Kryptologie:
Hinleitung zum RSA-Verfahren

Facharbeit von

Niels Hagen Kirschke

vorgelegt im Schuljahr 2014/2015

Stiftisches Humanistisches Gymnasium
Leistungskurs Mathematik bei Hr. Jacobs

Inhaltsverzeichnis

„The decisions we make about communication security today will determine the kind of society we live in tomorrow."(Whitfield Diffie)
"Entscheidungen, die wir heute über Sicherheit treffen, werden die Gesellschaft der Zukunft bestimmen"

I. EINLEITUNG

1. Vorwort

Seit jeher wollte der Mensch Mitteilungen überbringen, ohne dass jemand Unerwünschtes sie mithören oder mitlesen kann. Also existierte seit jeher der Bedarf nach Verschlüsselungsmethoden. Während das Verschlüsseln vor der Zeit des Computers in der Regel mühselig war, so wurde es umso einfacher für einen jeden, nachdem der Computer als High-Speed-Verschlüsselungs- und Entschlüsselungsmaschine erfunden wurde. Allerdings ging damit der Anspruch einher, Verschlüsselungsmethoden mathematisch aufzubauen. Ein Computer versteht nur Mathematik, und man musste Verschlüsselungsmethoden immer sicherer aufbauen, da ein Computer nicht nur wunderbar verschlüsseln kann, sondern auch den Schlüssel von anderen schnell knacken kann.

Mittlerweile sind die Folgen für den einzelnen jedoch stärker spürbar als vor tausenden von Jahren. Wenn die Verschlüsselung beim Online-Banking versagt oder Ziffern von Buchungen verschluckt werden, weil die Verschlüsselungsfunktion nicht *injektiv* war, hat das zur Folge, dass der Kontoinhaber ggf. um einiges ärmer oder an Mahnungen reicher ist. Außerdem ist in der heutigen Zeit des Internets das Bewusstsein um private Daten erheblich gestiegen. Menschen wollen nicht alle Informationen über sich selbst preisgeben.

Aus diesem Grund stieg auch der Bedarf nach sicheren Verschlüsselungsmethoden im Privatbereich in den letzten Jahren erheblich an. Damit einher mussten viele Wissenschaftler, häufig Mathematiker und Informatiker, immer höheren Ansprüchen von Firmen- und Privatkunden nachkommen. Fehlende Sicherheit oder auch nur mangelnde Sicherheit ist somit zum K.O.-Schlag für datensensible Unternehmen geworden.

Aus diesem Anspruch an erhöhte Sicherheit durch die Mathematik entstand 1977 das RSA-Verfahren (nach Ron **R**ivest, Abi **S**hamir und Leonard **A**dleman benannt). Aus welchen Ideen das RSA-Verfahren entstand, möchte ich in dieser Fach-

arbeit erläutern. Aber ich möchte auch die Funktionsweise des RSA-Verfahren und ein paar andere kleinerer Methoden hier erklären.

2. Einführung in das Thema

Um eine Nachricht zu verschlüsseln muss sie in Ziffern übertragen werden. Man kann wie Cäsar, in Kapitel II.1 beschrieben, allen Buchstaben von A-Z die Ziffern 0-25 zuweisen. Spätestens wenn man Umlaute, Satzzeichen und Zahlen mit dazu nimmt, zeigt das System Grenzen auf. Was würde aus der Nachricht: „Bring 2 Äpfel und 7 Birnen mit!“ Die Umlaute könnte man zur Not noch in ae, oe und ue umwandeln. Wenn man aber die Zahlen übertragen will, muss man sie ausschreiben, was alsbald sehr mühselig wird, da man den Buchstaben schon die Zahlen 0-25 zugewiesen hat.

Somit ist heutzutage der einheitliche Ziffernstandard für Buchstaben, Satzzeichen, Zahlen und Akzente etc. der ASCII Code, Tab. 6. Mit ihm lassen sich auch Groß- und Kleinbuchstaben unterscheiden. Der Einfachheit halber wird der ASCII-Code jedoch nur in Kapitel III verwendet.

Um mit Nachrichten in Zahlenform rechnen zu können, braucht man auch die Sicherheit, dass die Nachricht nach Ver- und Entschlüsselung wieder dieselbe ist, eine Falltür oder Sicherheitsmechanismus. Dieser ist häufig die Rechenoperation ***a mod b = c.*** Sie sorgt dafür, dass nach Addition etc. nicht ein neuer Buchstabe herauskommt, sondern nach dem letzten Buchstaben wieder beim ersten angefangen wird. Außerdem muss eine Verschlüsselungs- beziehungsweise Entschlüsselungsformel stets injektiv sein, weil eine Zahl immer eindeutig einem Zeichen zugeordnet werden muss.

Die Modulo-Funktion kreiert ***b*** Restklassen. Denn eine Division durch zum Beispiel drei als Modul, kann nur den Rest 0, 1, 2 haben. Verschlüsselungsfunktionen verändern essentiell nur die Restklassen des Klartexts (siehe Kapitel III.4).

