Euklidische Ringe

Sei R ein Integritätsring. Man nennt R einen euklidischen Ring, wenn es eine Gradfunktion $d : R\backslash\{0\} \to \mathbb{Z}^{\geq 0}$ mit der folgenden Eigenschaft gibt: Zu $a, b \in R$ und $b \neq 0$ gibt es $s, r \in R$ mit $a = sb + r$ und $r = 0$ oder $d(r) < d(b)$ (Division mit Rest).

Notation wieder $s = a$ div b und $r = a \bmod b$.

Thm: Für jedes Ideal I von R gibt es ein $b \in R$ mit $I = Rb$ (das heißt I ist ein Hauptideal).

Sind $a_1, a_2 \in R$, so gibt es daher ein $c \in R$ mit $Rc = Ra_1 + Ra_2$. Es gibt also $\lambda_1, \lambda_2 \in R$ mit $c = \lambda_1 a_1 + \lambda_2 a_2$ und $c \mid a_1$, $c \mid a_2$.

Die Elemente λ_i können mit dem euklidischen Algorithmus ausgerechnet werden.

Primelemente und Faktorisierung

Sei $c \in R\backslash(\{0\} \cup R^\times)$. Folgt aus $c \mid (ab)$ bereits $c \mid a$ oder $c \mid b$ für alle $a, b \in R$, so heißt c Primelement von R. Folgt aus $c = ab$ bereits $a \in R^\times$ oder $b \in R^\times$, so heißt c irreduzibel.

Thm: Sei R euklidisch.

i) Die Menge der Primelemente ist gleich der Menge der irreduziblen Elemente.

ii) Für jedes $a \in R\backslash\{0\}$ gibt es $u \in R^\times$, Primelemente p_i und $e_i \in \mathbb{Z}^{\geq 1}$ mit

$$a = u \prod_{i=1}^{n} p_i^{e_i}.$$

Die e_i und p_i (bis auf Elemente in $R^\times$) sind eindeutig bestimmt.

Wir erhalten Funktionen $v_{p_i} : R\backslash\{0\} \to \mathbb{Z}$ mit $v_{p_i}(a) = e_i$.

Größter gemeinsamer Teiler

Sei $c = \lambda_1 a_1 + \lambda_2 a_2$ die Ausgabe des euklidischen Algorithmus.

- Für $d \mid a_1$ und $d \mid a_2$ gilt wegen $c = \lambda_1 a_1 + \lambda_2 a_2$ auch $d \mid c$. Daher ist c ein größter gemeinsamer Teiler von a_1 und a_2, geschrieben $c = \gcd\{a_1, a_2\}$.
- Andererseits gilt $\gcd\{a_1, a_2\} = \prod_i p_i^{\min\{v_{p_i}(a_1), v_{p_i}(a_2)\}}$, wenn die p_i die in $a_1 a_2$ vorkommenden Primelemente bezeichnen.
- Es gilt $R\gcd\{a_1, a_2\} = Ra_1 + Ra_2$.

Zwei Elemente $a, b \in R$ heißen teilerfremd, wenn $\gcd\{a, b\} = 1$ oder äquivalenterweise $R = Ra + Rb$.

Kleinstes gemeinsames Vielfaches

Sei $c = a_1 a_2 / \gcd\{a_1, a_2\} = \prod_i p_i^{\max\{v_{p_i}(a_1), v_{p_i}(a_2)\}}$.

- Es gilt $a_1 \mid c$ und $a_2 \mid c$ und für jedes $d \in R$ mit $a_1 \mid d$ und $a_2 \mid d$ folgt $c \mid d$. Damit ist c ein kleinstes gemeinsames Vielfaches von a_1 und a_2, geschrieben $c = \mathrm{lcm}\{a_1, a_2\}$.
- Es gilt $R\,\mathrm{lcm}\{a_1, a_2\} = Ra_1 \cap Ra_2$.
- Sind a_1 und a_2 teilerfremd, so folgt $Ra_1 \cap Ra_2 = Ra_1 a_2$.

