Entwicklung der Asymmetrischen Kryptographie und deren Einsatz

Peter Kraml, 5a hlw

Schuljahr 2013/14

Inhaltsverzeichnis

Kapitel 1

Einleitung

Ich bin Teil einer Generation die mit dem Internet aufgewachsen ist. Meine erste Webseite hatte ich zum Beispiel im Alter von 5 Jahren. Schon früh begann ich mich deshalb dafür zu interessieren, wie das Internet funktioniert. Eines der spannendsten Punkte war für mich dabei schon immer, wie man im Internet Nachrichten sicher austauschen kann. Warum man zum Beispiel im Internet Bankgeschäfte erledigen kann, oder seinen Steuerausgleich abgeben kann. All das, sind sensible Daten, die nicht einfach im Klartext durch das Internet transportiert werden dürfen, weil sie sonst von Dritten einfach mitgelesen werden können.

Die Menschheit entwickelt schon seit Jahrtausenden Verfahren und Hilfsmittel um Nachrichten so zu transportieren, dass nur der Sender und der Empfänger sie lesen können. So kann die Fähigkeit, Nachrichten sicher zu verschlüsseln, wichtige Wettbewerbsvorteile bringen oder bei einem Krieg entscheidend sein.

So ist es nicht verwunderlich, dass eine Großzahl der Techniken mit denen Nachrichten heutzutage verschlüsselt werden aus der Zeit des Kalten Krieges kommen. Ein weiterer Punkt der hier sicherlich nicht unwesentlich ist, ist der Anstieg der Rechenleistung in dieser Zeit. Dies führte dazu, dass bessere Verschlüsselungstechniken implementiert werden konnten, für die andere Apparate zu langsam gewesen wären.

Eines der großen Probleme bei herkömmlicher symmetrischer Verschlüsselung ist die Schlüsselübertragung. Da derselbe Schlüssel zum Ver- und Entschlüsseln verwendet wird, muss der Schlüssel für sich bereits sicher übertragen werden. Würde ein Angreifer den Schlüssel mitlesen, so könnte er Nachrichten sowohl verschlüsseln, viel wichtiger aber, er könnte sie auch lesen.

Als Alternative zu dieser relativ unsicheren Methode der Verschlüsselung, wurde in den 1970er Jahren die asymmetrische Kryptographie entwickelt. Bei ihr werden zwei Schlüssel verwendet. Einer zum Verschlüsseln und ein anderer zum Entschlüsseln. Um eine Nachricht sicher zu transportieren, kann der Sender sie mit dem öffentlichen Schlüssel des Empfängers verschlüsseln. Entschlüsselt werden kann die Nachricht allerdings nur mehr mit dem privaten Schlüssel des Empfängers, den der Sender, aber auch ein möglicher Angreifer, nicht kennt.

Dadurch eignet sich die asymmetrische Kryptographie hervorragend für den Einsatz im Internet, wo potenziell jeder Teilnehmer ein Angreifer sein könnte. Das RSA-Verfahren ist das wohl meist bekannteste Beispiel aus dieser Familie, und der de-facto Standard wenn im Internet Daten sicher transportiert werden müssen.

Kapitel 2

Asymmetrische Kryptographie

2.1 Die Geschichte der Verschlüsselung

Wie bereits in der Einleitung erwähnt, wird bei der symmetrischen Verschlüsselung nur ein Schlüssel verwendet. Mit diesem Schlüssel kann eine Nachricht sowohl ent- als auch verschlüsselt werden. Ein einfaches Beispiel dafür ist das Caesar-Verfahren.

Beim Caesar-Verfahren, werden die Buchstaben im Alphabet beim Verschlüsseln verschoben. Der Schlüssel gibt in diesem Fall an, um wie viele Buchstaben der Text verschoben wird. Nimmt man als Schlüssel also zum Beispiel »C«, oder 3, so würde das verschobene Alphabet wie folgt aussehen:

```
Ausgang:    ABCDEFGHIJKLMNOPQRSTUVWXYZ
Caesar:     DEFGHIJKLMNOPQRSTUVWXYZABC
```

Will man also eine Nachricht verschlüsseln, müssen lediglich alle Buchstaben richtig verschoben werden. Zum Entschlüsseln verwendet man ebenfalls wieder den Schlüssel »C«, bzw. 3, verschiebt allerdings das Alphabet in die entgegengesetzte Richtung um wieder auf den Klartext zu kommen.

Vgl. [1] S. 25f

Kann ein Angreifer die verschlüsselte Nachricht mitlesen, so kann er einfach alle möglichen Schlüssel durchprobieren, bis er wieder zur originalen Nachricht zurückkommt. Bei maximal 25 Versuchen (Eine Verschiebung um 26 Buchstaben im Alphabet würde wieder den Ausgangsbuchstaben zurückgeben) keine große Herausforderung für moderne Computer.

Vgl. [2] »Entzifferung und Sicherheit«

Eine erste Form der Weiterentwicklung könnte daher zum Beispiel sein, das Alphabet das zum Verschlüsseln verwendet wird, nicht in der richtigen Reihenfolge anzubringen. Dadurch ergeben sich bereits signifikant mehr Möglichkeiten ($26! \cong 4 * 10^{26}$) die ein Angreifer ausprobieren müsste um auf den originalen Text zurückzukommen.

Vgl. [1] S. 26

Ein Problem das mit der erhöhten Schlüsselanzahl aber noch immer nicht behoben ist, ist dass der Inhalt kryptographisch analysiert werden kann. In der geschriebenen Sprache kommen Buchstaben unterschiedlich häufig vor. Diese Häufigkeitsverteilung bleibt beim Verschieben des Alphabetes erhalten. Deswegen kann mit etwas Raffinesse auch hier wieder auf den ursprünglichen Inhalt geschlossen werden.