II. HINLEITUNG ZUM RSA-VERFAHREN

1. Verschlüsseln durch Addition

Die Verschlüsselung durch Addition war ein, von Cäsar populär gemachtes Verschlüsselungsverfahren, bei dem ein Buchstabe ***N*** um eine bestimmte Anzahl an Stellen ***s*** im Alphabet verschoben wird. Die Buchstaben A-Z werden durch die Zahlen 0-25 ersetzt, Tab. 4, und nicht nach dem ASCII Code aufgelöst. Für sein Verfahren nutzte Cäsar zwei Scheiben, wovon eine beweglich war. Diese stellte die innere Scheibe dar. Auf beiden wurde das Alphabet abgedruckt, siehe Abb.1. Somit ist mathematisch ausgedrückt Cäsars Verfahren folgendes:

$$C(M) = M + s \; mod \; 26$$

Die Funktion mod ***26*** wird hier genutzt, um sicherzustellen, dass der Definitionsbereich $D \in [0{,}25]$ eingehalten wird. Wenn Y, Nummer 24, zum Beispiel um 10 Stellen verschoben wird nicht 34 zugeordnet wird, sondern 9; also J. Wenn wir nun den Text „caesar methode“ mit ***s = 8*** verschlüsseln. So kommt „kimaiz umbpwlm“ als verschlüsselte Nachricht heraus.

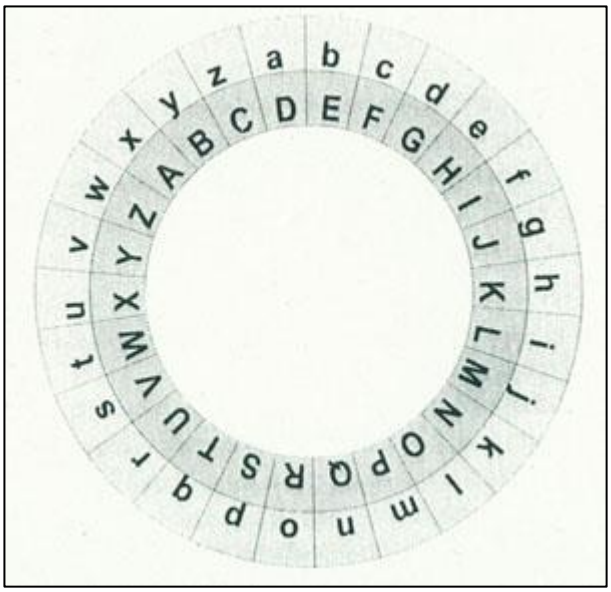

Abbildung 1

Aber sobald ***s*** bekannt ist, lässt sich der Text durch einfache Subtraktion wieder entschlüsseln. Aus der Funktion $C(M) = M + s \; mod \; 26$ wird: $M(C) = C - s \; mod \; 26$. Mathematisch gesehen ist aber auch interessant zu sehen, ob es eine ganze Zahl gibt, die Funktion in ihrer Struktur erhält:

$$M(C) = C + s \; mod \; 26$$

w nennt sich eine additive Inverse zu ***s***. Für Inverse gilt in diesem Fall, dass sie in derselben Restklasse sein müssen, wie die Anzahl der Möglichkeiten, in diesem Fall 26. Mathematisch ausgedrückt bedeutet das:

$$s + w = 0 \; mod \; 26$$

Da ***w*** die Unbekannte ist, wird die oben genannte Gleichung wie folgt umgestellt:

$$s + w = 0 \; mod \; 26$$

$$w = -s \; mod \; 26$$

Wenn man nun für ***s*** = 8 einsetzt erhält man als additives Inverses 18. Denn 18+8=26. Für das Beispiel bedeutet das:

$$M(C) = C + 18 \, mod \, 26$$
$$M(10) = 10 + 18 \, mod \, 26$$
$$= 28 \, mod \, 26$$
$$= 2$$

Wenn man nun in der Tabelle den, der zwei zugeordneten, Buchstaben nachsieht, wird aus der zwei wieder „c“ und aus „kimaiz umbpwlm“ wird „caesar methode“.

Die Sicherheit dieses Verfahren ist nur sehr gering. Computer können in Sekundenschnelle 26 Möglichkeiten durchprobieren und ein Mensch kann darauf rasch die vom Computer ausgegebenen Möglichkeiten durchlesen und die sinnvolle heraus suchen. Aus diesem Grund wird das Cäsar Verfahren heutzutage auch nicht mehr als alleiniges Verfahren angewendet. Einzig als hybride Verschlüsselung mit einem weiteren Verfahren findet es heute noch vereinzelt Anwendung.

