Faktorizace pomocí $\varphi(n)$

21. přednáška z kryptografie

Obsah

1. **Faktorizace pomocí $\varphi(n)$**
2. **Algoritmus na poznání perfektní mocniny**

Faktorizace pomocí $\varphi(n)$

V případě Fermatova a Millerova-Rabinova testu prvočíselnosti jsme si všimli, že volba některých svědků složenosti čísla n umožní najít faktor čísla n.

- Volba $a \in \mathbb{Z}_n^+ \setminus \mathbb{Z}_n^*$ poskytne faktor $d = \gcd(a, n) > 1$.
- Volba $a \in K_n \setminus L_n$ vygeneruje netriviální druhou odmocninu z 1 (tj. $c \neq \pm 1$, kde $c^2 = 1$ v $\mathbb{Z}_n$), která poskytne faktory $d_{1,2} = \gcd(c \pm 1, n) > 1$.

Použijeme techniku z důkazu odhadu počtů falešných svědků prvočíselnosti u Millerova-Rabinova testu a ukážeme, že problém faktorizace n je ekvivalentní problému znalosti $\varphi(n)$.

Faktorizace pomocí $\varphi(n)$

Tvrzení

Problém faktorizace n je ekvivalentní problému znalosti $\varphi(n)$, aneb ze znalosti jednoho lze spočítat druhé v polynomiálním čase.

- Z faktorizace $n = \prod_{i=1}^{r} p_i^{e_i}$ máme $\varphi(n) = \prod_{i=1}^{r} p_i^{e_i - 1}(p_i - 1)$.
- Pro $n = pq$ ze znalosti $\varphi(n)$ spočteme p, q jako řešení kvadratické rovnice $x^2 - (n + 1 - \varphi(n))x + n = 0$. Protože $\varphi(n) = (p-1)(q-1) = n - (p+q) + 1$, známe součet a součin obou řešení.
- Pro libovolné n navrhneme polynomiální algoritmus, který ze znalosti násobku exponentu grupy $\mathbb{Z}_n^*$ spočte faktorizaci n.

Faktorizace pomocí $\varphi(n)$

Exponent grupy $\mathbb{Z}_n^*$

Exponent grupy $\mathbb{Z}_n^*$ je nejmenší $m > 0$ takové, že $a^m = 1$ pro všechna $a \in \mathbb{Z}_n^*$. Značí se $\lambda(n)$, tzv. Carmichaelova funkce.

- $\lambda(\prod_{i=1}^{r} p_i^{e_i}) = \mathrm{lcm}(\lambda(p_1^{e_1}), \ldots, \lambda(p_r^{e_r}))$.
- $\lambda(p^e) = \varphi(p^e) = p^{e-1}(p-1)$ pro prvočísla $p > 2$
- $\lambda(2^e) = \frac{\varphi(2^e)}{2} = 2^{e-2}$ pro $e \geq 3$, $\lambda(4) = 2$, $\lambda(2) = 1$.

Důsledek

- Pro každé n platí $\lambda(n) \mid \varphi(n)$, aneb $\varphi(n)$ je násobek exponentu grupy $\mathbb{Z}_n^*$.
- Pro každé $n > 2$ je $\lambda(n)$ sudé.
- Pokud $d \mid n$, pak $\lambda(d) \mid \lambda(n)$.

Faktorizace pomocí $\varphi(n)$

Algoritmus na nalezení faktoru čísla n pomocí násobku $\lambda(n)$

Vstup: $n > 1$ liché, kde $n \neq p^e$ pro žádné prvočíslo p,
m takové, že $\lambda(n) \mid m$, $m = t\,2^h$ pro t liché;
Výstup: d, kde $d \mid n$, $1 < d < n$, nebo hláška "neúspěch"

- $a \xleftarrow{q} \mathbb{Z}_n^+$
- $d \leftarrow gcd(a, n)$
- if $d > 1$ then output d and halt endif
- $b \leftarrow a^t$ v $\mathbb{Z}_n$ (nyní je $a \in \mathbb{Z}_n^*$, tedy $a^m = 1$ v $\mathbb{Z}_n$)
- for $j \leftarrow 0$ to $h-1$ do
  - $d \leftarrow \gcd(b-1, n)$
  - it $1 < d < n$ then output d and halt endif
  - $b \leftarrow b^2$ v $\mathbb{Z}_n$ enddo
- output "neúspěch"

Faktorizace pomocí $\varphi(n)$

Tvrzení

Pravděpodobnost, že algoritmus nalezne faktor čísla n je aspoň $\frac{1}{2}$.

Volba $a \in \mathbb{Z}_n^+ \setminus \mathbb{Z}_n^*$ vede k faktorizaci v 1.části, volba $a \in \mathbb{Z}_n^*$ vede vždy k vygenerování nějaké druhé odmocniny z jedné v 2.části. Algoritmus může ohlásit neúspěch jen při volbě $a \in L$, kde $L = \{a \in \mathbb{Z}_n^*, \text{když } a^{t\,2^j} = 1, \text{pak } a^{t\,2^{j-1}} = \pm 1, \text{pro } 1 \leq j \leq h\}$.
Podobně jako u Millerova-Rabinova testu lze dokázat, že pro $n = \prod_{i=1}^{r} p_i^{e_i}$, kde $r \geq 2$, p_i lichá prvočísla, platí:

$$|L| \leq \frac{2}{2^r}|Ker\rho_{t2^g}| \leq \frac{1}{2}|\mathbb{Z}_n^*|,$$

kde $\rho_{t2^g} : x \mapsto x^{t2^g}$, $g = \min\{h, h_1, \ldots, h_r\}$, $m = t2^h$, $\varphi(p_i^{e_i}) = t_i 2^{h_i}$ a t, t_i jsou lichá.