Beispiele

In $\mathbb{Z}$ gilt $\mathbb{Z}^\times = \{-1, 1\}$. Die Primelemente sind $\pm 2, \pm, 3, \pm 5, \ldots$.

In $k[x]$ gilt $k[x]^\times = k^\times$. Die Primelemente vom Grad eins sind $u(x - x_0)$ für $u \in k^\times$ und $x_0 \in k$.

In $\mathbb{Z}$ ist ± 6 ein ggT und ± 210 ein kgV von 42 und 30.
In $k[x]$ gilt $v_{x+1}(x^2 - 1) = 1$ und $v_x(x^2 - 1) = 0$.

Es gilt $3\mathbb{Z} + 2\mathbb{Z} = \mathbb{Z}$ und $2\mathbb{Z} \cap 3\mathbb{Z} = 2\mathbb{Z} \cdot 3\mathbb{Z} = 6\mathbb{Z}$.

Chinesischer Restsatz

Seien $a_i \in R$ paarweise teilerfremd, $I_i = Ra_i$, $a = \prod_i a_i$ und $I = Ra$.
Wir betrachten den Homomorphismus $f : R \to \prod_{i=1}^n R/I_i$,
$x \mapsto (x + I_i)_{1 \le i \le n}$.

Es gilt $\ker(f) = \cap_{i=1}^n I_i = \prod_i I_i = I$, so daß $g : R/I \to \prod_{i=1}^n R/I_i$, $g(x + I) \mapsto f(x)$ injektiv ist.

Thm: Der Monomorphismus g ist ein Isomorphismus,

$$R/I \cong \prod_{i=1}^n R/I_i.$$

Bew: Es bleibt die Surjektivität zu zeigen. Setze $b_i = \prod_{j \neq i} a_j$. Die b_i haben keinen gemeinsamen Primteiler und daraus folgt $R = \sum_i Rb_i$. Es gibt also $\lambda_i \in R$ mit $1 = \sum_i \lambda_i b_i$. Sei $(c_i + I_i)_i \in \prod_{i=1}^n R/I_i$ mit $c_i \in R$. Setze $c = \sum_i c_i \lambda_i b_i$. Wegen $a_j \mid b_i$ für alle $j \neq i$ ergibt sich $1 = \lambda_i b_i \bmod I_i$ und $c = c_i \bmod I_i$. Folglich $g(c) = (c_i + I_i)_i$ und g ist surjekiv. $\square$

Chinesischer Restsatz

Allgemeiner $\lambda_i = b_i^{-1} \bmod a_i$ möglich.

Formel von Garner: $a = a_1 a_2$ mit a_1, a_2 prim und teilerfremd. Sei $c = ((c_1 - c_2)(a_2^{-1} \bmod a_1) \bmod a_1)a_2 + c_2$. Dann $c = c_1 \bmod a_1$ und $c = c_2 \bmod a_2$.

Formel von Garner kann induktiv auf beliebig viele a_i verallgemeinert werden.

Englische Abkürzung: CRT (Chinese Remainder Theorem).

Beispiele

Es gilt $\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/3\mathbb{Z} \times \mathbb{Z}/5\mathbb{Z} \cong \mathbb{Z}/30\mathbb{Z}$. Wegen $1 = -3 \cdot 5 + 2 \cdot 5 + 2 \cdot 3$ entspricht das Element $(1, 2, 3)$ dem Wert
$1 \cdot -3 \cdot 5 + 2 \cdot 2 \cdot 5 + 3 \cdot 2 \cdot 3 = -7 = 23 \bmod 30$.
Check: $23 = 1 \bmod 2$, $23 = 2 \bmod 3$ und $23 = 3 \bmod 5$.

Es gilt $k[x]/(x-1)k[x] \times k[x]/(x+1)k[x] \cong k[x]/(x^2-1)k[x]$.
Elemente der linken Seite werden durch Paare $(c_1, c_2) \in k \times k$ repräsentiert.
Wegen $1 = (x+1)/2 - (x-1)/2$ entspricht das Element (c_1, c_2) dem Wert $c = c_1(x+1)/2 - c_2(x-1)/2 = (c_1 + c_2 + (c_1 - c_2)x)/2 \bmod x^2 - 1$.
Check: Für $x = 1$ ergibt sich $c(1) = c_1$ und für $x = -1$ ergibt sich $c(-1) = c_2$.