Vgl. [2] »Entzifferung und Sicherheit«

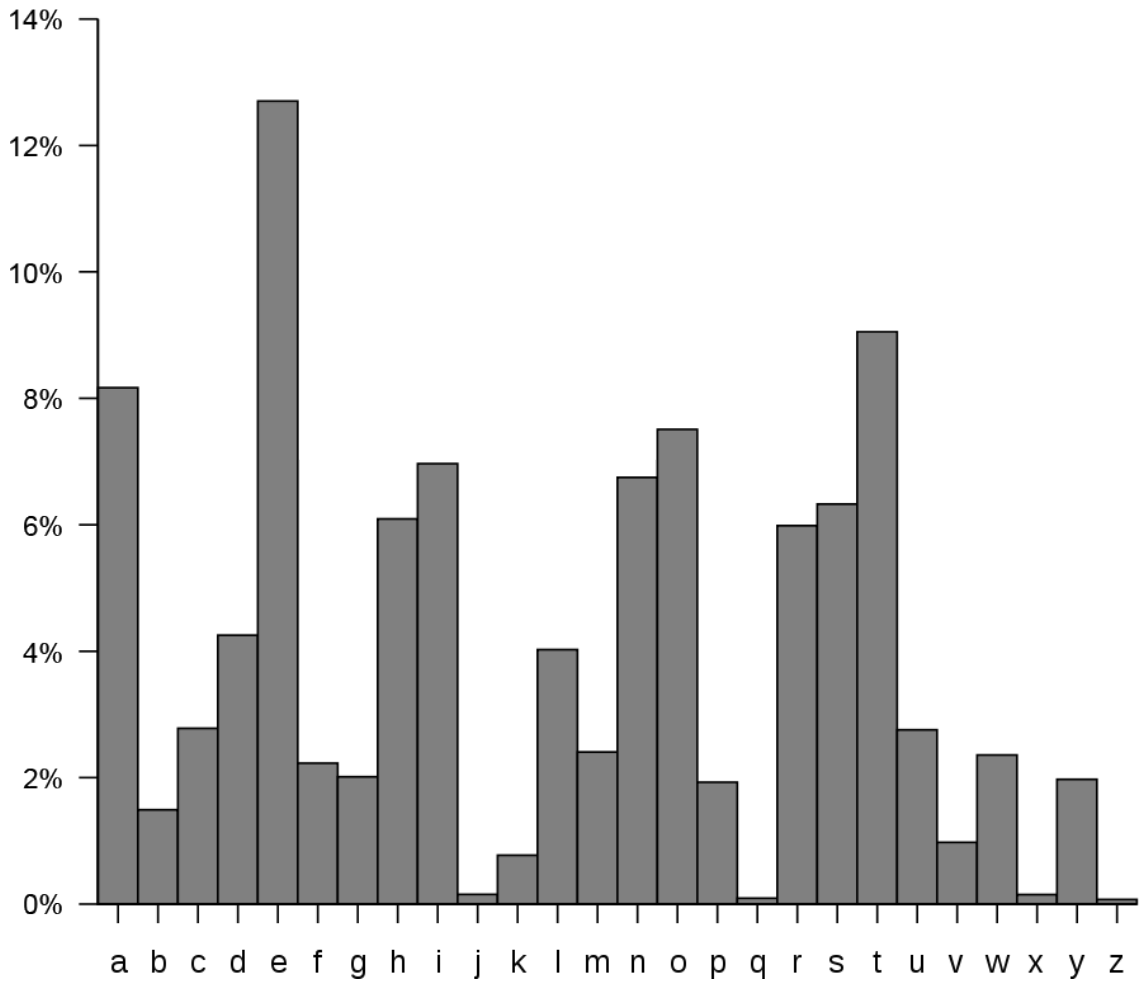

Abbildung 2.1: Buchstabenhäufigkeit der Englischen Sprache

2.2 Grundproblem: Der Schlüsselaustausch

Schlussendlich ergibt sich aber noch ein viel wichtigeres Problem: Der Sender muss dem Empfänger den verwendeten Schlüssel irgendwie mitteilen, damit dieser die Nachricht korrekt entschlüsseln kann. Gelingt es dem Angreifer diese Schlüsselübertragung ebenfalls abzuhören, so ist die ganze Verschlüsselung ebenfalls gebrochen. Der sicherste Weg wäre daher ein direkter Austausch des Schlüssels von Person zu Person. Für das Internet ist dies aber logischerweise keine akzeptable Möglichkeit.

Eine Lösung für das Problem, den Schlüssel sicher zu übertragen, fanden erstmals die Mathematiker Martin Hellman, Whitfield Diffie und Ralph Merkle von der Stanford Universität in Kalifornien. Sie veröffentlichten ihr Ergebnis, den »Diffie-Hellman-Schlüsselaustausch« im Jahr 1976, welches auch die Basis für das später beschriebene RSA-Verfahren bildet.

Vgl. [1] S. 320

Abbildung 2.2: V. l. n. r.: Ralph Merkle, Martin Hellman, Whitfield Diffie

Das Verfahren kann mithilfe einer Kiste und drei Vorhängeschlössern und zugehörigen Schlüsseln dargestellt werden. In Abbildung 2.3 wird davon ausgegangen, dass Bob Alice einen Liebesbrief verschlüsselt senden möchte.

Dazu legt Alice in eine Kiste das Schloss C mit einem dazugehörigem Schlüssel. Den zweiten dazugehörigen Schlüssel behält sie bei sich um am Ende die Nachricht von ihrem Liebhaber wieder entschlüsseln zu können.

Nun sperrt sie die Kiste mit ihrem Schloss A zu und sendet sie zu Bob. Bob sperrt die Kiste nun zusätzlich mit seinem Schlüssel B ab.

Die doppelt gesicherte Kiste schickt Bob nun wieder zurück an Alice, welche ihr Schloss A von der Kiste entfernt. Daraufhin schickt sie die Kiste ein letztes Mal zu Bob.

Die Kiste ist jetzt nur noch mit seinem Schloss B gesichert, zu welchem er den Schlüssel hat. Somit kann er die Kiste öffnen und das Schloss C mit dem zweiten dazugehörigem Schlüssel herausnehmen.

Er kann nun Nachrichten mit dem Schloss C zusperren, welche Alice dann mit ihrem Schlüssel aufsperren kann. Somit fand ein Schlüsselaustausch zwischen Alice und Bob statt, ohne dass ein Angreifer den Schlüssel einfach mitlesen könnte.

