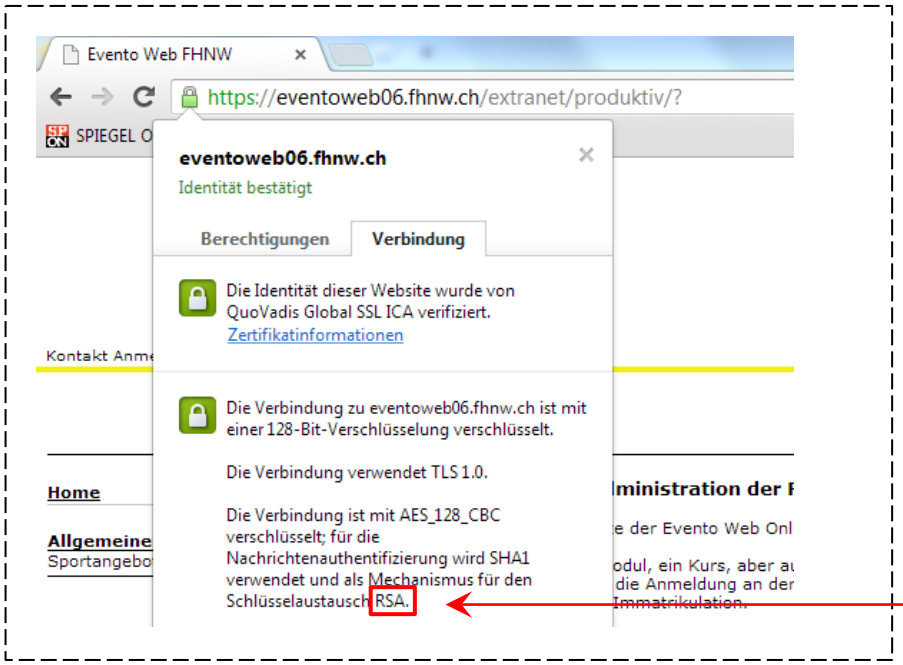
Zahlentheorie

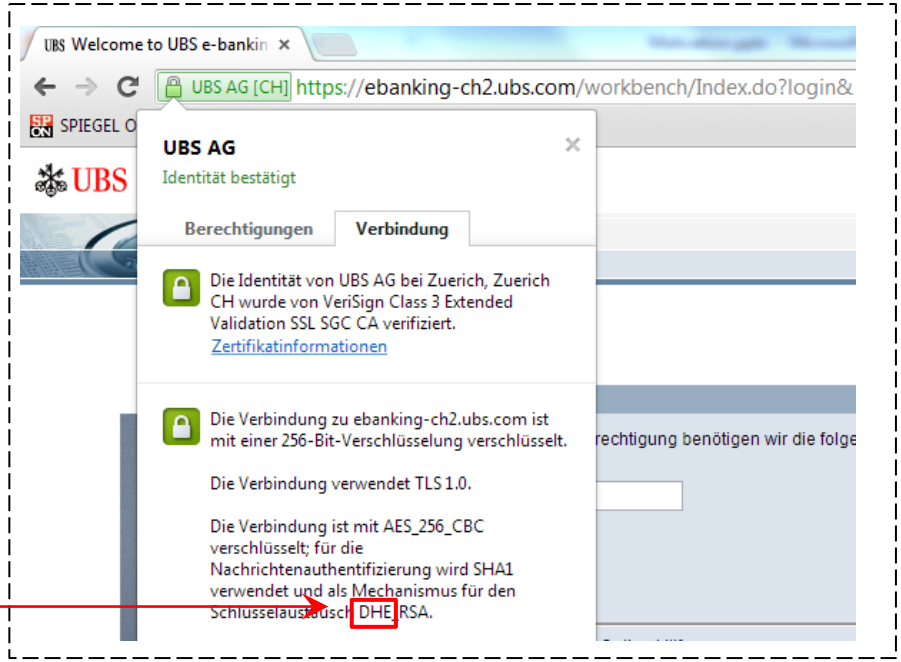
Mathematik für die Datenkommunikation

Prof. Dr. Andreas Vogt

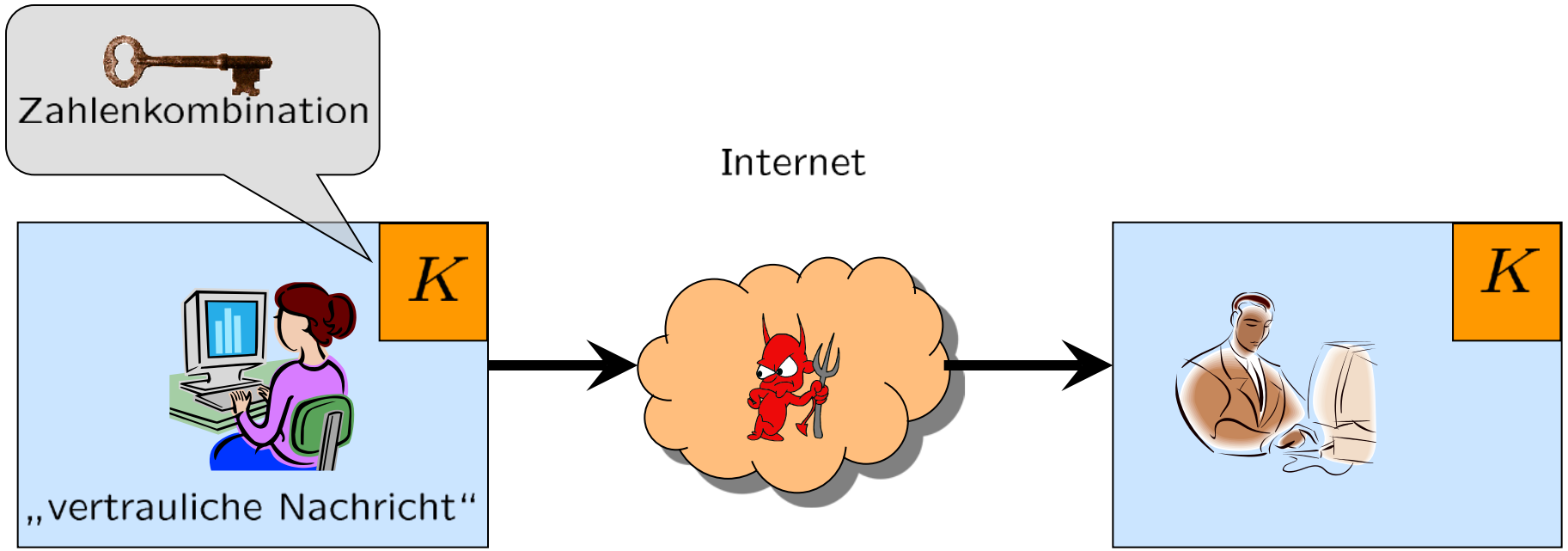
Fachhochschule
Nordwestschweiz

Ziel: Funktionsweise und Grundlagen von





Symmetrische Verschlüsselung



„vertrauliche Nachricht“

Symmetrische Verschlüsselung

Nachteile:

- Schlüsselaustausch problematisch
- Schlüsselexplosion
  (Je zwei Kommunikationsteilnehmer benötigen eigenes Schlüsselpaar.)