Regel: $g(x) \bmod (x - x_0) = g(x_0)$.

Lagrange Interpolation

Ist chinesischer Restsatz für lineare Polynome.

Seien $x_i \in k$ für $1 \leq i \leq n$ paarweise verschieden, und $y_i \in k$. Wollen $f \in k[x]$ mit $\deg(f) < n$ und $f(x_i) = y_i$ finden.

Ansatz: $b_i(x) = \prod_{j \neq i}(x - x_j)/(x_i - x_j)$. Dann gilt
$1 = b_i(x_i) = b_i(x) \bmod (x - x_i)$ und $0 = b_i(x_j) = b_i(x) \bmod (x - x_j)$.
Folglich $1 = \sum_{i=1}^{n} b_i(x)$ zunächst modulo $\prod_i (x - x_i)$ und dann exakt aus Gradgründen. Desweiteren ist $f(x) = \sum_i y_i b_i(x)$ ist das gesuchte Polynom.

Shamir's Secret Sharing

Aufgabe:

n Teilnehmer sollen sich ein Geheimnis teilen (z.B. soll ein Masterschlüssel aufgeteilt werden).
Genau dann, wenn $\geq t$ Teilnehmer zusammenarbeiten, sollen sie das Geheimnis rekonstruieren können.

Aufteilen des Geheimnisses:

Sei $y_0 \in k$ das Geheimnis und $x_0 \in k$ öffentlich bekannt.

1. Wähle ein zufälliges Polynom $f \in k[x]$ vom Grad $t - 1$ mit $f(x_0) = y_0$.
2. Wähle paarweise verschiedene $x_1, \ldots, x_n \in k$ mit $x_i \neq x_0$ und berechne $y_i = f(x_i)$.
3. Gebe (x_i, y_i) für $1 \leq i \leq n$ an den i-ten Teilnehmer.

Shamir's Secret Sharing

Rekonstruktion des Geheimnisses:

$\geq t$ Teilnehmer können das Polynom f eindeutig aus ihren Werten (x_i, y_i) mit Lagrange Interpolation rekonstruieren, und damit den Wert $y_0 = f(x_0)$ berechnen.

Bei $< t$ Teilnehmer ist das Polynom nicht eindeutig bestimmt, $f(x_0)$ kann theoretisch jeden beliebigen Wert aus k annehmen.

Langzahlarithmetik

Müssen mit großen ganzen Zahlen rechnen.

Seien $f, g \in \mathbb{Z}^{\geq 1}$ mit $b = \log(f) \geq \log(g)$. Aufwand in Bitoperationen:

- $f + g$, $f - g$: $O(\log(f) + \log(g)) = O(b)$
- fg: $O(\log(f)\log(g)) = O(b^2)$.
- $f = sg + r$: $O(\log(s)\log(g)) = O(b^2)$.
- $\gcd\{f, g\}$: $O(b^2)$.

Analoges gilt für $f, g \in k[x]$ mit $\log$ ersetzt durch $\deg$ und Aufwand in Operationen in k.

Es gibt asymptotisch bessere Algorithmen. Die Multiplikation kann z.B. in $O(b^{1.58})$ oder auch $O(b \log(b)^2)$ ausgeführt werden. Spielt in der Kryptographie keine so große Rolle, da die Zahlen nicht groß genug sind.

Langzahlarithmetik

Müssen mit großen ganzen Zahlen modulo n rechnen.

Seien $f, g \in \mathbb{Z}/n\mathbb{Z}$ mit $b = \log(n)$. Aufwand in Bitoperationen:

- $f+g$, $f-g$: $O(b)$
- fg, f/g: $O(b^2)$.
- f^c: $O(\log(c)b^2)$.

Analoges gilt für $f, g \in k[x]/hk[x]$ mit $b = \deg(h)$ und Aufwand in Operationen in k.

