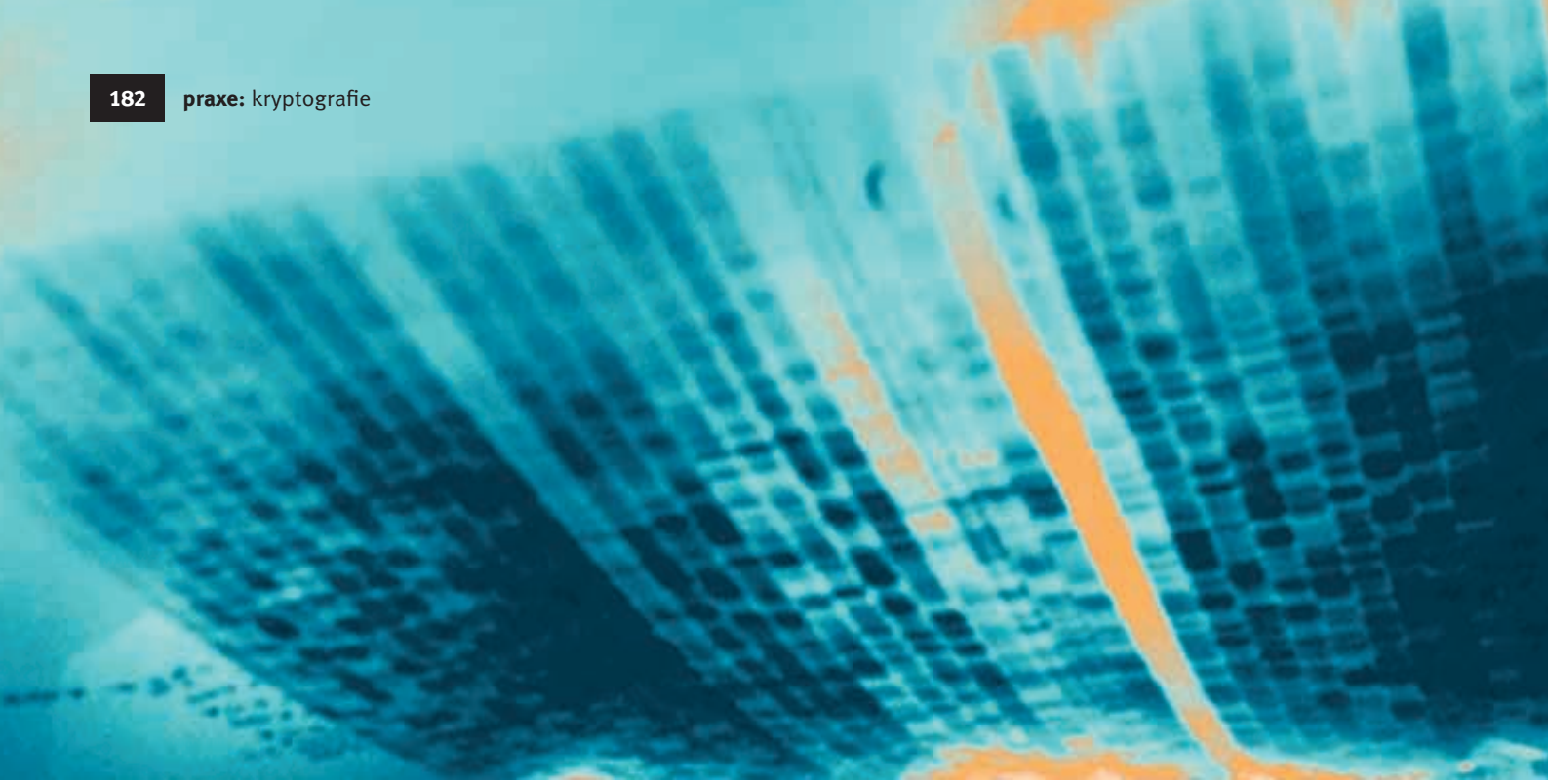

FAKTORIZACE (2)

Dvě čísla za 200 000 dolarů

Současný rekord ve faktorizaci čísla typu n = p*q je 155 dekadických číslic (512 bitů). V tomto povídání o problémech faktorizace vás seznámíme s metodou, které za to vděčíme. A odpovíme také na otázku, jak velké číslo bychom byli schopni faktorizovat, kdybychom měli všechny znalosti současné vědy a veškerou výpočetní techniku na Zemi.

NEJÚSPĚŠNĚJŠÍ JE GNFS

Nejlepší současnou metodou faktorizace je tzv. **metoda prosévání** neboli *General Number Field Sieve* (GNFS). Její popis je velmi složitý, proto je vynechávána i v kvalitních kryptografických příručkách – dílem kvůli rozsahu potřebných pojmů, dílem kvůli složitosti a teoretickým nárokům na čtenáře. Pokusíme se ale vysvětlit alespoň základní ideu tak, jak je popsána v monografii [LL].

GNFS se skládá ze čtyř fází. Hlavním cílem je opět nalézt čísla y, x tak, že $y^2 \equiv x^2 \pmod n$, viz minulý díl. Změnou oproti jiným metodám je kombinované využití čísel a polynomů. Označme tedy Z celá čísla a Z[α] okruh polynomů do stupně d -1 včetně, které mají celočíselné koeficienty, tj. jedná se o polynomy $a_{d-1}\alpha^{d-1} + a_{d-2}\alpha^{d-2} + \ldots + a_0$.