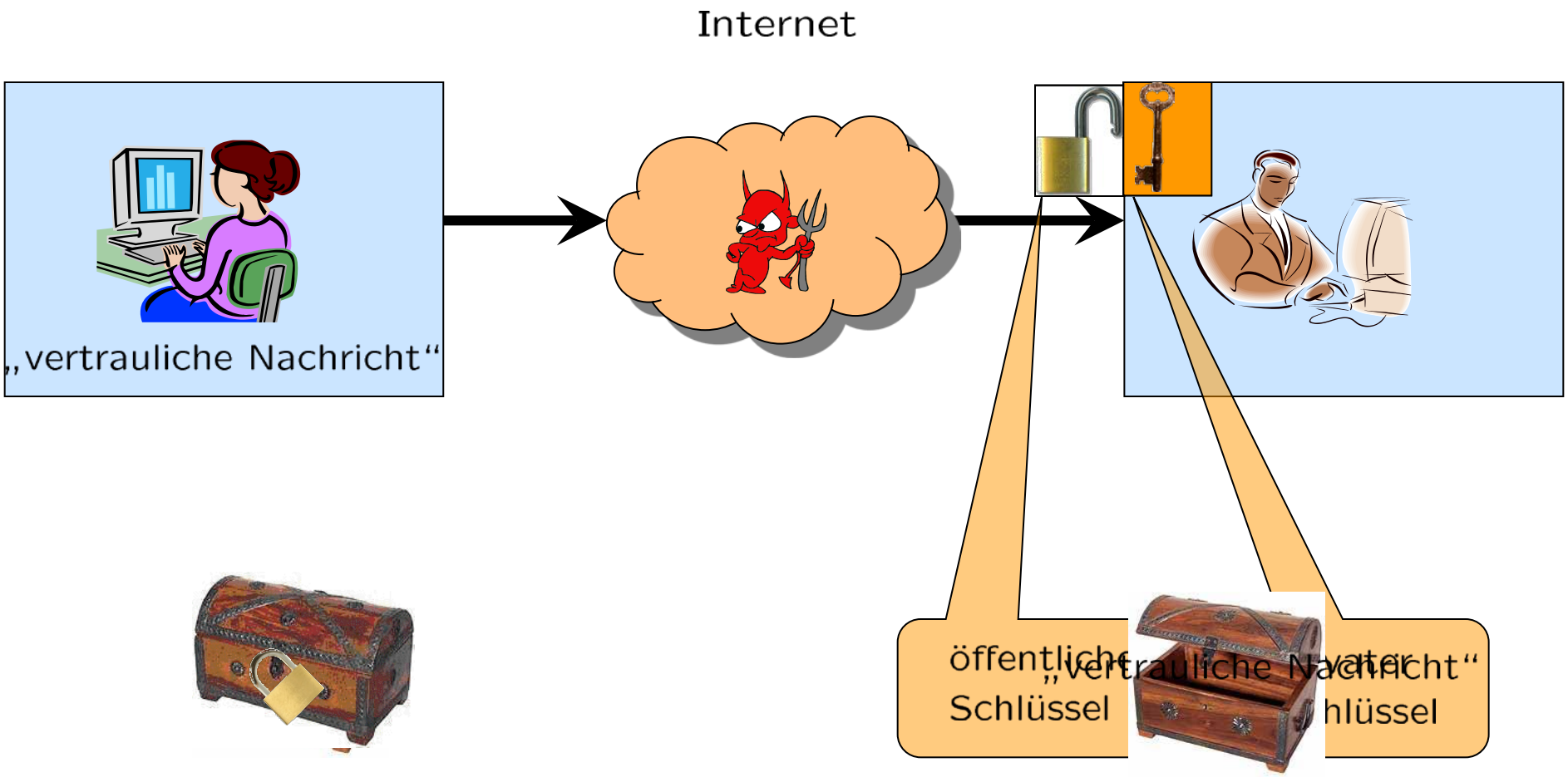
Vorteil:

- sehr effizient möglich

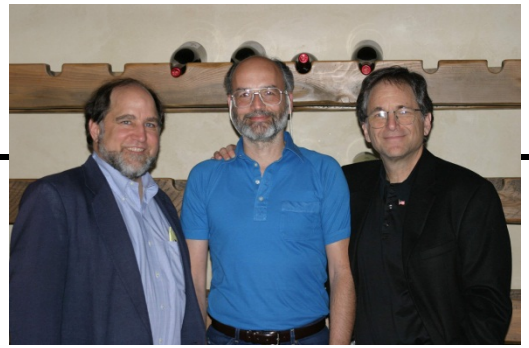
Die Behandlung moderner symmetrischer Verschlüsselungsverfahren (wie etwa AES) erfolgt im Modul kryg/krysi.

Obige Nachteile sind bei asymmetrischen Verfahren weitgehend gelöst.

Asymmetrische Verschlüsselung



Das Kryptosystem: RSA



Ron Rivest Adi Shamir Len Adleman

Mit ElGamal das wichtigste asymmetrische Kryptosystem!

Privater Schlüssel: (n, d)

Öffentlicher Schlüssel: (n, e)

$n = p \cdot q$ mit Primzahlen $p \neq q$,

$e, d \in \mathbb{Z}^*_{\varphi(n)}$ mit $e \cdot d \equiv 1 \pmod{\varphi(n)}$

Verschlüsselung einer Nachricht $x \in \{0, 1, \ldots, n-1\}$: $x^e \bmod n$

Entschlüsselung einer Nachricht $y \in \{0, 1, \ldots, n-1\}$: $y^d \bmod n$

- Was bedeutet das alles?
- Wie finde ich zu einer Zahl $e \in \mathbb{Z}^*_m$ eine Zahl $d \in \mathbb{Z}^*_m$ mit $e \cdot d \equiv 1 \,(\mathrm{mod} m)$?
- Wie kann man effizient $x^e \bmod n$ bzw. $y^d \bmod n$ berechnen?
- Wieso kommt beim Entschlüsseln die ursprüngliche Nachricht raus?
  D.h. warum gilt $(x^e \bmod n)^d \bmod n = x$?
- Wie sieht es mit der Sicherheit von RSA aus?

