4 Faktoriseringsalgoritmer

I dette kapitel ønsker vi at se på faktoriseringsalgoritmer. Det viser sig nemlig, at en af de anvendelser som elliptiske kurver besidder, er indenfor faktoriseringen af heltal. Faktoriseringsproblemet, altså hvordan man bestemmer en faktor for et tal n er yderst relevant, da alle heltal kan faktoriseres:

Sætning 5 (Aritmetikkens fundamentalsætning)**.** *Et heltal $n > 1$ kan faktoriseres entydigt som et produkt af primtal, så hvis*

$$p_1 \cdot p_2 \cdot \ldots \cdot p_k = q_1 \cdot q_2 \cdot \ldots \cdot q,$$

hvor p_i og q_j er primtal for $1 \leq i \leq k$ og $1 \leq j \leq l$ er $k = l$ og $p_i = q_i$ for alle $i = 1, 2, \ldots, k$ (efter eventuelle ombytninger). Desuden er faktorerne $p_1^{n_1}, p_2^{n_2}, \ldots, p_k^{n_k}$ entydigt bestemte.

For et bevis af sætningen se f.eks. [1]. Det vigtige at bemærke er, at beviset ikke er konstruktivt og dermed ikke giver os en måde, hvorpå vi kan finde disse faktorer. Men hvordan kan vi så finde disse faktorer, som vi nu ved findes? Hvis vi har et sammensat tal n, som vi ønsker at faktorisere kunne vi angribe problemet med en naiv tilgang. Vi antager for nemhedens skyld at $n = pq$, hvilket gør det klart at $\min\{p, q\} \leq \sqrt{n}$. Vi kan altså finde en faktor ved at undersøge om først $2 \mid n$, dernæst om $3 \mid n$ osv. indtil at vi finder en faktor, hvilket vil ske senest når vi når til $\sqrt{n}$. Denne løsning er fin for tilstrækkeligt små tal, men det bliver hurtigt uoverkommeligt for store tal (eksempler på hvor lang tid det tager?).

Sikkerheden i moderne kryptosystemer hviler på det faktum, at det tager lang tid at faktorisere et heltal. Derfor er det interessant at undersøge om man gøre det hurtigere end den med den naive tilgang. Vi skal se på to af sådanne algoritmer, nemlig Pollards $p-1$ algoritme og Lenstras algoritme, som benytter elliptiske kurver til at finde en faktor. Idéen med begge algoritmer er at finde et $x \in \mathbb{Z}$ sådan, at $x \not\equiv 0 \pmod{n}$ og $x \equiv 0 \pmod{p}$ for en eller anden primfaktor p i n. Da har vi nemlig, at $\gcd(x, n)$ er en ikke-triviel divisor i n.

4.1 Pollards $p - 1$ algoritme

Da Lenstras algoritme er stærkt inspireret af Pollards $p - 1$ algoritme og devlist kan ses som værende en analog til denne, vælger vi at behandle den

først. Pollards $p-1$ algoritme blev først præsenteret i [2] i 1970'erne af J. M. Pollard. Algoritmen er en måde hvorpå vi kan finde primfaktorer p for et heltal n når $p-1$ kun har *små* primfaktorer.

Vi ser nu på, hvordan Pollards $p-1$ algoritme prøver at finde en faktor. Lad n være et sammensat tal og lad p være en primfaktor i n. For $a \in \mathbb{Z}$ sådan at $\gcd(a,n) = 1$ har vi fra Fermats lille sætning, at

$$a^{p-1} \equiv 1 \pmod{p}.$$

Antag nu, at $p-1 \mid \text{LCM}[1,2,\ldots,K] = k$ for et alle andet $K \in \mathbb{Z}^+$, hvor LCM er mindste fælles multiplum. Da har vi, at

$$a^k = a^{m(p-1)} = (a^{p-1})^m \equiv 1 \pmod{p}.$$

Lader vi $x = a^k - 1$ har vi nu, at $p \mid d = \gcd(x,n)$. Hvis nu $x \not\equiv 0 \pmod{n}$ er d en ikke-triviel divisor i n. Bemærk, at vi i de udregninger ikke har haft brug for at kende p. I praksis vil vi dog udregne $\gcd(x \pmod{n}, n)$ da vi ellers kan risikere at x bliver meget stort og besværligt at regne med, men det ændrer ikke på resultatet. Med den nu fundne faktor har vi en faktorisering $n = d \cdot \frac{n}{d}$ og vi kan gentage processen på disse to faktorer, hvis de ikke allerede er primtal.