Vgl. [1] S. 313

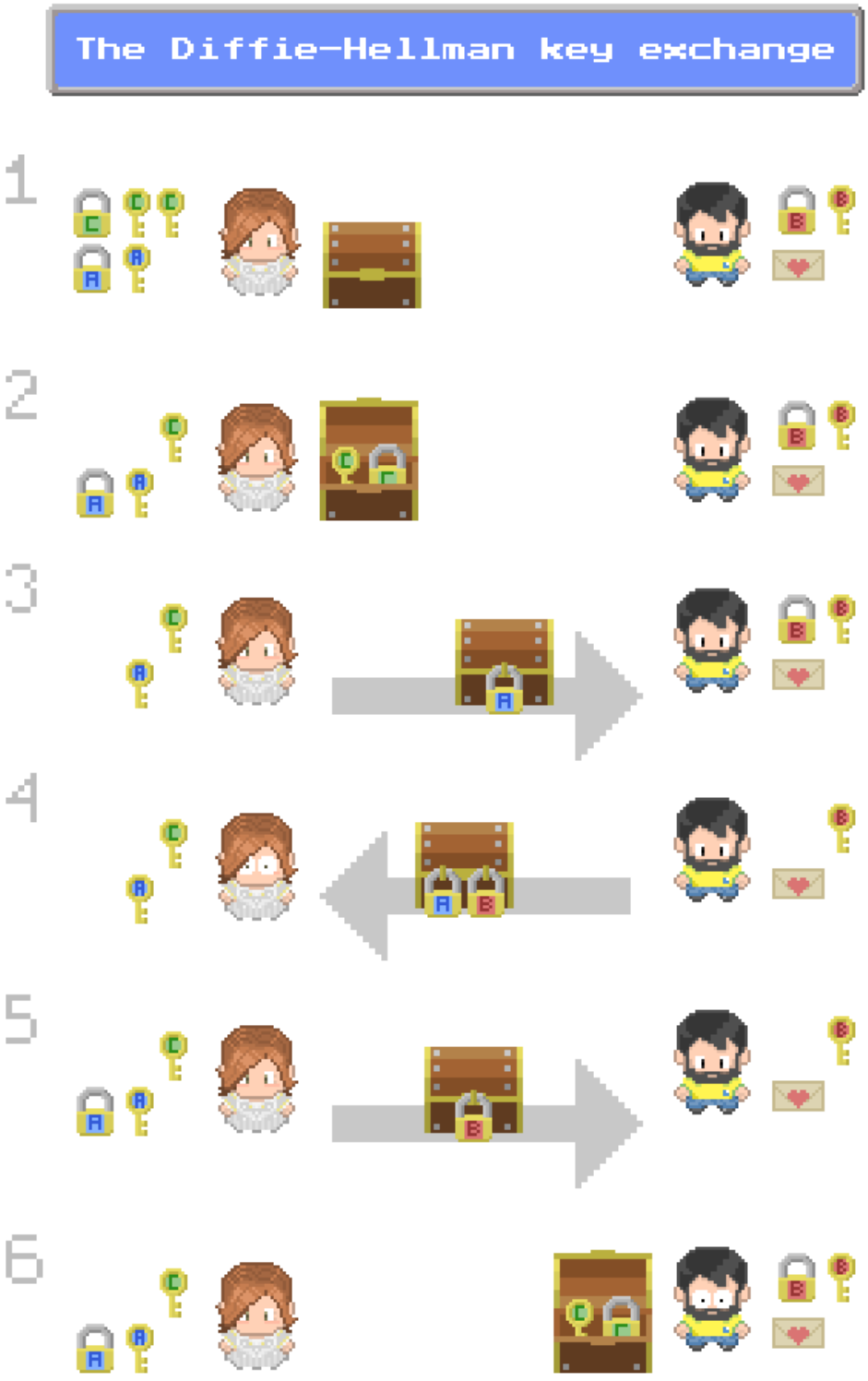

Abbildung 2.3: Der Diffie-Hellman Schlüsselaustausch verinfacht dargestellt

2.3 Einwegfunktionen

Ein solches »Schloss« kann in der Mathematik als eine Funktion betrachtet werden, welche sich zwar leicht berechnen lässt, die Umkehrung der Funktion allerdings sehr schwer ist. Ist man aber im Besitz einer »geheimen Zutat«, dem privaten Schlüssel, so muss sie sich doch umkehren lassen. Ähnlich wie bei einem Vorhängeschloss, welches sich zwar einfach schließen lässt, ohne einen Schlüssel lässt es sich aber nur schwer öffnen.

Eine solche Funktion wird als Einwegfunktion bezeichnet. Für die Definition einer solchen Funktion (»leicht« in eine Richtung, »schwer« in die entgegengesetzte Richtung berechenbar) muss zunächst die Frage beantwortet werden was eine »leicht« bzw. eine »schwer« lösbare Funktion ist.

Vgl. [3] 1.2 »Einwegfunktionen«

2.4 Die Polynomialzeit

Als Grenze zwischen einer einfach und einer schweren Funktion wird die Polynomialzeit verwendet. Es handelt sich dabei um die Zeit, die für die Berechnung einer Funktion benötigt wird, in Abhängigkeit mit der Problemgröße oder Schlüssellänge. Steigt die benötigte Zeit für einen größeren Schlüssel nicht mehr als eine Polynomfunktion der Form
$P(x) = a_n * x^n + ... + a_2 * x^2 + a_1 * x + a0$ so handelt es sich um eine einfach lösbare Funktion. Die Polynomgrad ist dabei ein Merkmal dafür, wie leicht die Funktion zu berechnen ist.

Steigt die benötigte Berechnungszeit aber zum Beispiel exponentiell (a^x) dann überschreitet sie diese Grenze der Polynomialzeit und die Funktion gilt als schwer lösbar.