Zahlentheorie

Definition: Es seien $a, b \in \mathbb{Z}$.

a teilt b, geschrieben $a \mid b$, falls eine Zahl $k \in \mathbb{Z}$ existiert mit $b = k \cdot a$.

a heisst dann Teiler von b, und b Vielfaches von a.

Beispiele: (1) $3 \mid 18$, da $18 = 6 \cdot 3$ und $6 \in \mathbb{Z}$.

(2) $-9 \mid 18$, da $18 = (-2) \cdot (-9)$ und $-2 \in \mathbb{Z}$.

(3) Die Teiler von 12 sind: 1,2,3,4,6,12,-1,-2,-3,-4,-6,-12.

Satz 1.1: (1) $\forall a \in \mathbb{Z} : 1 \mid a$ $\quad a = a \cdot 1$ und $a \in \mathbb{Z}$

(2) $\forall a, b \in \mathbb{Z} : a \mid b \Rightarrow (-a) \mid b$ $\quad b = k \cdot a$ mit $k \in \mathbb{Z} \Rightarrow b = (-k) \cdot (-a)$ und $-k \in \mathbb{Z}$

(3) $\forall a \in \mathbb{Z} : a \mid 0$ $\quad 0 = 0 \cdot a$ und $0 \in \mathbb{Z}$

(4) $\forall a, b, c \in \mathbb{Z} : (a \mid b) \wedge (b \mid c) \Rightarrow a \mid c$

$b = k_1 \cdot a \wedge c = k_2 \cdot b$ mit $k_1, k_2 \in \mathbb{Z} \Rightarrow c = (k_1 \cdot k_2) \cdot a$ und $k_1 \cdot k_2 \in \mathbb{Z}$

(5) $\forall a, b, c \in \mathbb{Z} : (a \mid b) \wedge (a \mid c) \Rightarrow a \mid (b + c)$

(6) $\forall a, b, c, d \in \mathbb{Z} : (a \mid b) \wedge (c \mid d) \Rightarrow (a \cdot c) \mid (b \cdot d)$

Zahlentheorie

Definition:

Eine Zahl $n \in \mathbb{N}$ heisst Primzahl, wenn sie genau zwei verschiedene Teiler in $\mathbb{N}$ hat.

Bemerkung: 1 ist **keine** Primzahl, da 1 nur einen Teiler in $\mathbb{N}$ hat, nämlich 1.

Die ersten Primzahlen lauten: 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43,...

Primzahlen sind gewissermassen die Bausteine aller Zahlen. Dies ist die Aussage des sogenannten Fundamentalsatzes der Arithmetik:

Fundamentalsatz der Arithmetik

Es sei $n \in \mathbb{N}$ mit $n \geq 2$. Dann existieren (bis auf die Reihenfolge) eindeutige Primzahlen $p_1, \ldots, p_k$ und natürliche Zahlen $e_1, \ldots, e_k \geq 1$ mit

$$n = p_1^{e_1} \cdot \ldots \cdot p_k^{e_k}.$$

Beispiele: (1) $99 = 3^2 \cdot 11$

(2) $360 = 2^3 \cdot 3^2 \cdot 5$

Wie berechnet man die Primfaktorzerlegung einer natürlichen Zahl?

Dies ist das berühmte Faktorisierungsproblem.

Das Faktorisierungsproblem – Bedeutung

Kryptographie

Blum-Blum-Shub Pseudorandom Bit Generator

Rabin Signaturschema

Blum-Goldwasser Kryptosystem

Rabin Kryptosystem

⋮

RSA Kryptosystem

⋮

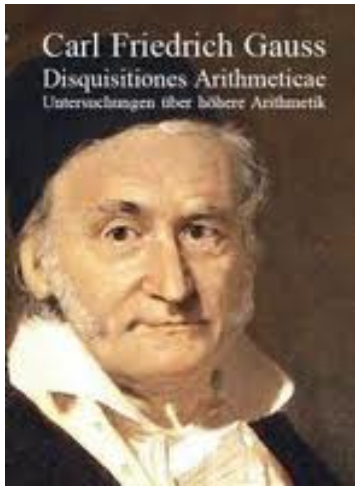
Sicher, genau dann, wenn Faktorisierung-problem schwer ist!

Unsicher, wenn das Faktorisierungproblem nicht schwer ist!

Später mehr dazu!

Das Faktorisierungsproblem – Bedeutung

Carl Friedrich Gauss
Untersuchungen über
höhere Arithmetik (1889)



Zerlegung zusammengesetzter Zahlen. 387

Zwei Methoden, zusammengesetzte Zahlen von primen zu unterscheiden und ihre Factoren zu ermitteln.

329.

Dass die Aufgabe, die Primzahlen von den zusammengesetzten zu unterscheiden und letztere in ihre Primfactoren zu zerlegen, zu den wichtigsten und nützlichsten der gesamten Arithmetik gehört und die Bemühungen und den Scharfsinn sowohl der alten wie auch der neueren Geometer in Anspruch genommen hat, ist so bekannt, dass es überflüssig wäre, hierüber viele Worte zu verlieren. Trotzdem muss man gestehen, dass alle bisher angegebenen Methoden entweder auf sehr specielle Fälle beschränkt oder so mühsam und weitläufig sind, dass sie schon für solche

vorkommenden Fällen jedenfalls ausreichen, so bietet sich doch dem erfahrenen Rechner nicht selten die Gelegenheit dar, aus der Zerlegung grosser Zahlen in Factoren grosse Vorteile zu ziehen, welche den mässigen Aufwand an Zeit reichlich wieder ausgleichen; ausserdem aber dürfte es die Würde

Zahlen in Factoren grosse Vorteile zu ziehen, welche den mässigen Aufwand an Zeit reichlich wieder ausgleichen; ausserdem aber dürfte es die Würde der Wissenschaft erheischen, alle Hülfsmittel zur Lösung jenes so eleganten und berühmten Problems fleissig zu vervollkommnen. Aus diesen Gründen zweifeln wir nicht, dass die beiden folgenden Methoden, deren Wirksamkeit und Kürze wir durch eine lange Erfahrung bestätigen können, den Lieb-

müdlichen Rechner unerträgliche Arbeit erforderten, behandelt worden.

Bevor man die folgenden Methoden anwendet, ist es immer vorteilhaft, die Division irgend einer gegebenen Zahl durch einige der kleinsten Primzahlen, z. B. durch 2, 3, 5, 7, ... bis zu 19 oder noch weiter hinaus, zu versuchen, nicht nur, damit es nicht reut, eine solche Zahl, falls sie Divisor ist, durch subtile und künstliche Methoden erhalten zu haben, die man viel leichter durch blosse Division hätte finden können*), sondern auch deshalb,

*) Um so mehr, weil sich, allgemein zu reden, unter sechs Zahlen kaum eine findet, die nicht durch eine der Zahlen 2, 3, 5, ..., 19 teilbar wäre.