2. Verschlüsseln durch Multiplikation

Wenn man die Unsicherheit einer additiven Verschlüsselung vor Augen sieht, könnte man auf die Idee kommen, dass eine multiplikative Verschlüsselung sicherer ist. Um multiplikativ zu verschlüsseln, geht man ähnlich wie bei der Cäsar Methode so vor, dass man einen Klartext ***M*** mit einem Faktor ***s*** multipliziert, wie zuvor bei der Cäsar Methode addiert. Danach kann man mit dem modular multiplikativen Inversen von ***s***, ***d***, die verschlüsselte Nachricht ***C*** in den Klartext umwandeln. Für die Formeln bedeutet das:

$$C(M) = M \cdot s \, mod \, N$$
$$M(C) = C \cdot d \, mod \, N$$

Nun kann man die Funktionstüchtigkeit der Verschlüsselung für ***s*** = 7 ausprobieren. ***N*** sollte diesmal 27 sein, da zwar 26 Buchstaben vergeben sind, aber die Funktion nur mit ganzen positiven Zahlen funktioniert und damit nicht injektiv wäre, wenn a = 00 ist. Der Einfachheit halber wird a = 26 und der Text „multiplikation“ wird somit wie folgt verschlüsselt.

1. „multiplikation“ wird in Zahlen umgewandelt (Tab. 5, S. 17)
   - „multiplikation“ → 12 20 11 19 08 15 11 08 10 26 19 08 14 13
2. die Zahlen werden mit ***s*** = 4 und ***N*** = verschlüsselt

$$C(12) = 12 \cdot 7 \, mod \, 27$$
$$C(12) = 3$$

   - es entsteht: 03 05 23 25 02 24 23 02 16 20 25 02 17 10

Nun gibt es aber ein Problem, den Entschlüsselungsfaktor ***d*** zu finden. Wie bei der Cäsar-Methode kann man zunächst versuchen, die Nachricht durch ***s*** zu teilen. Das macht aber keinen Sinn, wenn ein Ergebnis herauskommt welches nicht $M \in \mathbb{Z}$ ist. Zum Beispiel würde bei der Division von drei durch den Schlüssel sieben $\frac{3}{7}$ herauskommen, welches keinem Buchstaben zugeordnet ist.

Da die Division durch den Schlüssel nicht funktioniert, kann man versuchen, ähnlich wie in der Cäsar-Methode auch, das Inverse des Schlüssels zu finden. In diesem Falle ist das allerdings nicht das additive sondern das multiplikative Inverse zu erzeugen. Wichtig ist, dass die beiden Schlüssel, ***d*** und ***e***, auch modulare Inverse sind, um die Injektivität der Funktion und den Definitionsbereich einzuhalten. Das bedeutet:

$$d = e^{-1}$$

$$d = \frac{1}{e}$$

Um das modulare Inverse einer Zahl zu bestimmen, nutzt man die Formel:

$$a \cdot b \; mod \; N = 1$$

$$d \cdot e \; mod \; 27 = 1$$

In diesem Falle muss man lediglich in einer Multiplikationstafel, welche Modulo 27 ist (Tab. 1), nachsehen welche beiden Zahlen das Produkt eins bilden (vgl. Puhlmann, S.3). Diese Produkte sind in der Tabelle grün eingezeichnet. Abgesehen von 1 · 1 ist nur die sieben als ***e*** und die vier als ***d*** funktionstüchtig. Aus diesem Grund kann man für das Beispiel die Schlüssel ***e*** = 7 und ***d*** = 4 vollkommen legitim verwenden.

mod27	0	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8	9	10
2	0	2	4	6	8	10	12	14	16	18	20
3	0	3	6	9	12	15	18	21	24	0	3
4	0	4	8	12	16	20	24	1	5	9	13
5	0	5	10	15	20	25	3	8	13	18	23
6	0	6	12	18	24	3	9	15	21	0	6
7	0	7	14	21	1	8	15	22	2	9	16
8	0	8	16	24	5	13	21	2	10	18	26
9	0	9	18	0	9	18	0	9	18	0	9
10	0	10	20	3	13	23	6	16	26	9	19

Tabelle 1

Aus der Tabelle geht aber auch eine weitere Regel hervor: Der Modul und der Schlüssel müssen teilerfremd sein. Das heißt$ggT(s,n) = 1$. Für das Beispiel lässt sich dies zweifelsfrei bestätigen. Ansonsten gäbe es zu dem Schlüssel ***e*** keinen Faktor zu denen das Produkt gleich eins ist.

Nun setzt man die verschlüsselte Nachricht in die Funktion ein:

$$M(C) = C \cdot 4 \; mod \; 27$$
$$M(3) = 3 \; \cdot 4 \; mod \; 27$$
$$M(3) = 12$$

Und aus der Zahlenkolonne wird wieder „multiplikation“.

Leider ist das Multiplikationsverfahren in keiner Weise sicherer als die Cäsar-Methode, da es bei dem Knacken der Nachricht nur $N - 1$ Möglichkeiten gibt. Bei ***N*** = 27 existieren somit nur 26 Möglichkeiten. Angenommen der Versender der Nachricht hat einen funktionstüchtigen invertierbaren Schlüssel genutzt, reduzieren sich die möglichen Schlüssel nochmals radikal.- Für den Raum von 1-10 gibt es nur drei Möglichkeiten.