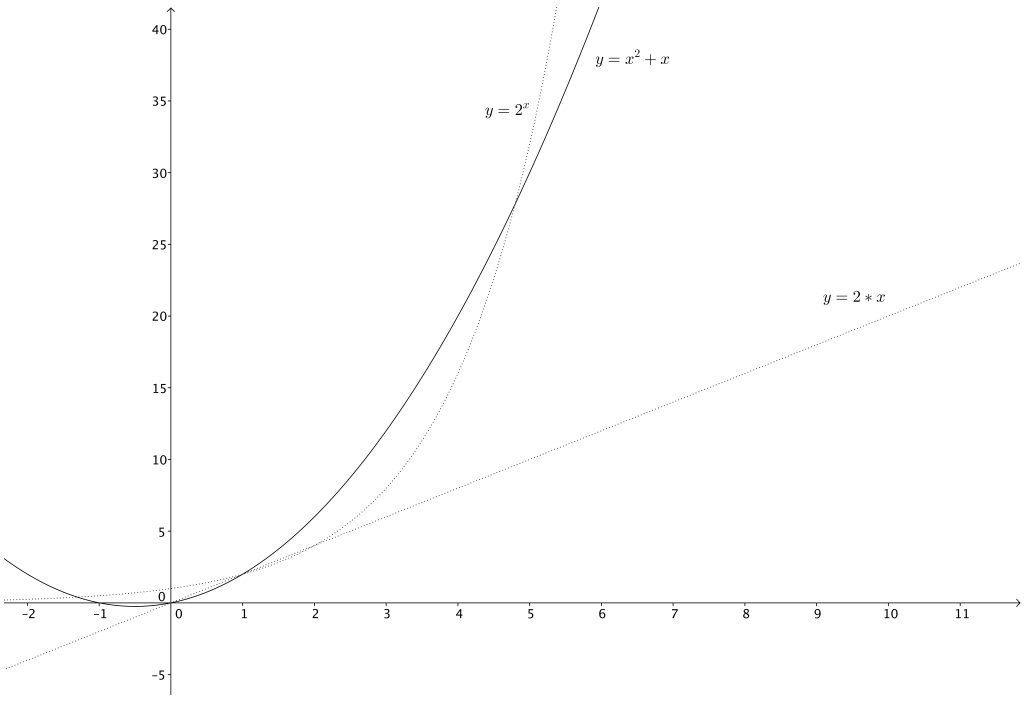

Abbildung 2.4: Beispiel einer Polynomialfunktion für verschiedene Funktionen.

Die Zeit und die Problemgröße können dann im Kontext des Computers zum Beispiel als die Nummer der benötigten Operationen und die Anzahl der Bits festgelegt werden. Ob eine Funktion deswegen innerhalb der Polynomialzeit berechnet werden kann, hängt deswegen auch von dem verwendeten Algorithmus ab.

Vgl. [12]

Obwohl die Polynomialzeit oft als geeignete Grenze zwischen praktisch lösbar und praktisch unlösbar angesehen wird, wachsen Polynome höheren Grades so schnell, dass sie eigentlich schon wieder als praktisch unlösbar eingestuft werden müssten.

Kapitel 3

Das RSA-Verfahren

Das RSA-Verfahen, benannt nach den drei Erfindern Rivest, Shamir und Adleman ist ein Algorithmus zur asymmetrischen Verschlüsselung einer Nachricht und der dazugehörigen Schlüsselfindung. Das Verfahren wurde 1977 basierend auf der Forschung von Rivest, Shamir und Adleman entwickelt. Die 3 Mathematiker beschäftigten sich am MIT mit der von Whitfield Diffie und Martin Hellmann entwickelten Theorie des Schlüsselaustausches.

Vgl. [1] S. 329ff; [4]

Abbildung 3.1: V. l. n. r.: Shamir, Rivest und Adleman

Obwohl in den 60er Jahren vom Government Communications Headquarter (einem britischen Nachrichten- und Geheimdienst der sich auf Kryptographie spezialisiert hat) bereits ein ähnlicher Algorithmus entwickelt wurde, konnten die drei Forscher ihren Algorithmus trotzdem zum Patent anmelden. Dies lag daran, dass das GCHQ den Algorithmus unter Geheimhaltung hielt und nicht wissenschaftlich publizierte.

Vgl. [1] S. 338

3.1 Schlüsselgenerierung

Beim RSA Verfahren bestehen sowohl der private als auch der öffentliche Schlüssel aus einem Zahlenpaar. Der private Schlüssel aus den Zahlen d und N, der öffentliche aus e und N, wobei N für beide Schlüssel identisch ist.

Um sich diese Werte zu berechnen werden zunächst zwei zufällige Primzahlen, p und q, gesucht, welche nicht identisch sein dürfen.

Aus diesen zwei Zahlen wird per Multiplikation der RSA-Modul berechnet:

$N = p * q$

Vgl. [4] S. 6

Danach wird für N die Eulersche-Phi-Funktion berechnet. Diese Funktion gibt an, wie viele Zahlen die kleiner als N sind, auch teilerfremd, also keinen gemeinsamen Teiler besitzen, sind. Bei der Berechnung der Eulerschen-Phi-Funktion kann auf zwei wichtige Eigenschaften der Funktion zurückgegriffen werden. Auf der einen Seite sind bei einer Primzahl alle Zahlen die kleiner als sie selbst sind teilerfremd zu ihr. Dadurch kann die Funktion auf $\Phi(N) = N - 1$ vereinfacht werden. Außerdem ist die Eulersche-Phi-Funktion eine multiplikative Funktion. Das bedeutet, dass $\Phi(p * q) = \Phi(p) * \Phi(q)$ ist.

Die Eulersche-Phi-Funktion von N kann also als $\Phi(N) = (p - 1) * (q - 1)$ angeschrieben und berechnet werden.

Für den Verschlüsselungsexponenten e wird daraufhin eine Zahl gesucht, die teilerfremd und kleiner als $\Phi(N)$ ist, es gilt also $1 < e < \Phi(N)$. Zu letzt wird für die Berechnung des Entschlüsselungsexponenten d eine Zahl gesucht, die folgende Kongruenz erfüllt:

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

Kongruenzen sind Beziehungen zwischen zwei Zahlen in Bezug zu einem Modulus oder Divisor. In diesem Fall ist die Kongruenz erfüllt, wenn $\frac{e*d}{\Phi(N)}$ den selben Rest ergibt wie $\frac{1}{\Phi(N)}$.