25*

Das Faktorisierungsproblem – Übersicht

300 v. Chr. | ~ 1643 | 1889 | 1974 | 1975 | 1982 | 1985 | 1988 | 1990

Euklid
Probedivision

Teste der Reihe nach alle Primzahlen, ob sie Teiler sind. Wenn man einen Teiler gefunden hat, teile dadurch und fange wieder von vorne an.

Fermat

Gauss

Pollard
$p-1$-Methode

Shanks
Square Form Factorization

Pollard
ρ-Methode

Morrison & Brillhart
Continued Fraction Factorization

Pomerance
Quadratisches Sieb

Williams
$p+1$-Methode

Lenstra
Elliptic Curve Method

Pollard
spezielles Zahlkörpersieb

zahlreiche Varianten & Verbesserungen

Buhler, Lenstra & Pomerance
allgemeines Zahlkörpersieb

Mehr dazu im Kryptographie-Modul. Hier nur so viel:

Bisher ist kein effizienter Algorithmus bekannt, der zu einer Zahl die Primfaktorzerlegung berechnet. Die erwartete Laufzeit bei in der Kryptographie üblichen Zahlen liegt bei vielen Millionen Jahren, selbst wenn alle Computer der Welt zusammenarbeiten würden.

d.h. Polynomzeit

Zahlentheorie

Wir haben gesehen, dass ein Teil des öffenlichen Schlüssels des RSA-Kryptosystems aus dem Produkt zweier Primzahlen besteht.

Dies wirft zwei Fragen auf:

(1) Gibt es denn genug Primzahlen?

(2) Wie finden wir effizient (grosse) Primzahlen?

Wie viele Primzahlen gibt es?

Satz 1.2: Es gibt unendlich viele Primzahlen!

Beweis: Wir nehmen an, dass es nur endlich viele Primzahlen $p_1 < p_2 < \ldots < p_n$ gibt.

Nun sei $p := p_1 \cdot p_2 \cdot \ldots \cdot p_n + 1$.

Da $p > p_n$ kann p keine Primzahl sein. ← p_n ist ja gemäss unserer Annahme die grösste Primzahl.

Es gibt also eine Primzahl p^*, die p teilt. ← Dies tut jede Zahl aus der Primfaktorzerlegung von p.

Es gilt nun $p^* \neq p_i$ für alle $i = 1, 2, \ldots, n$.

Ansonsten würde gelten $p^* \mid p$ **und** $p^* \mid (p_1 \cdot p_2 \cdot \ldots \cdot p_n)$, also auch

$$p^* \mid \underbrace{(p - p_1 \cdot p_2 \cdot \ldots \cdot p_n)}_{=1}, \text{ was nicht sein kann. (Da } p^* > 1.)$$

Wie findet man effizient Primzahlen?

In der Praxis wählt man zufällig eine (grosse) Zahl, und testet, ob diese Zahl eine Primzahl ist. Wenn nicht, macht man weiter!

Damit das funktioniert, müssen zwei Dinge erfüllt sein:

(1) Eine Primzahl darf nicht die absolute Ausnahme sein. (Sonst ist die Wahrscheinlichkeit sehr klein, eine zu treffen.)

Primzahlsatz:

Es bezeichne $\pi(n)$ die Anzahl aller Primzahlen $\leq n$. Dann gilt für $n \geq 55$:

$$\frac{n}{\ln n + 2} < \pi(n) < \frac{n}{\ln n - 4}.$$

Beispiel: Zwischen 10 Millionen und 100 Millionen gibt es

$$\pi(100\text{ Mio.}) - \pi(10\text{ Mio.}) > \frac{100\text{ Mio.}}{\ln 100\text{ Mio.}+2} - \frac{10\text{ Mio.}}{\ln 10\text{ Mio.}-4} > 4\text{ Mio. Primzahlen.}$$

Wenn man also zufällig eine der 45 Mio. ungeraden Zahlen zwischen 10 Mio. und 100 Mio. wählt, dann ist im Schnitt mehr als jede zwölfte Zahl eine Primzahl.

(2) Man muss effizient überprüfen können, ob eine Zahl eine Primzahl ist.

(2) Man muss effizient überprüfen können, ob eine Zahl eine Primzahl ist.

Es gibt einen effizienten Algorithmus entscheidet, ob eine gegebene Zahl eine Primzahl ist.

d.h. Polynomzeit

Interessanterweise kann man also einer Zahl (effizient) ansehen, ob sie eine Primzahl ist, aber es ist (bisher) i.A. nicht effizient möglich, einen Teiler zu bestimmen.

In der Praxis verwendet man sogenannte probabilistische Primzahltests, etwa den Miller-Rabin-Primzahltest.

Er gibt bei Eingabe $n \in \mathbb{N}$ entweder "n ist keine Primzahl" oder "n ist wahrscheinlich eine Primzahl" aus.

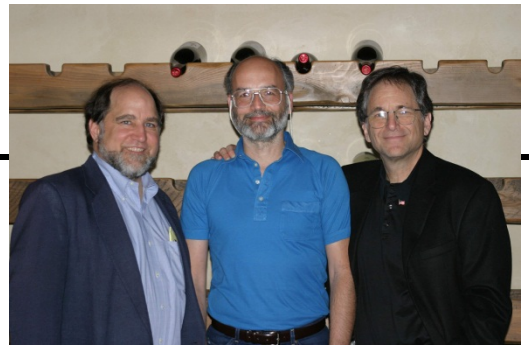
Hierbei ist die Wahrscheinlichkeit dafür, dass er ausgibt, dass n eine Primzahl ist, obwohl n keine Primzahl ist, kleiner als $\frac{1}{4}$.

Man kann den Test allerdings mehrfach anweden, so dass man dadurch die Fehlerwahrscheinlichkeit reduzieren kann.

Wenn man den Test etwa 10 mal nacheinander ausführt, liegt die Fehlerwahrscheinlichkeit nur noch bei $\frac{1}{4^{10}} \approx 0,0000009$.

$\rightarrow$ mehr dazu (evtl.) im Kryptographie-Modul

Das Kryptosystem: RSA



Ron Rivest Adi Shamir Len Adleman

Mit ElGamal das wichtigste asymmetrische Kryptosystem!