Det hele hviler altså på, at n skal have en primfaktor p sådan, at

$$p-1 \mid \text{LCM}[1,2,\ldots,K].$$

Dette vil der være stor sandsynlighed for, hvis $p-1$ har mange små primfaktorer. Vi er da klar til, at præsentere algoritmen, som er inspireret af gennemgangen af algoritmen i [3]:

Algoritme 1 (Pollards $p-1$ algoritme)**.** Lad $n \geq 2$ være et sammensat tal, som er tallet vi ønsker at finde en faktor for.

1. Vælg $k \in \mathbb{Z}^+$ sådan, at k er et produkt af mange små primtal opløftet i små potenser. Eksempelvis kan k vælges til at være

   $$k = \text{LCM}[1,2,\ldots,K],$$

   for et $K \in \mathbb{Z}^+$ og hvor LCM er det mindste fælles multiplum.

2. Vælg et heltal a sådan, at $1 < a < n$.

3. Udregn $\gcd(a,n)$. Hvis $\gcd(a,n) > 1$ har vi fundet en ikke-triviel faktor for n og vi er færdige. Ellers fortsæt til næste trin.

4. Udregn $D = \gcd(a^k - 1 \pmod{n}, n)$. Hvis $1 < D < n$ er D en ikke-triviel faktor for n og vi er færdige. Hvis $D = 1$ gå da tilbage til trin 1 og vælg et større k. Hvis $D = n$ gå da til trin 2 og vælg et nyt a.

Denne version af algoritmen vil på et tidspunkt stoppe, da vi på et tidspunkt vil have at $K = \frac{1}{2}(p-1)$ i trin 1 for et eller andet $p \mid n$, hvilket betyder at $p - 1 \mid k$. Hvis der ikke bliver fundet en faktor før dette sker er algoritmen dog yderst ineffektiv og man vil i praksis kun teste til en fastsat grænse for K.

Følgende er et eksempel på anvendelsen af Pollards algoritme, hvor det går godt, altså hvor $p - 1$ har små primfaktorer:

Eksempel 4. Vi vil forsøge at faktorisere

$$n = 30042491.$$

Vi ser at $2^{n-1} = 2^{30042490} \equiv 25171326 \pmod{30042491}$, så N er ikke et primtal. Vi vælger som beskrevet i algoritmen

$$a = 2 \quad \text{og} \quad k = \text{LCM}[1, 2, \ldots, 7] = 420.$$

Da $420 = 2^2 + 2^5 + 2^7 + 2^8$ skal vi udregne 2^{2^i} for $0 \leq i \leq 8$. Dette resulterer i følgende tabel:

i	$2^{2^i} \pmod{n}$		
1	4	5	28933574
2	16	6	27713768
3	256	7	10802810
4	65536	8	16714289
5	28933574		

Denne tabel gør det forholdsvist let for os, at bestemme

$$\begin{aligned} 2^{420} &= 2^{2^2+2^5+2^7+2^8} \\ &\equiv 16 \cdot 28933574 \cdot 10802810 \cdot 16714289 \pmod{30042491} \\ &\equiv 27976515 \pmod{30042491}. \end{aligned}$$

Ved anvendelse af den euklidiske algoritme finder vi dernæst, at

$$\gcd(2^{420} - 1 \pmod{n}, n) = \gcd(27976514, 30042491) = 1.$$

Her fejler testen altså og vi er nået frem til, at N ikke har nogle primtalsfaktorer p sådan, at $p - 1$ deler 420. Algoritmen foreskriver da, at vi skal vælge et nyt k. Vi lader

$$k = \text{LCM}[1, 2, \ldots, 11] = 27720.$$

Da $27720 = 2^{14}+2^{13}+2^{11}+2^{10}+2^6+2^3$ skal vi udvide tabellen til at indeholde værdierne for 2^{2^i} for $0 \leq i \leq 14$:

i	$2^{2^i} \pmod n$		
9	19694714	12	26818902
10	3779241	13	8658967
11	11677316	14	3783587

Vi fortsætter på samme måde, som vi gjorde før og bestemmer

$$\begin{aligned} 2^{27720} &= 2^{2^3+2^{2^6}+2^{2^{10}}+2^{2^{11}}+2^{2^{13}}+2^{2^{14}}} \\ &= 256 \cdot 27713768 \cdot 3779241 \cdot 11677316 \cdot 8658967 \cdot 3783587 \\ &= 16458222 \pmod{30042491}. \end{aligned}$$

Vi finder dernæst, at

$$\gcd(2^{27720} - 1 \pmod n, n) = \gcd(16458221, 30042491) = 9241,$$

hvilket betyder at vi har fundet en ikke-triviel faktor for n. Mere præcist har vi fundet faktoriseringen

$$30042491 = 3251 \cdot 9241.$$