Es gibt Bibliotheken für die Langzahlarithmetik in $\mathbb{Z}$ und $\mathbb{Z}/n\mathbb{Z}$ bzw. für $k[x]$ und $k[x]/hk[x]$ (z.B. GMP, NTL, ...).

Langzahlarithmetik mit CRT

Rechnen in $\mathbb{Z}/n\mathbb{Z}$ für $n = pq$ mit p, q prim, teilerfremd und $\log(p) \approx \log(q)$:

- $\mathbb{Z}/n\mathbb{Z} \cong \mathbb{Z}/p\mathbb{Z} \times \mathbb{Z}/q\mathbb{Z}$.
- Statt in $\mathbb{Z}/n\mathbb{Z}$ in $\mathbb{Z}/p\mathbb{Z} \times \mathbb{Z}/q\mathbb{Z}$ rechnen.
- Zwischen $\mathbb{Z}/n\mathbb{Z}$ und $\mathbb{Z}/p\mathbb{Z} \times \mathbb{Z}/q\mathbb{Z}$ mit $\bmod$ und Garner hin- und herschalten.

Multiplikation in $\mathbb{Z}/p\mathbb{Z}$ bzw. $\mathbb{Z}/q\mathbb{Z}$ viermal so schnell wie in $\mathbb{Z}/n\mathbb{Z}$.
Daher doppelt so schnell in $\mathbb{Z}/p\mathbb{Z} \times \mathbb{Z}/q\mathbb{Z}$ wie in $\mathbb{Z}/n\mathbb{Z}$.

Exponentiationen für zufälligen Exponenten noch besser, $b = \log_2(n)$:

- In $\mathbb{Z}/n\mathbb{Z}$: $\approx 3b/2$ b-Bit Multiplikationen.
- In $\mathbb{Z}/p\mathbb{Z}$: $\approx 3b/4$ $b/2$-Bit Multiplikationen.
- In $\mathbb{Z}/p\mathbb{Z} \times \mathbb{Z}/q\mathbb{Z}$: $\approx 3b/2$ $b/2$-Bit Multiplikationen.

Daher 3-4 mal schneller.

Euler Phi Funktion

Sei $R \in \{\mathbb{Z}, k[x]\}$ für $k = \mathbb{F}_q$ und $n \in R\backslash\{0\}$. Dann $\phi(n) = \#(R/nR)^\times$.

Eigenschaften:

- $\phi(n) = 0$ genau dann, wenn $n \in R^\times$.
- $\phi(n^e) = \#(R/nR)^\times \cdot \#(R/nR)^{e-1}$.
- Speziell $\phi(n^e) = (n-1)n^{e-1}$ für primes $n \in \mathbb{Z}$ und $e \in \mathbb{Z}^{\geq 0}$.
- Speziell $\phi(n^e) = (q^d - 1)q^{d(e-1)}$ für primes $n \in k[x]$ mit $\deg(n) = d$ und $e \in \mathbb{Z}^{\geq 0}$.
- $\phi(n_1 n_2) = \phi(n_1)\phi(n_2)$ für teilerfremde Nichteinheiten $n_1, n_2 \in R$.

RSA Kryptosystem

Wurde 1977 von Rivest, Shamir und Adleman erfunden.

- Genaue Beschreibung im PKCS #1.
- RSA US-Patent 2000 ausgelaufen (aber nicht für spezielle Techniken - „multiprime RSA“).
- De-facto Standard für asymmetrische Kryptosysteme.
- 1973 im geheimen vom CESG (UK) erfunden.

Schlüsselerzeugung:

Seien p, q zwei verschiedene, ungerade Primzahlen und $n = pq$.
Sei $1 < e < \phi(n)$ teilerfremd zu $\phi(n) = (p-1)(q-1)$ und $1 < d < \phi(n)$ mit $ed = 1 \bmod \phi(n)$.

Der öffentliche Schlüssel ist (n, e).
Der geheimen Schlüssel ist $(p, q, \phi(n), d)$.

RSA Kryptosystem

Die RSA Funktion wird durch $f : \mathbb{Z}/n\mathbb{Z} \to \mathbb{Z}/n\mathbb{Z}$, $x \mapsto x^e \bmod n$ definiert.