Privater Schlüssel: (n, d)

Öffentlicher Schlüssel: (n, e)

$n = p \cdot q$ mit Primzahlen $p \neq q$,

$e, d \in \mathbb{Z}^*_{\varphi(n)}$ mit $e \cdot d \equiv 1 \pmod{\varphi(n)}$

Verschlüsselung einer Nachricht $x \in \{0, 1, \ldots, n-1\}$: $x^e \bmod n$

Entschlüsselung einer Nachricht $y \in \{0, 1, \ldots, n-1\}$: $y^d \bmod n$

- Was bedeutet das alles?
- Wie finde ich zu einer Zahl $e \in \mathbb{Z}^*_m$ eine Zahl $d \in \mathbb{Z}^*_m$ mit $e \cdot d \equiv 1 \,(\mathrm{mod} m)$?
- Wie kann man effizient $x^e \bmod n$ bzw. $y^d \bmod n$ berechnen?
- Wieso kommt beim Entschlüsseln die ursprüngliche Nachricht raus?
  D.h. warum gilt $(x^e \bmod n)^d \bmod n = x$?
- Wie sieht es mit der Sicherheit von RSA aus?

ggT

Definition: Es seien $a, b \in \mathbb{Z}$, nicht beide gleich 0.

Dann heisst $ggT(a,b) := \max\{x \in \mathbb{Z} \mid (x \mid a) \wedge (x \mid b)\}$ grösster gemeinsamer Teiler von a und b.

a und b heissen teilerfremd, falls $ggT(a,b) = 1$.

Beispiele: (1) $ggT(6,9) =$ □ (2) $ggT(-6,9) =$ □ (3) $ggT(-6,-9) =$ □

Satz 1.3: Für $a, b \in \mathbb{N}$ gilt $a \mid b \Leftrightarrow ggT(a,b) = a$.

Es gelte $a \mid b$.

Die grösste Zahl, die a teit, ist a selber, und da a auch b teilt, ist a die grösste Zahl, die a und b teilt.

Es gelte $ggT(a,b) = a$.

Dann ist a die grösste Zahl, die a und b teilt. Insbesondere teilt a also b.

In obigem Satz ist die Voraussetzung $a, b \in \mathbb{N}$ wesentlich.

Es gilt z.B. $-3 \mid 6$, aber $ggT(-3,6) = 3$.

Satz 1.4: Für alle $a \in \mathbb{N}$ gilt $ggT(a,0) =$ □

0 wird von allen Zahlen geteilt, und die grösste Zahl, die a teilt ist a selber.

ggT

Wie berechnet man den ggT zweier Zahlen?

Möglichkeit 1: Über die Primfaktorzerlegung

Was ist der ggT von $2079 = 3^3 \cdot 7 \cdot 11$ und $5733 = 3^2 \cdot 7^2 \cdot 13$?

3^2 und 7 kommen in beiden Zahlen vor, also $ggT(2079, 5733) = 3^2 \cdot 7 = 63$.

Der ggT zweier Zahlen ist das Produkt aller Primfaktoren, die in beiden Zahlen vorkommen, potenziert mit der niedrigeren der beiden Potenzen.

Für grosse Zahlen ungeeignet, da die Primfaktorzerlegung benötigt wird!

Möglichkeit 2: Mit dem (erweiterten) euklidischen Algorithmus

$\rightarrow$ später

bis 1.9

Eulersche φ-Funktion

Definition: Es sei $n \geq 2$. Dann ist

$$\mathbb{Z}_n^* = \{a \in \{0, 1, \ldots, n-1\} \mid ggT(a, n) = 1\}$$

und die Eulersche φ-Funktion $\varphi(n)$ definiert durch $\varphi(n) = \left|\mathbb{Z}_n^*\right|$.

Beispiel: $\varphi(12) = \left|\{a \in \{0, 1, \ldots, 11\} \mid ggT(a, 12) = 1\}\right| = \left|\boxed{}\right| \boxed{}$

Satz 1.5: Es sei p Primzahl. Dann gilt: $\varphi(p) = \boxed{}$

Satz 1.6:

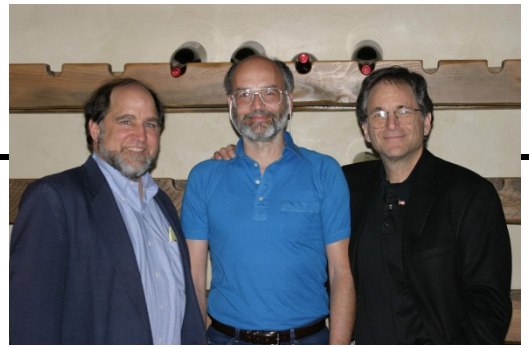
Primfaktorzerlegung von n

Es sei $n \geq 2$ und $n = p_1^{e_1} \cdot \ldots \cdot p_k^{e_k}$, für paarweise verschiedene Primzahlen $p_1, \ldots, p_k$ und $e_1, \ldots, e_k \in \mathbb{N}$.

Dann gilt: $\varphi(n) = (p_1 - 1) \cdot p_1^{e_1 - 1} \cdot \ldots \cdot (p_k - 1) p_k^{e_k - 1}$.

Beispiel: $\varphi(12) = \boxed{}$

Das Kryptosystem: RSA



Ron Rivest Adi Shamir Len Adleman

Mit ElGamal das wichtigste asymmetrische Kryptosystem!

Privater Schlüssel: (n, d)

Öffentlicher Schlüssel: (n, e)

$n = p \cdot q$ mit Primzahlen $p \neq q$,

$e, d \in \mathbb{Z}^*_{\varphi(n)}$ mit $e \cdot d \equiv 1 \pmod{\varphi(n)}$

Verschlüsselung einer Nachricht $x \in \{0, 1, \ldots, n-1\}$: $x^e \bmod n$

Entschlüsselung einer Nachricht $y \in \{0, 1, \ldots, n-1\}$: $y^d \bmod n$

- Was bedeutet das alles?
- Wie finde ich zu einer Zahl $e \in \mathbb{Z}^*_m$ eine Zahl $d \in \mathbb{Z}^*_m$ mit $e \cdot d \equiv 1 \,(\mathrm{mod} m)$?
- Wie kann man effizient $x^e \bmod n$ bzw. $y^d \bmod n$ berechnen?
- Wieso kommt beim Entschlüsseln die ursprüngliche Nachricht raus?
  D.h. warum gilt $(x^e \bmod n)^d \bmod n = x$?
- Wie sieht es mit der Sicherheit von RSA aus?

Modulo

Satz 1.7 (und Definition):

Es seien $n \in \mathbb{N}$ und $a \in \mathbb{Z}$.

Es existiert genau ein $q \in \mathbb{Z}$ und ein $r \in \{0, \ldots, n-1\}$ mit $a = n \cdot q + r$.

a div $n := q$

a mod $n := r$ Rest von a modulo n

Beispiele:

$10 \bmod 4 = 2$ $\quad -1 \bmod 4 = 3 \quad (-1 = -1 \cdot 4 + 3)$

$10 \text{ div } 4 = 2$ $\quad -1 \text{ div } 4 = -1$

Bemerkung:

Für negative Zahlen gibt es abweichende Definitionen und Implementierungen von div und mod.