3.1.1 Beispiel

1. $p = 3$ und $q = 7$
2. $N = 21$
3. $\Phi(N) = 2 * 6 = 12$
4. $e = 5$
5. $d = 17$

Vgl. [13]

Der private Schlüssel lautet somit also $(d, N) = (17, 21)$ und der öffentliche Schlüssel $(e, N) = (5, 21)$.

3.2 Schlüssellängen

In der Praxis werden natürlich längere Schlüssel erzeugt. Die NIST (National Institute of Standards and Technology) stuft Schlüssel mit einer Länge von 2048 Bit (=256 Byte) als sicher bis 2030 ein. Sicher sind Schlüssel ab einer Länge von 3072 Bit (=384 Byte). Dies hängt damit zusammen, dass die Sicherheit der asymmetrischen Kryptographie darauf beruht, dass bestimmte Funktionen mit einer bestimmten Problemgröße praktisch unlösbar sind. Da Computer allerdings nach dem Mooreschen Gesetz ungefähr alle zwei Jahre ihre Rechenleistung verdoppeln, müssen die Problemgrößen ebenfalls mitwachsen.

Vgl. [5]

Zu bedenken ist hierbei, dass die Schlüssellängen normalerweise in Bit angegeben werden. Um sich diese Größen besser vorstellen zu können, kann man die Schlüssellängen aber auch in Dezimalstellen umrechnen. Mithilfe der Formel:

$n = \lceil b * \log_{10} 2 \rceil$

lässt sich somit zum Beispiel berechnen, dass ein Schlüssel mit $b = 2048$ Bits 617 Dezimalstellen besitzt. Die eckigen Klammern links und rechts geben dabei an, dass das Ergebnis der Berechnung auf die nächst größere Ganzzahl gerundet werden soll. Ein Schlüssel mit 3072 Bits besitzt also insgesamt 925 Dezimalstellen. Also zum Beispiel eine 1 mit 924 Nullen daran ($1 * 10^{924}$).

Vgl. [11] »Real Numbers«

3.3 Verschlüsselung

Zur Verschlüsselung einer Nachricht m wird der öffentliche Schlüssel (e, N) verwendet. Sie erfolgt nach der einfachen Formel:

$c = m^e \bmod N$

Wichtig ist hierbei allerdings, dass die Nachricht m kleiner als der RSA-Modul N sein muss. Dies ergibt sich, weil für den verschlüsselten Text der Rest der Division durch N verwendet wird und der Rest von m/N gleich dem Rest von $m + N/N$ ist und eine eindeutige Entschlüsselung somit nicht mehr möglich wäre.

Vgl. [4] S. 6

3.3.1 Beispiel

Um nun zum Beispiel die Zahl 4 zu Verschlüsseln, wird sie einfach in die Formel eingesetzt und die Formel berechnet.

1. $c = 4^5 \bmod 21$
2. $c = 1024 \bmod 21$
3. $c = 16$

Vgl. [13]

3.4 Entschlüsselung

Eine Verschlüsselte Nachricht c kann nun nur mehr mit dem privaten Schlüssel (d, N) entschlüsselt werden. Dazu wird die Nachricht in folgende Formel eingesetzt und anschließend berechnet:

$m = c^d \bmod N$

Vgl. [4] S. 6

3.4.1 Beispiel

1. $m = 16^{17} \bmod 21$
2. $m = 295147905179352825856 \bmod 21$
3. $m = 4$

Vgl. [13]

3.5 Anmerkung: Textnachrichten

Da im Normalfall Textnachrichten verschlüsselt übertragen werden, müssen diese logischerweise noch in eine Zahl übertragen werden. In der Informatik hat sich früher dafür das ASCII-Verfahren angeboten, welches insgesamt 128 Steuerzeichen und Buchstaben auf den Zahlen 0 - 127 abgebildet hat. Im Laufe der Zeit hat sich aber herausgestellt, dass mehr als 128 Zeichen benötigt werden. Heutzutage wird oft UTF8 beziehungsweise UTF16 verwendet. Dies hat mit der eigentlichen Verschlüsselungstechnik allerdings nichts zu tun, es handelt sich hierbei nur um eine Möglichkeite Texte in Zahlen umzuwandeln, beziehungsweise am Computer zu verarbeiten.

Vgl. [1] S. 298; [4] S. 6

3.6 Beweis des RSA-Verfahrens

Der Beweis, dass das RSA-Verfahren eine beliebige Nachricht verschlüsseln und anschließend wieder korrekt entschlüsseln kann, beruht auf zwei Sätzen. Mit Hilfe des erweiterten euklidischen Satzes und des kleinem Satz von Fermat kann jedoch nur bewiesen werden, dass eine verschlüsselte Nachricht wieder korrekt entschlüsselt werden kann, aber nicht, dass das RSA-Verfahren vor Angriffen sicher ist.

3.6.1 Erweiterter euklidischer Algorithmus

Der erweiterte Euklidische Algorithmus basiert auf der, seit der Antike durch den griechischen Mathematiker Euklid bekannten, Methode den größten gemeinsamen Teiler von zwei Zahlen zu bestimmen. Sind die zwei Zahlen teilerfremd zueinander (der größte gemeinsame Teiler ist also 1) so lässt sich der erweiterte Algorithmus anwenden. Mit diesem kann man das Inverse einer Kongruenz berechnen. Mit diesem Algorithmus lässt sich also zum Beispiel bestimmen, mit welchem x folgende Kongruenz korrekt ist:

$17 * x \equiv 1 \bmod 43$

Es wird also eine Zahl gesucht mit der 17 multipliziert werden kann, so dass der Rest der Division von $\frac{17*x}{43}$ gleich dem Rest von $\frac{1}{43}$ ist.

Dazu wird zunächst der Euklidische Algorithmus angewendet. Dabei wird jeweils die kleinere Zahl (in diesem Fall 17) so oft wie möglich von der größeren Zahl (im Beispiel 43) abgezogen und der Rest als Addition angeschrieben. Dies wird solange fortgesetzt, bis ein größter gemeinsamer Teiler gefunden wird.

