Kryptographie

Teilnehmer:

Kevin Huber	Georg-Forster-Oberschule
Philippe Gruse	Georg-Forster-Oberschule
Vera Koldewitz	Georg-Forster-Oberschule
Philipp Jakubahs	Georg-Forster-Oberschule
Julian Zimmert	Hermann-Hesse-Oberschule
Maximilian Werk	Heinrich-Hertz-Oberschule

Gruppenleiter:

Ulf Kühn	Humboldt-Universität zu Berlin.

Bei der RSA-Verschlüsselung handelt es sich um ein asymmetrisches Codierungsverfahren, da zwei unterschiedliche Schlüssel zum Ver- und Entschlüsseln existieren, der sogenannte Public Key (öffentlicher Schlüssel) sowie der Private Key (privater Schlüssel).

Dabei veröffentlicht der Empfänger den Public Key, mit dem der Absender seine Nachricht an den Empfänger mit einem öffentlich bekannten Verfahren verschlüsselt. Der Empfänger kann mit dem nur ihm bekannten Private Key die verschlüsselte Nachricht entschlüsseln. Der Clou ist nun, dass nur mit dem passenden Private Key die Nachricht wieder entschlüsselt werden kann. Namensgebend für das Verfahren waren jeweils die Anfangsbuchstaben der Nachnamen der drei Entwickler Ron Rivest, Adi Shamir und Leonard Adleman, welche 1977 das RSA-Prinzip hervorbrachten.

1 Grundlagen

1.1 Der euklidische Algorithmus

1.1.1 Beispiel

Zu gegebenen zwei ganzen Zahlen a, b funktioniert der euklidische Algorithmus nach folgendem Prinzip:

$$\begin{aligned}
a &= q_1 \cdot b + r_1 && mit \quad 0 < r_1 < |b| \quad (\Rightarrow b \neq 0),\\
b &= q_2 \cdot r_1 + r_2 && mit \quad 0 < r_2 < r_1,\\
r_1 &= q_3 \cdot r_2 + r_3 && mit \quad 0 < r_3 < r_2,\\
r_2 &= q_4 \cdot r_3 + r_4 && mit \quad 0 < r_4 < r_3,\\
&\vdots\\
r_{n-2} &= q_n \cdot r_{n-1} + r_n && mit \quad 0 < r_n < r_{n-1},\\
r_{n-1} &= q_{n+1} \cdot r_n.
\end{aligned}$$

Da die Reste immer kleiner werden, bricht der Algorithmus in der Tat nach n Schritten ab. Der letzte Rest r_n ist der größte gemeinsame Teiler(in Zeichen: (a, b)) von a und b. Offensichtlich ist r_1 eine Linearkombination von a und b; nämlich $r_1 = a - q_1 \cdot b$. Nach Multiplikation von r_1 mit einer Konstanten erhält man wieder eine Linearkombination. Eine Addition mit $x \cdot b$ ergibt logischerweise wieder eine Linearkombination von a und b. Damit muss auch r_2 eine Linearkombination sein. Nach Weiterführung dieser Argumentation ergibt sich, dass auch r_n Linearkombination ist. Verallgemeinert folgt:

1.1.2 **Satz** (Der euklidische Algorithmus)

Seien $a, b \in \mathbb{Z}, b \neq 0$. Dann gibt es zwei Zahlen $x, y \in \mathbb{Z}$, so dass gilt:

$$(a, b) = x \cdot a + y \cdot b.$$

Das heisst, der ggT von a und b lässt sich immer als (ganzzahlige) Linearkombination von a und b schreiben.
Sind speziell a und b teilerfremd, dann gibt es zwei Zahlen x und y, so dass gilt:

$$1 = x \cdot a + y \cdot b.$$

Beispiel:

$$\begin{aligned} 7514 &= 1 \cdot 5057 + 2457 \\ 5057 &= 2 \cdot 2457 + 143 \\ 2457 &= 17 \cdot 143 + 26 \\ 143 &= 5 \cdot 26 + 13 \\ 26 &= 2 \cdot 13. \end{aligned}$$

Ein mögliche Linearkombination ist: $13 = 263 \cdot 5057 - 177 \cdot 7514$.

1.2 Kleiner Satz von Fermat und Verallgemeinerung von Euler

1.2.1 Satz

Falls p eine Primzahl ist und $a \in \mathbb{Z}$ nicht Vielfaches von p ist, gilt:

$$a^{(p-1)} \equiv 1 \mod p.$$

Diesen Satz beweisen wir nun zweimal: einmal durch vollständige Induktion und einmal durch einen elementaren Beweis, welcher zunächst nichts mit dem eigentlichen Thema zu tun hat.