Modulo

Rechenregeln Modulo:

Es seien $n \in \mathbb{N}$ und $a, b \in \mathbb{Z}$.

(1) $a \bmod n = b \bmod n \Leftrightarrow n \mid (a - b)$

(2) $a \bmod n = (a \bmod n) \bmod n$

(3) $(a + b) \bmod n = ((a \bmod n) + b) \bmod n$
also auch $(a + b) \bmod n = (a + (b \bmod n)) \bmod n$
und $(a + b) \bmod n = ((a \bmod n) + (b \bmod n)) \bmod n$.

(4) $(a \cdot b) \bmod n = ((a \bmod n) \cdot b) \bmod n$
also auch $(a \cdot b) \bmod n = (a \cdot (b \bmod n)) \bmod n$
und $(a \cdot b) \bmod n = ((a \bmod n) \cdot (b \bmod n)) \bmod n$.

Wenn in Summen und Produkten am Ende ein mod n steht, kann man überall innerhalb des Ausdrucks beliebig mod n hinzufügen oder weglassen.

Beispiel:

$(6 \cdot 17^{23} + 18 \cdot 12345 + 15) \bmod 6 =$

Modulo

Rechenregeln Modulo (Beweise):

Es seien $n \in \mathbb{N}$ und $a, b \in \mathbb{Z}$.

(1) $a \bmod n = b \bmod n \Leftrightarrow n \mid (a-b)$

Es ist $a = q_1 \cdot n + r_1$ und $b = q_2 \cdot n + r_2$ mit $q_1, q_2, r_1, r_2 \in \mathbb{Z}$ und $0 \leq r_1, r_2 < n$.

Es gelte $a \bmod n = b \bmod n$.

Dann ist also $r_1 = r_2$, also $a - b = q_1 n + r_1 - (q_2 n + r_2) = (q_1 - q_2) \cdot n$, womit, wegen $q_1 - q_2 \in \mathbb{Z}$ gilt, dass $n \mid (a-b)$.

Es gelte $n \mid (a-b)$.

Dann ist $a - b = k \cdot n$ mit $k \in \mathbb{Z}$, also $k \cdot n = (q_1 - q_2) \cdot n + (r_1 - r_2)$.

Es folgt $r_1 - r_2 = (k - q_1 + q_2) \cdot n$, also $n \mid (r_1 - r_2)$.

Es folgt $r_1 - r_2 = 0$, also $a \bmod n = b \bmod n$, da $-n < r_1 - r_2 < n$ und 0 die einzige Zahl in dem Bereich ist, die von n geteilt wird.

(2) $a \bmod n = (a \bmod n) \bmod n$

Gemäss (1) reicht es zu zeigen, dass $n \mid (a - a \bmod n)$. Es ist $a = q \cdot n + r$ mit $q, r \in \mathbb{Z}$ und $0 \leq r < n$.

Also: $a - a \bmod n = a - r = q \cdot n$, was $n \mid (a - a \bmod n)$ bedeutet.

(3) $(a+b) \bmod n = ((a \bmod n) + b) \bmod n$

$a + b - (a \bmod n + b) = a - a \bmod n$ wird von n geteilt.

also auch $(a+b) \bmod n = (a + (b \bmod n)) \bmod n$

und $(a+b) \bmod n = ((a \bmod n) + (b \bmod n)) \bmod n$.

(4) $(a \cdot b) \bmod n = ((a \bmod n) \cdot b) \bmod n$

also auch $(a \cdot b) \bmod n = (a \cdot (b \bmod n)) \bmod n$

und $(a \cdot b) \bmod n = ((a \bmod n) \cdot (b \bmod n)) \bmod n$.

Modulo

Definition: Es seien $n \in \mathbb{N}$ und $a, b \in \mathbb{Z}$.

Wir schreiben $a \equiv b \pmod{n}$ oder $a \equiv_n b$ falls $a \bmod n = b \bmod n$ und sagen:

a ist kongruent zu b modulo n.

Beispiele: (1) $7 \equiv_3 4$ (2) $7 \not\equiv_4 4$ (3) $7 \equiv_5 17$ (3) $-1 \equiv_3 2$

Satz 1.8: Es sei $n \in \mathbb{N}$. Dann ist $\equiv_n$ eine Äquivalenzrelation auf $\mathbb{Z}$.

D.h.: (1) $\equiv_n$ ist reflexiv, d.h. $\forall a \in \mathbb{Z} : a \equiv_n a$

(2) $\equiv_n$ ist symmetrisch, d.h. $\forall a, b \in \mathbb{Z} : a \equiv_n b \Rightarrow b \equiv_n a$

(3) $\equiv_n$ ist transitiv, d.h. $\forall a, b, c \in \mathbb{Z} : (a \equiv_n b) \wedge (b \equiv_n c) \Rightarrow (a \equiv_n c)$.

Eine Äquivalenzrelation induziert bekannlich eine Partitionierung (oder Klasseneinteilung) der zu Grunde liegende Menge (hier $\mathbb{Z}$) und umgekehrt.

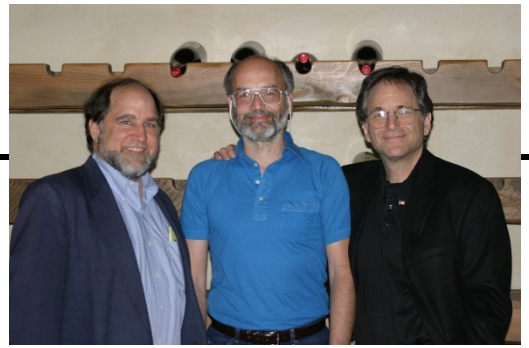
Die Äquivalenzklasse $[a]_n$ enthält alle Elemente, die zu a kongruent modulo n sind.

Beispiele: (1) $[7]_3 = \{a \in \mathbb{Z} \mid a \equiv_3 7\} =$ ☐

(2) $[1]_3 = [7]_3$

(3) $[1]_4 = \{\ldots -11, -7, -3, 1, 5, 9, 13, \ldots\}$

Das Kryptosystem: RSA



Ron Rivest Adi Shamir Len Adleman

Mit ElGamal das wichtigste asymmetrische Kryptosystem!

Privater Schlüssel: (n, d)

Öffentlicher Schlüssel: (n, e)

$n = p \cdot q$ mit Primzahlen $p \neq q$.