Beim Versand der Nachricht unterscheidet man bei den Schlüsseln in *private* und *public* key. Der *public key* besteht aus ***N*** und ***e***. Dieser *key* wird dem Versender der Nachricht übergeben. Dieser verschlüsselt damit die Nachricht und sendet die Nachricht dann dem Versender des *public key* zu. Dieser entschlüsselt die Botschaft mit dem geheimen *private key* ***d***.

In den folgenden Tabellen findet sich eine weitere Übersicht über kleine Multiplikationstafeln anhand derer man für mod 3, 4 und 5 die Schlüssel ablesen kann.

mod3	0	1	2	3
0	0	0	0	0
1	0	1	2	0
2	0	2	1	0
3	0	0	0	0

mod4	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	0	2
3	0	3	2	1

mod5	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	4	6
3	0	3	6	9

Man könnte an diesem Punkt noch die Verschlüsselung durch Potenzieren anbringen. Da diese jedoch nahezu identisch mit dem RSA-Verfahren in Kapitel III ist, wird davon Abstand genommen, redundante Informationen an diesem Punkt anzuführen.

III. DAS RSA-VERFAHREN

1. Die „keys“

Das RSA-Verfahren führt die Idee weiter, mit ansteigendem Schwierigkeitsgrad der Operationen auch eine höhere Sicherheit zu erlangen. Da diese Idee beim Multiplikationsverfahren nicht sehr hilfreich war, setzt das RSA-Verfahren auf mehr als nur das Potenzieren der Nachricht mit dem Schlüssel.

Für das RSA-Verfahren benötigt man mehrere Komponenten, die sich wie beim Multiplikationsverfahren in zwei Kerngruppen unterteilen lassen: Es gibt den *private key* und den sogenannten *public key*, wobei der *public key* aus dem *private key* berechnet wird. Der private key ist stets geheim zu halten. Der Vorteil des RSA-Verfahren liegt darin, dass dieser Schlüssel nicht verschickt werden muss.

Für den *private key* braucht man lediglich zwei große Primzahlen ***p*** und ***q***. Diese werden miteinander multipliziert, sodass eine Zahl ***N*** entsteht. Auch wenn die Multiplikation zweier Primzahlen sehr simpel ist, braucht man für deren Umkehrung, also die Primfaktorzerlegung einen ungeheuren Aufwand. Bei kleineren Zahlen ist dies noch relativ einfach von Rechnern zu erledigen. Doch Primzahlen mit vielen hundert Stellen sind für Computer in einem tolerierbaren Zeitraum nicht herauszufinden. Aus diesem Grund zählt man die Multiplikation zweier großer Primzahlen auch zu den *Einwegfunktionen*. Einwegfunktionen sind simpel zu lösende Funktionen, deren Umkehrung hingegen nicht oder kaum lösbar ist.

Mathematisch ausgedrückt berechnet man ***N*** wie folgt:

$$N = p \cdot q$$

$$\text{für } p \wedge q = prim$$

Für das Beispiel ergibt sich:

$$N = 97 \cdot 83$$

$$= 8051$$

Neben ***N*** braucht man für das RSA-Verfahren aber auch einen Exponenten ***e***. Wichtig ist, dass ***e*** teilerfremd zu $\varphi(N)$ ist. Dazu berechnet man:

$$\varphi(N) = (p-1)(q-1)$$

$$\varphi(9797) = (97-1)(101-1)$$

$$= (96)(100)$$

$$= 9600$$

Durch die Primfaktorzerlegung kann man nun sämtliche ***e*** bestimmen, die zu $\varphi(N)$ **nicht** teilerfremd sind:

$$
\begin{aligned}
9600 &= 2 \cdot 4800 \\
&= 2 \cdot 2 \cdot 2400 \\
&= 2 \cdot 2 \cdot 2 \cdot 1200 \\
&= 2 \cdot 2 \cdot 2 \cdot 2 \cdot 600 \\
&= 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 \cdot 300 \\
&= 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 \cdot 150 \\
&= 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 \cdot 75 \\
&= 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 \cdot 3 \cdot 25 \\
&= 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 \cdot 3 \cdot 5 \cdot 5
\end{aligned}
$$

Für das Beispiel wissen wir nun, dass ***e*** teilerfremd zu 2, 3 und 5 sein muss. Also ist ***e*** = 13 eine legitime Wahl. Diese Komponenten, ***N*** und ***e***, machen den *public key* aus. Dieser wird dem Gesprächspartner zur Verschlüsselung übermittelt. Dabei ist die Geheimhaltung der Daten bei der Übermittlung unerheblich.

