verschlüsselung?“ selbstständig verfasst habe und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt wurden, sowie Zitate kenntlich gemacht habe.

Leegebruch, den 15. Oktober 2019 .....

Unterschrift