Beweis durch vollständige Induktion

Induktionsanfang: $a = 1$

$$1^p \equiv 1 \mod p.$$

Induktionsvoraussetzung:

$$a^p \equiv a \mod p$$

Induktionsschritt:

$$\begin{aligned} (a+1)^p &= \sum_{i=0}^{p} \binom{p}{i} a^i = a^p + 1 + \sum_{i=1}^{p-1} \binom{p}{i} a^i = a^p + 1 + \sum_{i=1}^{p-1} \frac{p!}{(p-i)! \cdot i!} a^i \\ &\equiv a^p + 1 \mod p \\ \iff \quad (a+1)^p &\equiv a^p + 1 \equiv a + 1 \mod p \end{aligned}$$

□

Elementarer Beweis

Vera möchte Ketten basteln. Dafür stehen ihr a Farben zur Verfügung. Da es ein mathematisch interessiertes Mädchen ist, fragt es sich, wie viele verschiedene Ketten sie aus p Perlen basteln kann. Sie mag besonders Primzahlen und deshalb denkt sie im Weiteren nur über p's nach, welche prim sind. Außerdem sind einfarbige Ketten doof.
Im folgenden werden offene Ketten weiterhin als Ketten und geschlossene Ketten als Ringe bezeichnet.
Zunächste stellt sie fest, dass sie a^p verschiedene Ketten basteln kann, da sie p mal überlegen muss, welche der a Farben sie benutzt. Ihr fällt auf, dass sie aber noch a Ketten abziehen muss, weil diese einfarbig sind. Somit kann sie $a^p - a$ mehrfarbige Ketten erstellen.
Vera denkt: Ringe sind eigentlich viel toller!
Aus diesem Grund möchte sie jetzt auch noch wissen, wie viele Ringe sie aus den Ketten basteln kann. Natürlich ist die Frage, wie viele Ketten denselben Ring ergeben, mit der Frage, in wie viele verschiedene Ketten sie einen Ring zerteilen kann, gleich. Sie kann den Ring an genau p Stellen trennen und somit p Ketten erhalten. Sind diese aber auch alle unterschiedlich? Wären zwei oder mehr gleiche dabei, so müssten die Perlen auf dem Ring periodisch angeordnet sein. Da p aber eine Primzahl ist, müssen die Periodenlängen 1 oder p sein. Somit sind tatsächlich alle Ketten, die man aus einem Ring bilden kann, unterschiedlich. Anders formuliert heißt dies, dass sie die Anzahl der möglichen Ketten durch p teilen muss, um die Anzahl der möglichen Ringe zu erhalten. Vera kann also $\frac{(a^p-a)}{p}$ Ringe basteln. Da sie aber in einer natürlichen Welt lebt, gibt es nur eine natürliche Anzahl an Ringen. Somit muss $a^p - a$ durch p teilbar sein. Nun möchte sie aber auch einmal mathematisch sein. Das heißt, es folgt:

$$a^p - a \equiv 0 \mod p \quad \Longleftrightarrow \quad a^p \equiv a \mod p$$

Sie freut sich. Da sie nicht nur ein verspieltes Mädchen, sondern auch ein mathematisch gebildetes ist, sieht sie, dass sie damit den kleinen Satz von Fermat hergeleitet hat.

Anmerkung: Dieser Beweis ist, so unwahrscheinlich es auch klingt, tatsächlich korrekt und man kann mit ihm den kleinen Satz von Fermat beweisen. Die einzige Einschränkung ist, dass es Vera wohl schwer fallen wird eine negative Anzahl an Farben zu haben.

1.2.2 **Satz** (Verallgemeinerung von Euler)

Seien p, q verschiedene Primzahlen und $n = p \cdot q$. Dann gilt für beliebiges $a \in \mathbb{Z}$ und $k \in \mathbb{N}$:

$$a^{k(p-1)(q-1)+1} \equiv a \mod n$$

Beweis

Nach dem kleinen Satz von Fermat gilt:

$$\begin{aligned} a^{p-1} &\equiv 1 \mod p \\ a^{q-1} &\equiv 1 \mod q \end{aligned}$$

Nach Umformungen folgt:

$$\begin{aligned} \left(a^{p-1}\right)^{k(q-1)} &\equiv 1 \mod p & \qquad \left(a^{q-1}\right)^{k(p-1)} &\equiv 1 \mod q \\ a^{k(p-1)(q-1)} &\equiv 1 \mod p & \qquad a^{k(q-1)(p-1)} &\equiv 1 \mod q \\ a^{k(p-1)(q-1)+1} &\equiv a \mod p & \qquad a^{k(p-1)(q-1)+1} &\equiv a \mod q \end{aligned}$$

Es ergibt sich:

$$a^{k(p-1)(q-1)+1} = a + k \cdot p = a + l \cdot q \quad \Longleftrightarrow \quad k \cdot p = l \cdot q$$

Da p und q Primzahlen sind, folgt $k = j \cdot q$. Damit muss gelten:

$$a^{k(p-1)(q-1)+1} = a + j \cdot q \cdot p = a + j \cdot n \quad \Longrightarrow \quad a^{k(p-1)(q-1)+1} \equiv a \mod n.$$

□

2 Das RSA-Verfahren

2.1 Einführung

Das RSA-Verfahren beruht darauf, dass das Produkt zweier Primzahlen leicht zu bilden ist, die Faktorisierung des Ergebnisses allerdings einen so enormen Aufwand hat, dass die Faktorisierung bei genügend großen Primzahlen trotz des Einsatzes von Computern nahezu unmöglich ist.
Im Folgenden werden die Primzahlen mit p und q und das Produkt mit n bezeichnet.