$e, d \in \mathbb{Z}^*_{\varphi(n)}$ mit $e \cdot d \equiv 1 \pmod{\varphi(n)}$

Verschlüsselung einer Nachricht $x \in \{0, 1, \ldots, n-1\}$: $x^e \bmod n$

Entschlüsselung einer Nachricht $y \in \{0, 1, \ldots, n-1\}$: $y^d \bmod n$

- Was bedeutet das alles?
- Wie finde ich zu einer Zahl $e \in \mathbb{Z}^*_m$ eine Zahl $d \in \mathbb{Z}^*_m$ mit $e \cdot d \equiv 1 \,(\mathrm{mod} m)$?
- Wie kann man effizient $x^e \bmod n$ bzw. $y^d \bmod n$ berechnen?
- Wieso kommt beim Entschlüsseln die ursprüngliche Nachricht raus?
  D.h. warum gilt $(x^e \bmod n)^d \bmod n = x$?
- Wie sieht es mit der Sicherheit von RSA aus?

Erweiterter Euklidscher Algorithmus

`Euklid`(a, b) *Vorbedingung*: $a, b \in \mathbb{Z}, a \geq b \geq 0$

1. Initialisierte Schleife. $a' = a, b' = b$

$$x_0 = 1, y_0 = 0, x_1 = 0, y_1 = 1$$

2. Schleife: `solange` $b' \neq 0$

Invariante: $\mathrm{ggT}(a, b) = \mathrm{ggT}(a', b')$
$a' = x_0 a + y_0 b, b' = x_1 a + y_1 b$
$a' \geq b' \geq 0.$

$$q = a' \text{ div } b', r = a' \mod b'$$

$$a' = b', b' = r$$

$$(x_0, y_0, x_1, y_1) = (x_1, y_1, x_0 - q x_1, y_0 - q y_1)$$

Nachbedingung: $a' = \mathrm{ggT}(a, b) = \boxed{x_0} a + \boxed{y_0} b$

Bezout-Koeffizienten

Satz 1.9:

Der erweiterte Euklidsche Algorithmus ist bzgl. Vor- und Nachbedingung korrekt und terminiert nach $\mathcal{O}(\log a)$ Schleifendurchläufen, wenn die Vorbedingung erfüllt ist. Alle auftretenden Zahlen sind betragsmässig durch a beschränkt.

Erweiterter Euklidscher Algorithmus

Folgerung: Der ggT von zwei Zahlen $a, b \in \mathbb{Z}$ ist effizient berechenbar.

Beispiel: ggT(18, 7):

$q = a' \text{ div } b', r = a' \mod b'$

$a' = b', b' = r$

$(x_0, y_0, x_1, y_1) = (x_1, y_1, x_0 - qx_1, y_0 - qy_1)$

a'	b'	x_0	y_0	x_1	y_1	q	r
18	7	1	0	0	1	2	4
7	4	0	1	1	-2	1	3
4	3	1	-2	-1	3	1	1
3	1	-1	3	2	-5	3	0
1	0	2	-5				

$\Rightarrow$ ggT(18, 7) = 1 = 2 · 18 + (−5) · 7

bis 1.19

Modulo

Wie finde ich zu einer Zahl $e \in \mathbb{Z}_m^*$ eine Zahl $d \in \mathbb{Z}_m^*$ mit $e \cdot d \equiv_m 1$?

Existiert so eine Zahl überhaupt?

Satz 1.10: Es seien $m \in \mathbb{N}$ mit $m \geq 2$ und $e \in \mathbb{Z}_m = \{0, 1, 2, \ldots, m-1\}$.

Dann existiert eine Zahl $d \in \mathbb{Z}_m$ mit $e \cdot d \equiv_m 1$ genau dann, wenn $ggT(m, e) = 1$, also $e \in \mathbb{Z}_m^*$.

Zudem ist dann auch $d \in \mathbb{Z}_m^*$.

Die Zahl d kann effizient mit Hilfe des erweiterten euklidischen Algorithmus berechnet werden.

Aus $1 = ggT(m, e) = x_0 \cdot m + y_0 \cdot e$ erhalten wir nämlich direkt $y_0 \cdot e \equiv_m 1$.

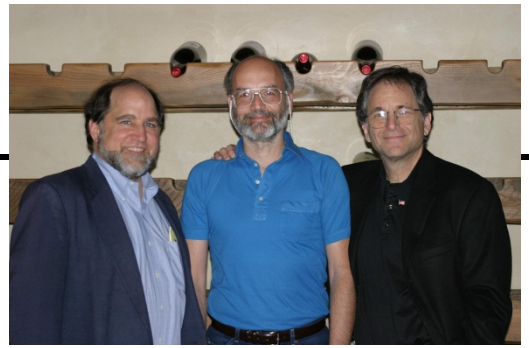
Aus $1 = x_0 \cdot m + y_0 \cdot e$ folgt $1 \bmod m = (x_0 \cdot m + y_0 \cdot e) \bmod m$,
also $1 = (x_0 \cdot (m \bmod m) + y_0 \cdot e) \bmod m$, also $1 = (y_0 \cdot e) \bmod m$.

Aufgabe: Berechnen Sie zu $e = 9$ eine Zahl $d \in \mathbb{Z}_{31}$ mit $e \cdot d \equiv_{31} 1$.

Lösung: $d = 7$

bis 1.26

Das Kryptosystem: RSA



Ron Rivest Adi Shamir Len Adleman

Mit ElGamal das wichtigste asymmetrische Kryptosystem!

Privater Schlüssel: (n, d)

Öffentlicher Schlüssel: (n, e)

$n = p \cdot q$ mit Primzahlen $p \neq q$.

$e, d \in \mathbb{Z}^*_{\varphi(n)}$ mit $e \cdot d \equiv 1 \pmod{\varphi(n)}$

Verschlüsselung einer Nachricht $x \in \{0, 1, \ldots, n-1\}$: $x^e \bmod n$

Entschlüsselung einer Nachricht $y \in \{0, 1, \ldots, n-1\}$: $y^d \bmod n$

- Was bedeutet das alles?
- Wie finde ich zu einer Zahl $e \in \mathbb{Z}^*_m$ eine Zahl $d \in \mathbb{Z}^*_m$ mit $e \cdot d \equiv 1 \,(\mathrm{mod} m)$?
- Wie kann man effizient $x^e \bmod n$ bzw. $y^d \bmod n$ berechnen?
- Wieso kommt beim Entschlüsseln die ursprüngliche Nachricht raus?
  D.h. warum gilt $(x^e \bmod n)^d \bmod n = x$?
- Wie sieht es mit der Sicherheit von RSA aus?

Modulare Exponentiation

Gegeben: $e, m \in \mathbb{N},\ x \in \mathbb{Z}$

Gesucht: Effizientes Verfahren um $x^e \mod m$ zu berechnen

Schlechte Idee:

Multipliziere e-mal x mit sich selber, und rechne anschliessend modulo m

Die entstehenden Zahlen werden zwischenzeitlich viel zu gross!