2. Die Verschlüsselung

Für die Verschlüsselung konvertiert der Gesprächspartner seine Nachricht, wie bei den anderen Methoden auch, seine Nachricht in Zahlen. Häufig wird beim RSA-Verfahren die Nachricht in Blöcke von mehreren Buchstaben respektive Zahlen geteilt. Damit lässt sich in einem Verschlüsselungsvorgang mehr von der Nachricht auf einmal verschlüsseln. Wichtig hierbei ist, dass die Zahlengruppe niemals größer ist als ***N***. Damit gilt die Bedingung ***M*** < ***N***; wobei M für die zu verschlüsselnde Nachricht, Message, steht.

p	97
q	101
N	9797
e	13
d	1477

Tabelle 2

Will also Partner A Partner B die Nachricht „ICH BIN ANDREAS.“ schicken, so teilt er sie zunächst in Zweiergruppen ein, weil ***N*** = 9797 ist und damit Zahlen, die länger als vier Stellen sind, ausschließt. Somit wird aus der Nachricht: „IC H BI N AN DR EA S.“ Im ASCII Code ausgedrückt lautet die Nachricht wie folgt: „7367 7232 6673 7832 6578 6882 6965 8346“ Man beachte, dass die Leerzeichen der Nachricht mit in die Zahlenfolge einbezogen wurden.

Nun, da man die geheime Botschaft in Zahlengruppen übertragen hat, kann man schließlich anfangen, die Nachricht zu verschlüsseln. Für die Verschlüsselung nutzt man folgende Funktion:

$$C = M^e mod\ N$$

Hierbei steht ***C*** für die verschlüsselte Nachricht bzw. **c**rypted message. Am Beispiel der ersten Zahlengruppe sieht man, dass man mit der neuen Zahl, die her-

auskommt, gar nichts anfangen kann, da man sie auch nicht mehr so einfach in Buchstaben umwandeln kann.

$$C = 7367^{13}\ mod\ 9797$$

$$C = 6955$$

Wenn man dies für alle Zahlengruppen durchführt erhält man folgende unten stehende Zahlenfolge. Anzumerken ist, dass die verschlüsselte Nachricht nicht mehr durch Leerzeichen zwischen den Gruppen getrennt werden muss, da die Leerzeichen der im Buchstabenformat vorliegenden Nachricht schon implementiert sind.

„6955838584586531641037548969183 9“

3. Die Entschlüsselung

Für die Entschlüsselung der Nachricht braucht man nebst *public key* auch den Entschlüsselungsexponenten ***d***, für **d**ecipher. Dieser ist das multiplikative Inverse zum Verschlüsselungsexponenten ***e***. Um ***d*** zu berechnen muss man zunächst dessen Definitionsbereich definieren. Der Definitionsbereich von ***d*** ist: $d \in]0\ , \varphi(N)[$. Für das Beispiel bedeutet das, dass 0 < ***d*** < 9600. Nachdem man den Definitionsbereich für ***d*** festgelegt hat, muss man nun mittels des erweiterten euklidischen Algorithmus ***d*** bestimmen (vgl. Beutelspacher, S. 107). Dafür ist es am sinnvollsten eine Tabelle anzulegen:

	e	$\boldsymbol{\varphi(N)}$	***z***	***R***	***s***	***t***
3	13	9600	0	13	***1477***	-2
2	9600	13	738	6	-2	1477
1	13	6	2	1	1	-2
0	6	1	6	0	0	1

Tabelle 3

Hierbei wird der ggT($\varphi(N), e$) ermittelt. Da 13 eine Primzahl ist, ist eigentlich schon von vorne herein ersichtlich, dass der ggT = 1 sein muss. Allerdings ermittelt man mithilfe des Rests der einzelnen Divisionsschritte und des Faktors das multiplikative Inverse zu ***e***. Hierbei wird nach der Durchführung des euklidischen Algorithmus von der unteren Stufe ausgehend die Tabelle nach oben hin ausgefüllt. Zur Ermittlung von ***s*** und ***t*** wird die folgende Formel genutzt:

$$t_n = s_{n-1} - (z \cdot t_{n-1})$$

Nach dieser Methode wird die restliche Tabelle ausgefüllt. Dabei gilt stets $\boldsymbol{s_n} = \boldsymbol{t_{n-1}}$. Schließlich lässt sich in der ersten bzw. zweiten Zeile der Entschlüsselungsexponent ***d*** ablesen. ***d*** ist sowohl das ***s*** der ersten Spalte als auch das ***t*** der zweiten Spalte. Für das Beispiel bedeutet das, dass ***d*** = 1477 ist:

$$t_1 = 0 - (2 \cdot 1) = -2$$
$$t_2 = 1 - (738 \cdot (-2)) = 1477$$
$$t_3 = (-2) - (0 \cdot 1477) = -2$$

Nachdem überprüft wurde, ob dieser Wert im Definitionsbereich von ***d*** liegt, kann man seine Korrektheit einfach überprüfen, indem man eine bekannte Buchstabenfolge: „IC“, beziehungsweise deren Zahlenwert: „7367“ verschlüsselt, um darauf durch ***d*** entschlüsselt zu werden. Kommt „IC“ heraus, ist es das richtige ***d***. Die Funktion für die Entschlüsselung lautet:

$$M = C^d \; mod \; N$$

Auf das Beispiel bezogen setzt man für ***C*** die verschlüsselte Zahlenfolge ein:

$$M = 6955^{1477} \; mod \; 9797$$
$$= 7367$$

Die unverschlüsselte Zahlenfolge: „7367“ stimmt mit der Botschaft überein. Zu beachten ist, dass große Exponenten die Rechenleistung eines normalen Taschenrechners übersteigen. Aus diesem Grund wird empfohlen für Vorführungszwecke kleine ***p*** und ***q*** zu wählen, damit ***N*** klein genug bleibt.