Außerdem werden noch zwei weitere Zahlen benötigt, welche c (chiffrieren) und d (dechiffrieren) genannt werden und folgende Bedingung erfüllen:

$$c \cdot d \equiv 1 \mod (p-1)(q-1)$$

Anders formuliert heißt dies, daß es ein $k \in \mathbb{N}$ gibt mit:

$$c \cdot d = 1 + k(p-1)(q-1) \tag{1}$$

2.2 Verfahren der Ver- und Entschlüsselung

Mit den oben entwickelten Zahlen c und d wird die Ver- und Entschlüsselung durchgeführt. Das Besondere ist, dass man c zum Verschlüsseln beliebig veröffentlichen kann und es somit ein öffentlicher Schlüssel (public Key) ist. Dabei wird eine gegebenen Botschaft (Message) m, zunächst in ASCII-Zeichen umgewandelt. Danach wird diese in Teilnachrichten zerlegt, welche jeweils nicht größer als n sein dürfen. Diese werden einzeln wie folgt verschlüsselt:

$$m_V = R_n(m^c); \tag{2}$$

hierbei ist $R_n(m^c)$ der Rest von m^c beim der Division durch n. Diese Verschlüsselung kann jeder, der den public Key kennt, durchführen, jedoch kann nur derjenige, dem n und d bekannt sind, die Nachricht wieder entschlüsseln:

$$m_E = R_n(m_V^d). \tag{3}$$

Es bleibt zu zeigen, dass die ursprüngliche Nachricht gleich der Entschlüsselten ist.

2.3 Nachweis der Funktionalität des RSA-Verfahrens

Nach Einsetzen von (2) in (3) folgt:

$$m_E = R_n((m^c)^d) \quad \Longleftrightarrow \quad m_E = R_n(m^{c \cdot d})$$

Nach (1) ergibt sich:

$$m_E = R_n(m^{1+k(p-1)(q-1)})$$

Nach Satz 1.2.2 folgt:

$$m_E \equiv m^{1+k(p-1)(q-1)} \equiv m \mod n$$

Da sowohl m_E (Rest nach Teilung durch n) als auch m (extra so gewählt) kleiner als n sind, muss gelten:

$$m_E = m$$

□

2.4 Wahl von c und d

Es bleibt die Frage, wie c und d konstruiert werden können.
Dafür wähle man zunächst eine zu $r = (p-1)(q-1)$ teilerfremde natürliche Zahl $c > 1$. Da somit $(c, r) = 1$ gilt, kann man unter Verwendung des euklidischen Algorithmus zwei ganze Zahlen u, v finden, so dass gilt:

$$1 = u \cdot c + v \cdot r \tag{4}$$

Wenn $u > 0$ gilt, muss $v < 0$ sein, weil $c, r > 1$. Folglich kann man schreiben:

$$u \cdot c = 1 + (-v) \cdot r.$$

Damit ist (1) mit $d = u$ und $k = -v$ erfüllt.
Wenn $u < 0$ gilt, lässt sich (4) wie folgt schreiben:

$$\begin{aligned} 1 &= (u + lr)c + (v - lc)r \\ \Leftrightarrow (u + lr)c &= 1 + (lc - v)r \end{aligned}$$

Wenn man $l \in \mathbb{N}$ genügend großwählt, ist (4) mit $d = u + lr$ und $k = lc - v$ erfüllt.
Somit sind zwei zur Ver- und Entschlüsselung geeignete Zahlen c und d gefunden.

2.5 Die technischen Vorteile von RSA - Die Sicherheit

Ein Verschlüsselungsverfahren lässt sich in erster Linie nach seiner Sicherheit bewerten. Der technische Aufwand der Realisierung ist aber auch ein nicht unwesentlicher Faktor bei der Wahl einer geeigneten Verschlüsselung. Die Vorteile von RSA sind speziell:

- Es existiert eine ausreichend große Zahl großer Primzahlen (>200 Stellen), so dass man genügend verschiedene Schlüssel generieren kann. Zum Beispiel existieren mehr als 10^{197} verschiedene 200-stellige Primzahlen.
- Man kann den Schlüssel beliebig sicher machen, indem man einfach größere Primzahlen wählt.
- Der Algorithmus ist relativ einfach, da vor allem multipliziert und addiert wird. Das rechnen mit Modulo lässt sich auch nahezu vollständig durch Multiplikation und Addition realisieren. Dies ist für Prozessoren von enormen Vorteil, da es nicht möglich ist, die Division ähnlich effektiv wie die anderen Grundrechenarten durchzuführen.
- Der wohl entscheidenste Vorteil von RSA ist allerdings, dass es bei großen Primzahlen quasi unmöglich ist, das n wieder zurückzufaktorisieren.