$43 = 17 * 2 + 9$
$17 = 9 * 1 + 8$
$9 = 8 * 1 + 1$

Es hat sich also bewiesen dass der größte gemeinsame Teiler 1 ist. Nun wird diese Rechnung von unten nach oben wieder aufgelöst indem die einzelnen Gleichungen nach den Resten umgeformt werden und füreinander eingesetzt werden.

$1 = 9 - 8$
$8 = 17 - 9$
$1 = 9 - (17 - 9)$
$1 = 9 - 17 + 9$
$1 = 9 * 2 - 17$
$9 = 43 - 17 * 2$
$1 = (43 - 17 * 2) * 2 - 17$
$1 = 2 * 43 - 4 * 17 - 17$
$1 = 2 * 43 - 5 * 17$

Da im nächsten Schritt Modulo 43 gerechnet wird, kann das 2 * 43 aus der Gleichung gestrichen werden. Es bleibt also

$1 = -5 * 17$

übrig.

$\frac{17}{1} \bmod 43 \equiv -5$
$\frac{17}{1} \bmod 43 \equiv 38$

Da eine positive Zahl gesucht wird, wird zu -5 +43 addiert. Dies hat aufgrund der Modulo Rechnung keinen Einfluss auf die linke Seite der Gleichung. Die gesuchte Zahl lautet also 38. Der Rest von 38 durch 43 ist 1 und der Rest von 1 durch 43 ist ebenfalls 1, somit wurde eine valide Zahl für die Variable x gefunden.

Vgl. [16]; [17]

3.6.2 Kleiner Satz von Fermat

Beim kleinen Satz von Fermat handelt es sich um eine Spezialform des Satzes von Euler ($a^{\Phi(m)} \equiv 1 \bmod m$). Ist die Zahl m nämlich eine Primzahl p und a kein Vielfaches dieser Zahl p, dann kann die Eulersche-Phi-Funktion aufgelöst werden in $\Phi(p) = p - 1$. Es gilt also:

$a^{p-1} \equiv 1 \bmod p$

Vgl. [18]

3.6.3 Der Beweis

Es soll nun bewiesen werden, dass sich ein verschlüsselter Text wieder korrekt entschlüsseln lässt. Wie bereits bekannt lautet der Algorithmus zum verschlüsseln eines Klartextes:

$c = m^e \bmod N$

Um wieder den Klartext zu erhalten muss also die Potenzierung von m^e aufgehoben werden. Dies kann durch erneutes Potenzieren mit e^{-1} erreicht werden:

$(m^e)^{e^{-1}}$

Mithilfe der Potenzregel können die Klammern aufgelöst werden, es ergibt sich:

$m^{e*e^{-1}}$ bzw. $m^{e*\frac{1}{e}}$

$e * \frac{1}{e}$ ergibt 1, die Potenz löst sich also auf und es bleibt der Klartext m stehen. Als nächstes muss also bewiesen werden, dass es dieses e^{-1} gibt.

Für den nächsten Schritt muss gelten dass der $ggT(e, \Phi(N)) = 1$ ist. (Dies wurde ja bei der Schlüsselgenerierung bereits berücksichtigt). Nun kann man den Erweiterten Euklidischen Algorithmus anwenden. Dieser berechnet ja einerseits den größten gemeinsamen Teiler, aber auch noch zwei zusätzliche Zahlen s und t:

$ggT(a, b) = s * a + t * b$

In diese Formel können wir jetzt also mit e und $\Phi(N)$ einsetzen:

$1 = s * \Phi(N) + t * e$

Dies kann nun durch $\Phi(N)$ dividiert und umgeformt werden:

$t * e \equiv 1 \bmod \Phi(N)$

Somit ist t der gesuchte Wert von e^{-1} und es ist bewiesen dass dieser Wert überhaupt existiert.

Als nächstes soll bewiesen werden, dass es nur einen Wert für e^{-1} oder kurz d gibt:

$e * f \equiv 1 \bmod \Phi(N)$

Dies kann mit d multipliziert werden:

$e * f * d \equiv d \bmod \Phi(N)$

Da wir wissen dass $d * e \equiv 1 \bmod \Phi(N)$ ist, können wir d * e aus der Gleichung kürzen:

$f \equiv d \bmod \Phi(N)$

Somit ist aus bewiesen dass es nur einen Wert für d geben kann.

Zum Schluss kann nun noch die Relation zu den beiden Primzahlen (p und q) bewiesen werden. Dazu wird wieder von $1 = s * \Phi(N) + d * e$ ausgegangen. Nun kann $Phi(N)$ durch $(p-1) * (q-1)$ ausgetauscht werden. Außerdem kann nach d * e umgeformt werden:

$d * e = 1 - s * (p-1) * (q-1)$

Nun können beide Seiten mit dem Klartext potenziert werden:

$m^{d*e} = m^{1-s*(p-1)*(q-1)}$

Nun kann m^1 noch gekürzt werden und Modulo N (p*q) gerechnet werden:

$m^{s*(p-1)*(q-1)} \equiv 1 \bmod N$

Diese Gleichung entspricht dem kleinen Satz von Fermat, somit ist auch diese Behauptung bestätigt.
Vgl. [4] S. 7; [8] S. 16

3.7 Sicherheit

Die Sicherheit des RSA-Verfahrens beruht darauf, dass drei mathematische Probleme nicht in einer praktikablen Zeit berechnet werden können.

3.7.1 RSA-Problem

Sind der öffentliche Schlüssel (N, e) und der Geheimtext c gegeben, so könnte man die e-te Wurzel modulo N ziehen um den Klartext m zu berechnen. Mit aktuellen Algorithmen und Computern ist dies aber als praktisch unlösbar einzustufen.
Vgl. [4] S. 13

3.7.2 RSA-Schlüsselproblem

Basierend auf dem öffentlichen Schlüssel (N, e) könnte man theoretisch auf den geheimen Schlüsselbestandteil d schließen. Dazu müsste allerdings die Eulersche-Phi-Funktion ohne Kenntnis der Primfaktoren p und q berechnet werden. Auch dieses Problem kann mit aktuellen Erkenntnissen nicht praktisch gelöst werden.
Vgl. [4] S. 12

3.7.3 Faktorisierungsproblem

Gelingt es den RSA-Modul N wieder in die zwei Primfaktoren p und q zu zerlegen, könnte ebenfalls der private Schlüssel berechnet werden. Da nicht bewiesen ist, dass es sich bei diesem Problem um ein schwieriges Problem handeln muss, ist dieses Problem von besonderem Interesse. Mit dem aktuellen Wissenstand und modernen Computern lässt sich aber auch dieses Problem nur mit sehr viel Zeitaufwand berechnen.