3. Die Übertragung

Will Andreas nun Sophie eine verschlüsselte Nachricht schicken, so bittet er Sophie, zwei große Primzahlen ***p*** und ***q*** miteinander zu multiplizieren. Das Produkt ***N*** und der von Sophie frei gewählte und zu $\varphi(N)$ teilerfremde Exponent ***e*** werden Andreas mitgeteilt, wobei ***p*** und ***q*** geheim gehalten werden müssen. Andreas verschlüsselt seine Nachricht mithilfe von ***N*** und ***e*** und schickt sie Sophie. Diese hat mit dem erweiterten euklidischen Algorithmus aus ***p*** und ***q*** den Entschlüsselungsexponenten ***d*** berechnet und entschlüsselt somit Andreas‘ Botschaft. Auf dem Weg kann keiner, der die Nachricht abgreift, diese entschlüsseln, da kein relevanter Schlüssel übertragen wurde. Verschlüsselungssysteme, in denen keine entschlüsselnden Schlüssel übertragen werden, nennt man asymmetrisch (vgl. Ferres, S. 24).

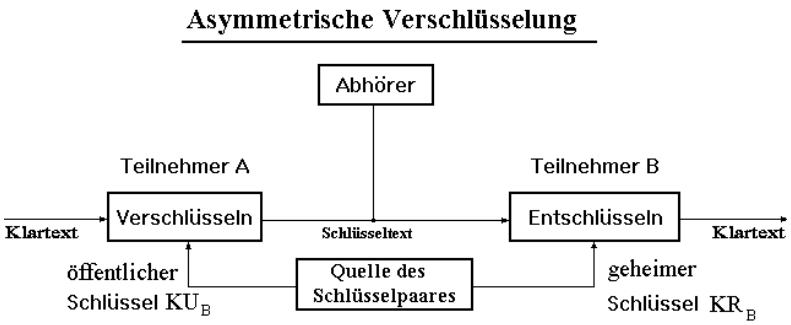

Abbildung 2

4. Die Mathematik dahinter

Bei jedem der vorgestellten Verfahren war es wichtig, dass die Form der Verschlüsselungsfunktion genau dieselbe war wie die der Entschlüsselungsfunktion. Nur in diesem Fall kann man mit dem Inversen des (Ver-)Schlüssels rechnen. Hierbei gibt es jedoch Unterschiede; während die Cäsar-Chiffre mit dem simplen additiven Inversen des Schlüssels ***e*** auskommt, muss man für die multiplikative Verschlüsselung das multiplikative Inverse des Schlüssels errechnen.

Wichtig ist auch, dass man sich bei jedem Verfahren bewusst ist, in welchen Restklassenringen man sich bewegt. Bei einer jeden Modulo Operation, mit dem Modul ***N***, entstehen ***N*** Restklassen. Damit eine Verschlüsselungsfunktion injektiv ist, muss es mindestens so viele Restklassen geben, wie es Zeichen in der Botschaft gibt: bei der Cäsar-Substitution 26 und im ASCII-Code 255.

Die Restklassen helfen, die Buchstaben, bei additiver bzw. multiplikativer Verschiebung beizubehalten. Daraus ergibt sich, dass eine Nachricht bei additiver Verschiebung aus einer Restklasse ***r*** mithilfe von ***e*** in die Restklasse ***t*** überführt wird. Umgekehrt muss das Inverse die Nachricht aus ***t*** in die ursprüngliche Restklasse ***r*** rücküberführen. Für Additionen bedeutet das, dass die Summe der Schlüssel gleich dem Modul ***N*** sein muss.

Addition:

$$e + d \; mod \; N = 0$$

$$e + d = N$$

Bei der Multiplikation werden Restklassen immer mehr als nur einfach um ***e*** verschoben, sondern ***e***-fach. Das bedeutet, dass bei der Verschlüsselung die Restklasse **2** mit dem Modul ***N*** = 10, also den Restklassen {**1**, **2**, **3**,…,**10**}, und dem Schlüssel ***e*** = 7 in die Restklasse **4** (4 **=** 2 · 7 mod 10) übergeht und nicht in **9** (9 = 2 + 7 mod 10). Da aufgrund der nicht injektiven Multiplikation mit null die Stelle 00 keinem Buchstaben zugeordnet werden kann, muss $e \cdot d \; mod \; N$ nicht gleich null sein sondern gleich eins.