Thm: Die zu f inverse Funktion ist durch $g : \mathbb{Z}/n\mathbb{Z} \to \mathbb{Z}/n\mathbb{Z}$, $y \mapsto y^d \bmod n$ gegeben.

Bew: Wir müssen zeigen: $x^{ed} = x$ für $x \in \mathbb{Z}/n\mathbb{Z}$. Es gilt $ed = 1 + m(p-1)$ für ein $m \in \mathbb{Z}$ und $x^{p-1} = 1$ für $x \in (\mathbb{Z}/p\mathbb{Z})^\times$, daher auch $x^{ed} = xx^{m(p-1)} = x$ für alle $x \in \mathbb{Z}/p\mathbb{Z}$. Analog für q. Also gilt $x^{eq} = x$ für $x \in \mathbb{Z}/p\mathbb{Z} \times \mathbb{Z}/q\mathbb{Z}$ und wegen $\mathbb{Z}/n\mathbb{Z} \cong \mathbb{Z}/p \times \mathbb{Z}/q\mathbb{Z}$ auch für $x \in \mathbb{Z}/n\mathbb{Z}$. $\square$

Verschlüsselung: Nachricht als $m \in \mathbb{Z}/n\mathbb{Z}$ darstellen, dann Chiffretext $c = f(m)$ mit öffentlichem Schlüssel e berechnen.

Entschlüsselung: $m = g(c)$ mit dem geheimen Schlüssel d berechnen.

RSA Parameter

Die Primzahlen p, q sollen eine ungefähr gleiche Länge von mindestens 512 Bit haben, also n eine Bitlänge von 1024.

Man erwartet, daß n mit 2048 Bit bis 2022 und n mit 4096 Bit bis 2050 sicher sind. Ein n mit 6800 Bit entspricht der Sicherheit von AES-128.

Übliche Möglichkeiten für e sind $e \in \{3, 5, 17, 65537\}$.

Der Wert von d hat dann im algemeinen eine Bitlänge ähnlich wie n. Verschlüsseln ist daher viel schneller als entschlüsseln.

Primzahlerzeugung

Thm (Primzahlsatz): Sei $\pi(x) = \#\{p \mid p \text{ Primzahl}, 1 \leq p \leq x\}$.
Für $x > 17$ gilt $x/\log(x) < \pi(x) < 1.25506x/\log(x)$.
Für $x \to \infty$ gilt $\pi(x)/(x/\log(x)) \to 1$.

Strategie:

- Zufällige ganze Zahlen x der gewünschten Bitlänge b erzeugen.
- Testen, ob x prim. Erfolgswahrscheinlichkeit $1/(b\log(2))$.

Primzahltests:

- Miller-Rabin: Eine Antwort „ist prim“ ist mit Wahrscheinlichkeit $\leq (1/4)^r$ falsch, bei r Iterationen. Kann man unter 2^{-128} drücken. Eine Antwort „ist nicht prim“ ist immer richtig.
- Einige andere probabilistische Tests ...
- Agrawal-Kayal-Saxena (2002): Deterministischer, polynomieller Primzahltest („primes in P“).

RSA Sicherheit

Thm: Aus einem Teil des geheimen Schlüssels $(p, q, \phi(n), d)$ und dem öffentlichen Schlüssel (n, e) kann man die anderen Teile des geheimen Schlüssels effizient ausrechnen.

- Gegeben p oder q ist das klar.
- Gegeben $\phi(n)$ haben wir zwei Gleichungen $pq = n$, $(p-1)(q-1) = \phi(n)$. Man kann leicht nach p und q auflösen.
- Gegeben d ist $ed - 1$ bei kleinem e ein kleines Vielfaches von $\phi(n)$. Durch Ausprobieren kann man $\phi(n)$ herausbekommen.
- Auch bei beliebigem e kann man aus d die Werte $p, q, \phi(n)$ berechnen (nächste Folie).

Ein Algorithmus zum Berechnen von $\phi(n)$ oder d ist also polynomiell äquivalent zu einem Algorithmus zum Faktorisieren von n.