Vgl. [4] S. 11

Im Dezember 2009 gelang es einem Team von Wissenschaftlern einen 768 Bit Schlüssel nach 3 jähriger Berechnungsdauer in seine zwei Primfaktoren aufzuteilen. Dies ist mitunter ein Grund, warum heutzutage Schlüssellängen von 2048 Bit verwendet werden.

Vgl. [5]

Solange diese drei Probleme nicht in Polynomialzeit lösbar sind, kann RSA, mit angemessener Schlüssellänge, als sicher eingestuft werden.

3.8 Padding

Ein weiteres wichtiges Problem des RSA-Verfahrens ist, dass es sich deterministisch verhält. Sind die Inhalte mehrerer verschlüsselter Nachrichten bekannt, so könnte versucht werden durch Probieren auf den privaten Schlüssel zu stoßen.

Um diesem Problem entgegen zu wirken, wird ein so genanntes Padding verwendet. Dabei wird die Nachricht m um eine zufällige Nachricht r erweitert. Diese Nachricht r kann nach verschiedenen Mustern generiert werden, ein wichtiger Bestandteil sind meist die Länge des Paddings sowie eine oder mehrere Zufallszahlen.

Da das Padding bei jedem Verschlüsselungsvorgang unterschiedlich ist, verhält sich die Verschlüsselungsmethode nicht mehr deterministisch. Es kann also nicht mehr versucht werden, durch den Inhalt der verschlüsselten Nachricht an den privaten Schlüssel zu gelangen.

Vgl. [9] S. 10f

3.9 Hybride Kryptographie

In der Praxis hat das RSA-Verfahren leider einen entscheidenden Nachteil: es ist rund drei Größenordnungen (10^3) langsamer als symmetrische Verschlüsselungsmethoden. Eine beliebte Möglichkeit um also Verbindungen per asymmetrischer Kryptographie zu schützen, aber auch die Geschwindigkeit von symmetrischer Kryptographie zu erreichen ist: einen Schlüssel für die symmetrische Kryptographie zu generieren, und nur diesen Schlüssel per RSA an den Empfänger zu senden. Die restlichen Daten werden daraufhin dann mit diesem symmetrischen Schlüssel verschlüsselt und entschlüsselt.

Allerdings gilt in diesem Zusammenhang »Eine Kette ist nur so stark wie ihr schwächstes Glied«. Ist also zum Beispiel das symmetrische Verschlüsselungsverfahren nicht sicher, so nützt die RSA-Verschlüsselung des Schlüssels nichts. Dies kann zum Beispiel der Fall sein, wenn der symmetrische Schlüssel nicht ausreichend zufallsbasiert ist, sondern vorhersagbar ist.

Vgl. [10] S. 58

Kapitel 4

Verwendung von RSA

4.1 PGP

Eine der ersten Anwendung von RSA ist die Software PGP (Kurz für Pretty Good Privacy). Sie wurde 1991 von Phil Zimmermann kostenlos zur Verfügung gestellt. Mit der Software ist es möglich eine E-Mail Nachricht sowohl zu signieren, als auch komplett zu verschlüsseln. Bei einer Signatur wird die Nachricht so zusagen ≫Unterschrieben≪. Damit ist es für den Empfänger möglich die Herkunft der Nachricht zu verifizieren. Es wird dabei eine kurze Repräsentation der E-Mail genommen, ein sogenannter Hash, welcher dann mit dem privaten Schlüssel des Senders verschlüsselt wird. Der Hash kann somit also von jedem Menschen mit Hilfe des öffentlichen Schlüssels des Senders entschlüsselt werden. Danach wird überprüft ob das entschlüsselte Ergebnis dem Hash der empfangenen Nachricht entspricht. Ist dies der Fall, so stammt die E-Mail tatsächlich vom angegebenem Sender und sie wurde nicht im Nachhinein verändert.

Bei PGP wird für jede Nachricht ein zufälliger symmetrischer Schlüssel generiert mit dem die Nachricht verschlüsselt wird. Dieser Schlüssel wird dann wie im Kapitel 3.9 besprochen mithilfe von RSA verschlüsselt und ebenfalls an die E-Mail angehängt. Dies ermöglicht es auch eine E-Mail an mehrere Empfänger zu senden indem mehrere Schlüssel an eine E-Mail angehängt werden. Die Schlüssel werden dabei immer mit dem öffentlichen Schlüssel des Empfängers verschlüsselt. Der Empfänger kann den Schlüssel, mit dem die ganze Nachricht verschlüsselt ist, dann mit seinem privaten Schlüssel entschlüsseln und verwenden.

Vgl. [1] S. 353ff

Abbildung 4.1: Phil Zimmermann, Entwickler von PGP

4.2 Die Crypto-Kriege

In den 1970er Jahren entbrannte ein regelrechter Kampf um die sichersten Verschlüsselungstechniken. Während auf der einen Seite zahlreiche Mathematiker vor allem an den Computeruniversitäten Amerikas an immer neuen und sichereren Methoden forschten, Nachrichten zu verschlüsseln, hatte vor allem die amerikanische Regierung ein großes Problem damit. Sie stand im Kalten Krieg gegen Russland und wolle nicht, dass die Russen ähnlich gute Verschlüsselungstechniken verwenden konnten.

Auf der anderen Seite entstand in dieser Zeit auch die Protestbewegung in Amerika, Hippies genannt. Der Regierung war es auch hier ein großes Anliegen den Nachrichtenaustausch der verschiedenen Organisationen mitlesen zu können um mögliche Aufstände, etc. frühzeitig erkennen zu können.