Multiplikation:

$$e \cdot d \; mod \; N = 1$$

$$e \cdot d = x \cdot N + 1$$

Auf das RSA-Verfahren bezogen:

Da aus der obigen Formel herausgeht, dass ***e*** mal ***d*** teilerfremd zu **N** sind, und **N** eine Zahl ist, die sich aus den Primzahlen ***p*** und ***q*** zusammensetzt, kann man daraus auch Ableiten, dass $\varphi(N)$ teilerfremd zu dem Produkt aus ***e*** und ***d*** sein muss. Denn wenn ***N*** aus zwei Primzahlen besteht, ist: $\varphi(N) = (p-1)(q-1)$, weil ***N*** nur

durch die eigenen Primfaktoren und eins teilbar ist. Will man als Empfänger der Nachricht ***d*** berechnen, was in diesem Fall auch ein multiplikatives Inverses ist, obwohl ***d*** ein Exponent ist, weil die Potenzen einer Zahl in derselben Restklasse wie die Zahl hoch eins liegen, muss gelten:

$$ggT\big(e \cdot d, \varphi(N)\big) = 1$$

Daher kann man ***d*** mithilfe des erweiterten euklidischen Algorithmus berechnen. Man kann sich die Funktion nämlich auch sinnlich leicht zurückformen, um zu beobachten, dass für den Exponenten ***e*** nur das multiplikative Inverse gebraucht wird:

$$C(M) = M^d \; mod \; N$$

$$M(C) = \sqrt[d]{C} \; mod \; N$$

$$M(C) = C^{\frac{1}{d}} \; mod \; N$$

Nun lässt sich leicht erkennen, dass zum Entschlüsseln lediglich mit dem multiplikativ modularen Inversen potenziert werden muss.

III. Bewertung

Alle der Beschriebenen Verfahren basieren darauf, dass eine Verschlüsselungsfunktion C(M) sich nur im Schlüssel von der Entschlüsselungsfunktion M(C) unterscheidet. Um diesen Umstand zu ermöglichen muss jeweils zur Entschlüsselung das Inverse des Schlüssels gefunden werden.

Dies gilt für alle beschriebenen Verfahren, weshalb sich behaupten lässt, dass das von ***R****ivest*, ***S****hamir* und ***A****dleman* vollendete und nach ihnen benannte RSA-Verfahren eine komplexe Weiterentwicklung der Cäsar-Methode ist.

Allerdings hat das asymmetrische RSA-Verfahren aber auch einige gravierende Nachteile. Um eine ausreichende Sicherheit zu gewähren, muss man sehr große Primzahlen verwenden. Diese erzeugen ***N*** mit mehr als 600 Stellen. Und dennoch gibt es gravierende Nachteile in der Praxis. Zum einen dauert das RSA-Verfahren sehr lange. Verglichen mit symmetrischen Verfahren braucht das RSA-Verfahren bei der Verschlüsselung, Übermittlung und Entschlüsselung um ein Vielfaches länger. Außerdem bietet das RSA-Verfahren unter gewissen Bedingungen auch Angriffsfläche gegenüber Hackern. Eine weitere Schwierigkeit ist das Auffinden von großen Primzahlen. Martin Mersenne (1588-1648), ein französischer Mönch, kreierte mithilfe von Eulers Theorie zu den Primzahlen eine Primzahl mit 78 Stellen, was man damals aufgrund der Größe nicht nachrechnen konnte (vgl. Bellos S. 271). Aber selbst diese Primzahl wäre zu klein gewesen, um ein ***N*** mit 600 Stellen zu erzeugen.

Aus diesen Gründen wird das RSA-Verfahren häufig mit symmetrischen Verschlüsselungssystemen verbunden. Dies nennt sich dann *hybride Verschlüsselung*, da die Nachteile der einzelnen Verfahren durch die Vorteile des jeweils anderen ausgeglichen werden. Hierbei wird das RSA-Verfahren in der Regel genutzt, um die Schlüssel des symmetrischen Systems zu übermitteln. Dadurch erlangt man höhere Sicherheitsstandards, ohne dass man große Geschwindigkeitseinbußen hinnehmen muss.

Sämtliche Systeme die ihre Schlüssel in *private* und *public* unterteilen, nennen sich heute auch *Public-Key Cryptography* (Henderson, S. 180f.). Anders als der Name des RSA-Verfahren vermuten lässt, sind Rivest, Shamir und Adleman nicht die Erfinder dieses Verfahren. Sie wurden auf die Methode erst aufmerksam, nachdem die amerikanischen Mathematiker und Informatiker Whitfield Diffie und Martin Hellmann einen wissenschaftlichen Text mit Formeln zum Thema *Public-Key Cryptography* in 1976 in der *IEEE Transactions on Information Theory* veröffentlicht haben (Henderson, S.145f.). Rivest, Shamir und Adleman adaptierten es damals nur und machten es praxistauglicher und gaben dem Verfahren ihren Namen.

Mittlerweile wird eine verfeinerte Form des RSA-Verfahren genutzt, um Bankpasswörter und Ausweisnummern zu verschlüsseln. Für mehr hat es aber aufgrund der langen Zeit nicht gereicht.