Faktorizace pomocí $\varphi(n)$

Časová náročnost

- Pokud je $m \in O(n)$ (což pro $\varphi(n)$ platí), pak algoritmus potřebuje čas $O(\mathrm{len}(n)^3)$. Očekávaný počet opakování, než nastane úspěch, je dva.
- Rozložíme-li $n = d_1 d_2$, pak $\lambda(d_i) \mid \lambda(n) \mid m$ a algoritmus můžeme použít rekurzivně. Rekurzivních volání algoritmu bude nejvýše $O(\mathrm{len}(n))$.
- Ověřování prvočíselnosti či perfektní mocniny trvá zhruba $O(\mathrm{len}(n)^3)$ (viz dále).
- Ze znalosti násobku $\lambda(n)$ získáme úplnou faktorizaci čísla n v čase zhruba $O(\mathrm{len}(n)^4)$.

Faktorizace pomocí $\varphi(n)$

Časová náročnost

Náš algoritmus, který nalezne netriviální faktor čísla n ze znalosti čísla m, kde $\lambda(n) \mid m$, funguje jen pro lichá $n \neq p^e$ pro libovolné prvočíslo p. To ale postačí, neboť:

- Sudé $n = 2^i\tilde{n}$, kde $\tilde{n}$ liché najdeme v čase $O(\mathrm{len}(n))$. Dále faktorizujeme $\tilde{n}$ naším algoritmem, neboť $\lambda(\tilde{n}) \mid m$.
- Perfektní mocninu $n = \tilde{n}^e$ poznáme v čase $O(\mathrm{len}(n)^3 \,\mathrm{len}(\mathrm{len}(n)))$ a faktorizujeme $\tilde{n}$, neboť $\lambda(\tilde{n}) \mid m$.
- Prvočíslo $n = p$ poznáme Millerovým-Rabinovým testem $MR(\cdot, k)$ v čase $O(k\,\mathrm{len}(n)^3)$ a už ho nefaktorizujeme.

Algoritmus na poznání perfektní mocniny

Výpočet celočíselné druhé odmocniny

Vstup: $n \in \mathbb{N}$
Výstup: $m = \lfloor\sqrt{n}\rfloor$
Poznámka: Je-li $2^{l-1} \leq n < 2^l$, pak $2^{\frac{l-1}{2}} \leq m < 2^{\frac{l}{2}}$.
Budeme druhou odmocninu z n počítat po bitech:

- $k \leftarrow \lfloor\frac{\mathrm{len}(n)-1}{2}\rfloor$
- $m \leftarrow 0$
- for $i \leftarrow k$ down to 0 do
  - if $(m + 2^i)^2 \leq n$ then $m \leftarrow m + 2^i$ endif enddo
- output m

Časová náročnost je $O(\frac{\mathrm{len}(n)}{2}\,\mathrm{len}(n)^2) = O(\mathrm{len}(n)^3)$,
nebo jen $O(\mathrm{len}(n)^2)$, protože umocnění na druhou je posun bitů.

Algoritmus na poznání perfektní mocniny

Výpočet celočíselné e−té odmocniny

Vstup: $n \in \mathbb{N}$
Výstup: $m = \lfloor\sqrt[e]{n}\rfloor$
Poznámka: Je-li $2^{l-1} \leq n < 2^l$, pak $2^{\frac{l-1}{e}} \leq m < 2^{\frac{l}{e}}$.

- $k \leftarrow \lfloor\frac{\mathrm{len}(n)-1}{e}\rfloor$
- $m \leftarrow 0$
- for $i \leftarrow k$ down to 0 do
  - if $(m + 2^i)^e \leq n$ then $m \leftarrow m + 2^i$ endif enddo
- output m

Časová náročnost je $O(\frac{1}{e}\,\mathrm{len}(n)^3)$.

Algoritmus na poznání perfektní mocniny

Algoritmus na poznání perfektní mocniny

Vstup: $n \in \mathbb{N}$
Výstup: Odpověď na otázku, zda je $n = m^e$ pro nějaké $m, e \in \mathbb{N}$.
Poznámka: $m \geq 2$, $2 \leq e \leq \mathrm{len}(n) + 1$

- for $e \leftarrow 2$ to $\mathrm{len}(n) + 1$ do
  - $m \leftarrow \lfloor\sqrt[e]{n}\rfloor$
  - if $m^e = n$ then output m, e and return *true* endif enddo
- return *false*

Časová náročnost je $O(\sum_{e=2}^{\mathrm{len}(n)+1} \frac{1}{e}\,\mathrm{len}(n)^3)$,
nahradíme-li sumu integrálem, získáme $O(\mathrm{len}(n)^3\,\mathrm{len}(\mathrm{len}(n)))$.

Faktorizace pomocí $\varphi(n)$

Literatura

- Shoup: A Computational Introduction to Number Theory and Algebra. Kapitola 10.
  http://shoup.net/ntb/