Die US-Regierung klassifizierte also Verschlüsselungsverfahren als Waffen. Sie fielen also nun nicht nur unter das US-Exportgesetz und durften nicht einfach exportiert werden, sie konnten auch innerhalb Amerikas eingeschränkt werden.

In den 1990er Jahren versuchte die US-Regierung den sogenannten Clipper-Chip einzuführen. Er wurde von der NSA entwickelt und sollte es Firmen ermöglichen einen Diffie-Hellman Schlüsselaustausch einfacher zu implementieren. Allerdings hatte der Clipper-Chip einen kleinen Hacken eingebaut: Jeder Schlüssel musste auch bei einer Behörde hinterlegt werden. Wollte nun zum Beispiel die NSA ein Telefon abhören, musste sie nur zu der Behörde gehen und um den Schlüssel ansuchen.

Gleichzeitig wurde jedoch PGP veröffentlicht, ein System das ganz ohne zentrale Stelle auskommt, es für die NSA also erheblich komplizierter machen würde Nachrichten abzuhören. Doch es gab für Zimmermann ein großes Problem mit PGP. Einerseits musste er für RSA Lizenzgebühren zahlen (ein Grund warum PGP aktuell mit dem Elgamal-Algorithmus arbeitet), anderseits durfte er PGP nicht aus den USA-Exportieren. Zimmermann fand jedoch eine Lösung für das Exportproblem: er druckte den Quellcode in Buchform aus und schickte so ein normales Buch nach Europa. In Europa fanden sich in kurzer Zeit zahlreiche Helfer die den gesamten Quellcode abtippten und neu kompilierten.

Außerdem wurde von der amerikanischen Regierung die Schlüssellänge bei den verschiedenen Verfahren auf 40 Bit regelementiert. Für die NSA dauerte es nur einige Millisekunden zu entschlüsseln. Dies traf aber auch zunehmend auf Widerstand, nicht zuletzt von der Industrie und von den Banken die Onlinebanking anbieten wollten.

So kam es dass Ende der 1990er Jahre bis Anfang der 2000er Jahre die Regelungen in Amerika und Europa gelockert wurden. Es kann nun jeder nach belieben seine digitale Kommunikation verschlüsseln. Somit sind auch Anwendungen wie Onlinebanking alle Türen geöffnet worden. Ein wichtiger Meilenstein in der Geschichte des Internets.

Vgl. [1], 374ff; [14]; [15]

Quellenangaben

1. Singh, Simon: Geheime Botschaften: Die Kunst der Verschlüsselung von der Antike bis in die Zeiten des Internet: dtv Verlag 2012
2. Wikipedia: Caesar-Verschlüsselung http://de.wikipedia.org/wiki/Caesar-Verschl%C3%BCsselung (Letzter Aufruf: 23. 11. 2013)
3. Lukas Dölle: Einwegfunktionen Variationen und Beispiele http://www2.informatik.hu-berlin.de/Forschung_Lehre/algorithmenII/Lehre/WS2002-2003/Perlen/Variationen.pdf (Letzter Aufruf: 23. 11. 2013)
4. Rivest, Ronald; Shamir, Adi; Adleman, Leonard: A Method for Obtaining Digital Signatures and Public-Key Cryptosystems http://people.csail.mit.edu/rivest/Rsapaper.pdf (Letzter Aufruf: 23. 11. 2013)
5. http://www.emc.com/emc-plus/rsa-labs/historical/rsa-768-factored.htm (Letzter Aufruf: 23. 11. 2013)
6. http://www.keylength.com/en/4/ (Letzter Aufruf: 23. 11. 2013)
7. Julius Leber Schule: Der Beweis des RSA-Verfahrens http://www.hh.schule.de/julius-leber-schule/melatob/beweisrsa.html (Letzter Aufruf: 23. 11. 2013)
8. Ge, Yimin: Die Mathematik von RSA http://www.yimin-ge.com/doc/mathematik_von_rsa.pdf (Letzter Aufruf: 23. 11. 2013)
9. Petersen, Sebastian: Zur Sicherheit von RSA http://www.mathematik.uni-kassel.de/mathfb16/WS11_12/DiskStruk/UebungDiskStrukII/beamer_rsa.pdf (Letzter Aufruf: 23. 11. 2013)
10. Das CrypTool-Team: Asymmetrische Kryptologie am Beispiel RSA entdecken https://www.cryptoportal.org/data/Asymmetrische%20Kryptologie%20am%20Beispiel%20RSA%20entdecken_v1.1.pdf (Letzter Aufruf: 23. 11. 2013)
11. Nam Sun Wang: Computer Methods in Chemical Engineering - Number System http://www.eng.umd.edu/~nsw/chbe250/number.htm (Letzter Aufruf 24. 11. 2013)
12. Khan Academy: Time Complexity https://www.khanacademy.org/math/applied-math/cryptography/modern-crypt/p/time-complexity-exploration (Letzter Aufruf: 24. 11. 2013)
13. JL Popyack: RSA Calculator https://www.cs.drexel.edu/~jpopyack/IntroCS/HW/RSAWorksheet.html
14. foundation for information policy research: The Crypto Wars Are Over! http://www.fipr.org/press/050525crypto.html (Letzter Aufruf: 24. 11. 2013)
15. Moechel, Erich: Die Crypto Wars - WikiLeaks 1.0 http://moechel.com/doqs/wikileaks_eins_null.pdf (Letzter Aufruf: 24. 11. 2013)

16. Embacher, Franz: Der erweiterte Euklidische Algorithmus `http://www.mathe-online.at/materialien/Franz.Embacher/files/RSA/Euklid.html` (Letzter Aufruf: 8. 12. 2013)

17. Wikipedia: Erweiterter Euklidischer Algorithmus `http://de.wikipedia.org/wiki/Erweiterter_euklidischer_Algorithmus` (Letzter Aufruf 8. 12. 2013)

18. Wikipedia: Kleiner Satz von Fermat `http://de.wikipedia.org/wiki/Kleiner_fermatscher_Satz` (Letzter Aufruf: 8. 12. 2013)

Abbildungsverzeichnis