> **„**If you depend on a secret for your security, what do you do when the secret is discovered? If it is easy to change, like a cryptographic key, you do so. If it's hard to change, like a cryptographic system or an operating system, you're stuck. You will be vulnerable until you invest the time and money to design another system.**”** (Whitfield Diffie)

> „Wenn die Sicherheit von Geheimnissen abhängt, ist die Frage: Was tut man, wenn die Geheinisse öffentlich werden? Wenn es einfach zu ändern ist, wie ein Schlüssel, ändere es! Wenn es schwierig zu beheben ist, wie ein Verschlüsselungs- oder Betriebssystem, bleibt man stecken. Man bleibt verwundbar, bis man Zeit und Geld hat, neue Systeme zu entwickeln“

IV. Quellen

1. Literatur

Bellos, Alex: Im Wunderland der Zahlen. Eine mathemagische Reise, München-Zürich 2013, S. 269-267.

Beutelsbacher, Albrecht: Kryptologie. Eine Einführung in die Wissenschaft vom Verschlüsseln, Verbergen und Verheimlichen, Braunschweig-Wiesbaden 2002[6], S. 94-107

Diffie, Whitfield: "Interview with Whitfield Diffie on the Development of Public Key Cryptography." Interview durch Franco Furger. *Karlsruher Institut Für Technologie*. Arnd Weber, 16 Jan. 2002. (01.03.2015). <http://www.itas.kit.edu/pub/m/2002/wedi02a.htm>.

Ferres, Thorsten: Facharbeit Mathematik – Kryptologie, S. 24-25.

Henderson, Harry: "Diffie, Bailey Whitfield." *Encyclopedia of Computer Science and Technology*. New York, NY: Facts On File, 2003. S. 145-46.

Henderson, Harry: "Encryption." *Encyclopedia of Computer Science and Technology*. New York, NY: Facts On File, 2003. S. 180-81.

Puhlmann, Dr. Hermann: Kryptologie verstehen. Ein schülergerechter Zugang zum RSA-Verfahren, Darmstadt 1998, S. 3, 8f.

Schäfer, Katrin: RSA-Verschlüsselung. „Ich weiß etwas, das du nicht weißt…“, Wuppertal 2010 (01.03.2015). <http://www.matheprisma.uni-wuppertal.de/Module/RSA/>

Thoma, Martin: "Erweiterter Euklidischer Algorithmus." *Wikipedia*. 20.09.2014. Web. 27.02.2015. <http://de.wikipedia.org/wiki/Erweiterter_euklidischer_Algorithmus>.

2. Bild- und Tafelwerkverzeichnis

S. 5 http://wwws.phil.uni-passau.de/histhw/TutKrypto/tutorien/verschluesselung-antike-mittelalter.htm, S. 12 http://www.fh-wedel.de/~si/seminare/ws96/ausarbeitung/sicherh/sicherh2.htm, S. 16 http://www.asciitable.com/

V. Appendix I

Cäsar Substitution												
A	B	C	D	E	F	G	H	I	J	K	L	M
0	1	2	3	4	5	6	7	8	9	10	11	12
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
13	14	15	16	17	18	19	20	21	22	23	24	25

Tabelle 4

Multiplikation												
A	B	C	D	E	F	G	H	I	J	K	L	M
26	1	2	3	4	5	6	7	8	9	10	11	12
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
13	14	15	16	17	18	19	20	21	22	23	24	25

Tabelle 5

Auszug aus dem ASCII Code												
A	B	C	D	E	F	G	H	I	J	K	L	M
65	66	67	68	69	70	71	72	73	74	75	76	77
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
78	79	80	81	82	83	84	85	86	87	88	89	90
a	b	c	d	e	f	g	h	i	j	k	l	m
97	98	99	100	101	102	103	104	105	106	107	108	109
n	o	p	q	r	s	t	u	v	w	x	y	z
110	111	112	113	114	115	116	117	118	119	120	121	122
	.	,	0	1	2	3	4	5	6	7	8	9
32	46	44	48	49	50	51	52	53	54	55	56	57

Tabelle 6

VI. Appendix II

Glossar

a mod b = c	„a durch b lässt den Rest c“ oder: „a Modulo b ist gleich c“
asymmetrisch	Verschlüsselung ohne Schlüsselaustausch; ein Schlüssel wird für das Verschlüsseln und einer für das Entschlüsseln genutzt
Einwegfunktion	Simple injektive Funktion, deren Umkehrung sehr umständlich umzusetzen ist. z.B. Multiplikation großer Primzahlen
injektiv	Funktion, die jedem x-Wert einen einzigen y-Wert zuordnet
Modulare Inverse	$b \cdot c \bmod a = 1$
private key	geheime Anteile des public key; hat nur eine Person
public key	öffentlicher Schlüssel für die Verschlüsselung; für jeden zugänglich
Restklasse	Zahlen, die bei einer Division durch eine weitere Zahl den gleichen Rest lassen
Rivest, Shamir, Adleman	haben das RSA-Verfahren populär gemacht
Whitfield Diffie, Martin Hellmann	Entwickler der Public Key Kryptografie

CD mit der Datei

Ich erkläre, dass ich die Facharbeit ohne fremde Hilfe angefertigt und nur die im Literaturverzeichnis angeführten Quellen und Hilfsmittel benutzt habe. Beratungsgespräche mit dem Fachlehrer haben an folgenden Terminen stattgefunden:

- ***15.12.2014***
- ***27.02.2015***