Besser:

Verwende $(a \cdot b) \bmod n = (a \bmod n \cdot b) \bmod n$ und rechne "zwischendurch" schon modulo m.

Es sind aber immer noch $e - 1$ Multiplikationen nötig.

Noch besser:

Verwende die sogenannte schnelle Exponentiation! (auch square & multiply genannt)

Schnelle Exponentiation

Idee: Nutze $x^{2^k} = \left(\left(x^2 \right)^{2 \cdots} \right)^2$ aus!

k-mal

Arbeitsersparnis: (Anzahl Multiplikationen)

	herkömmlich	iteriertes Quadrieren
x^{16}		

Was ist, wenn der Exponent keine Zweierpotenz ist?

Zerlege den Exponenten in eine Summe von Zweierpotenzen!

Binär-darstellung

Beispiel: $13 = 2^3 + 2^2 + 2^0 \Rightarrow x^{13} = x^{2^3} \cdot x^{2^2} \cdot x^{2^0}$ (Beachte: $13 = (1101)_2$)

"Wir gehen von hinten nach vorne über die Binärdarstellung, quadrieren jedesmal und multiplizieren, falls das Bit 1 ist."

Schnelle Exponentiation

"Wir gehen von hinten nach vorne über die Binärdarstellung, quadrieren jedesmal und multiplizieren, falls das Bit 1 ist."

Algorithmus: Schnelle Exponentiation

Eingabe: $x \in \mathbb{Z}$, $m \in \mathbb{N}$ und $e \in \mathbb{N}$ mit Binärdarstellung $e = (b(0)b(1)\ldots b(l))_2$

1. Initialisierung
   $i = l;\ h = 1;\ k = x;$
2. iteriertes Quadrieren
   Solange $i \geq 0$
   Falls $b(i) = 1$
   $h = h \cdot k \mod m$
   $k = k^2 \mod m$
   $i = i - 1$
3. Ergebnis ausgeben
   Gib h **aus**

Ausgabe: $x^e \mod m$

$\rightarrow \mathcal{O}(\log_2 e)$-Multiplikationen

Beispiel: Gesucht: $7^{13} \bmod 11$

$(13)_2 = 1101$

i	h	k
3	1	7
2	7	$49 \bmod 11 = 5$
1	7	3
0	10	9
-1	2	

Exponentiation

Heute übliche Schlüsselgrösse: 2048 Bit

Beispiel: Nehmen wir an, e wäre nur 141 Bit groß, also $e \approx 2^{141}$.

1) Multipliziere x $e-1$ mal mit sich selber: $2^{141}-1 > 2^{140}$ Multiplikationen nötig!

Computer mit 1 Mrd. TeraHertz Taktfrquenz

"ein ZettaHertz"

$\Rightarrow 10^{21} < 2^{70}$ Schritte pro Sekunde

Annahme: ein Schritt pro Taktzyklus

Laufzeit: Mehr als 2^{140-70} Sekunden, also 2^{70} Sekunden!

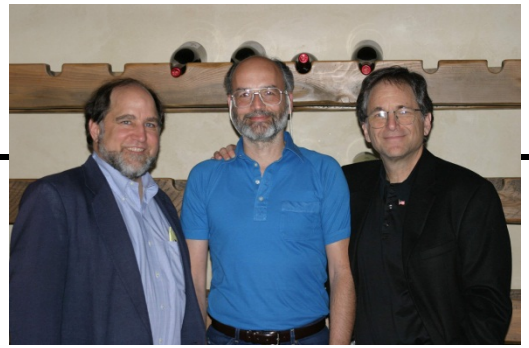
Zeit seit Urknall: 13,7 Mrd. Jahre $< 2^{60}$ Sekunden!

Also mehr als 1000 mal die Zeit seit Urknall wird benötigt!!

2) schnelle Exponentiation: Ungefähr $2 \cdot 141$ Multiplikationen nötig!!

Ginge zur Not sogar per Hand in wenigen Stunden!

Das Kryptosystem: RSA



Ron Rivest Adi Shamir Len Adleman

Mit ElGamal das wichtigste asymmetrische Kryptosystem!

Privater Schlüssel: (n, d)

Öffentlicher Schlüssel: (n, e)

$n = p \cdot q$ mit Primzahlen $p \neq q$.

$e, d \in \mathbb{Z}^*_{\varphi(n)}$ mit $e \cdot d \equiv 1 \pmod{\varphi(n)}$

Verschlüsselung einer Nachricht $x \in \{0, 1, \ldots, n-1\}$: $x^e \bmod n$

Entschlüsselung einer Nachricht $y \in \{0, 1, \ldots, n-1\}$: $y^d \bmod n$

- Was bedeutet das alles?
- Wie finde ich zu einer Zahl $e \in \mathbb{Z}^*_m$ eine Zahl $d \in \mathbb{Z}^*_m$ mit $e \cdot d \equiv 1 \,(\mathrm{mod} m)$?
- Wie kann man effizient $x^e \bmod n$ bzw. $y^d \bmod n$ berechnen?
- Wieso kommt beim Entschlüsseln die ursprüngliche Nachricht raus? später!
  D.h. warum gilt $(x^e \bmod n)^d \bmod n = x$?
  (Wir stellen noch Hilfsmittel dazu bereit)
- Wie sieht es mit der Sicherheit von RSA aus?

Serie 1 komplett

Sicherheitsbetrachtungen

1) Wieso ist es schwer, aus (n, e) den priv. Schl. (n, d) zu berechnen?

Privater Schlüssel: (n, d)

Öffentlicher Schlüssel: (n, e)

$n = p \cdot q$ mit Primzahlen $p \neq q$,

$e, d \in \mathbb{Z}^*_{\varphi(n)}$ mit $e \cdot d \equiv 1 \pmod{\varphi(n)}$

Wenn $\varphi(n)$ bekannt wäre, dann kann man aus e leicht d bestimmen.

erw. eukl. Alg.

$\varphi(n)$ kann leicht aus p, q berechnet werden, aber p, q sind unbekannt!

p, q sind die Primfaktoren von n und n ist bekannt.

Wenn man also die Primfaktoren von n bestimmen kann, dann kann man aus dem öffentlichen Schlüssel den privaten berechnen!

Wir haben aber bereits gesehen, dass das Faktorisierungsproblem (zumindest zurzeit) schwer ist!

Sicherheitsbetrachtungen

2) Wieso ist es schwer, x aus $x^e \mod n$ ohne d zu berechnen?

Wir haben gesehen, dass d schwer zu berechnen ist.

Vielleicht braucht man d aber gar nicht, um die Entschlüsselungsfunktion zu berechnen?

Es ist nicht bekannt, ob das Problem genauso schwer ist wie Faktorisieren.

Man nimmt auch an, dass das Problem sehr schwer ist. Das ist die RSA-Annahme.