1. FÁZE – SKLOUBENÍ ČÍSEL S POLYNOMY

V první fázi se nejprve vybere *ireducibilní polynom* f(α) stupně d › 1. Jeho pomocí vytvoříme těleso Z[α]/f(α), což je množina uvedených polynomů, s nimiž se pracuje modulo f(α); f(α) je tedy v tomto tělese nulový prvek. Každý polynom si zde můžeme reprezentovat vektorem jeho koeficientů a sčítání polynomů pak provádíme jako sčítání odpovídajících si koeficientů (sčítání koeficientů je běžné sčítání celých čísel). Násobení polynomů zde provádíme jako běžné násobení polynomů, ale výsledek na závěr modulujeme polynomem f(α). Tento popis nám zatím pro další použití Z[α]/f(α) stačí, jinak o těchto pojmech podrobněji pojednává jeden z článků T. Rosy (viz infotipy).

Nyní uvažujme Z_n, okruh celých čísel modulo n (není to těleso, protože n je naše složené číslo, které chceme faktorizovat) a celé číslo m takové, že f(m) ≡ 0 (mod n). Operace modulo n je zde významná, protože f je ireducibilní a m by neexistovalo. Nyní v okruhu celých čísel Z_n máme f(m) ≡ 0 (mod n) a v okruhu (dokonce tělese) polynomů Z[α]/f(α) je f(α) nulový prvek. To nám napovídá, že mezi oběma okruhy můžeme definovat *homomorfismus* φ: $Z[\alpha]/f(\alpha) \rightarrow Z_n$ tak, že φ (q(α)) = (q(m) mod n) pro libovolný polynom q, neboli $\phi(a_{d-1}\alpha^{d-1} + a_{d-2}\alpha^{d-2} + \ldots + a_0) = (a_{d-1}m^{d-1} + a_{d-2}m^{d-2} + \ldots + a_0) \bmod n$ a také φ(α) = (m mod n). V dalším využijeme té vlastnosti homomorfismu, že platí φ(polynom1 * polynom2) = φ(polynom1) * φ(polynom2).

2. FÁZE – PROSÉVÁNÍ

Nejprve si řekněme hlavní myšlenku GNFS. V této fázi hledáme cílově množinu S párů vzájemně nesoudělných čísel (a, b) tak, aby:

(A) součin čísel $\Pi_{(a,b)\in S}\,(a + bm)$ byl čtvercem v Z, // = x^2;

(B) součin polynomů $\Pi_{(a,b)\in S}\,(a + b\alpha)$ byl čtvercem v Z[α], // = β^2.

Pokud jsou uvedené výrazy čtvercem, existuje číslo x ∈ Z, které je odmocninou prvního výrazu, a existuje polynom β ∈ Z[α], který je odmocninou druhého výrazu. Nyní můžeme uvažovat x:= x mod n, aby x ‹ n (x ∈ Z_n) a β:= β mod f(α), aby stupeň polynomu β byl menší než →

→ stupeň polynomu $f(\alpha)$ ($\beta \in Z[\alpha]/f(\alpha)$). Protože homomorfismus ϕ převádí součinitele prvního výrazu na součinitele druhého výrazu, neboť podle definice platí $\phi(a + b\alpha) = (a + bm) \bmod n$, převádí se oba dva celé součiny homomorfismem na sebe. Tudíž platí
(R) $\phi(\beta^2) = (x^2 \bmod n)$ pro vhodná $x \in Z_n$ a $\beta \in Z[\alpha]/f(\alpha)$.

Označme y ten prvek ze Z_n, který je obrazem polynomu β ze $Z[\alpha]/f(\alpha)$, tj. $\phi(\beta) = y$. Potom díky homomorfismu platí také $\phi(\beta^*\beta) = y^2 \bmod n$. Odtud a z rovnosti (R) plyne $y^2 \equiv x^2 \pmod n$, což bylo naším cílem! Hledaný faktor n se pak vypočte známým trikem jako $\gcd(x - y, n)$. Poznamenejme ještě, že pravděpodobnost, že $x \pm y \equiv 0 \pmod n$, je malá, takže faktor n se určí skoro vždy. Samozřejmě zde vzniká mnoho otázek, například jak volit polynom $f(\alpha)$, číslo m, jak najít odmocninu β aj. Tyto otázky musíme nechat nezodpovězené, neboť co otázka, to problém na samostatnou kapitolu odborné knihy.

Nyní se vraťme k vlastnímu prosévání. Je to příhodný název, protože (a_i, b_i) vyhledáváme v „mříži" celých čísel ‹-A, A› x ‹ 0, B›, a navíc chceme, aby $(a_i + b_i m)$ a $(a_i + b_i\alpha)$ byly hladké a (a_i, b_i) co nejmenší. Množinu všech takto nalezených dvojic (a_i, b_i) označme D. O pojmu hladké číslo *(p-smooth)* a o prosévání si můžete podrobněji počíst v článcích [ROSA], viz infotipy. Protože nalezená čísla $(a_i + b_i m)$ jsou hladká, je možné je zapsat ve tvaru $\Pi_{pj \in F} p_j^{eij}$, kde F je zvolená faktorová báze – co nejmenší množina co nejmenších prvočísel p_j ($p_1 ‹ p_2 ‹ p_3 ‹ \ldots$). Každé z čísel $(a_i + b_i m)$ je tedy součinem nějakých mocnin nějakých prvočísel z množiny F. Prosévání končí nalezením množiny D.

3. FÁZE – ZPRACOVÁNÍ MATICE

V této fázi potom hledáme takovou podmnožinu $S \subseteq D$ všech nalezených párů (a_i, b_i), aby se exponent u každého prvočísla p_j v jim odpovídajícím součinu (A) poskládal z exponentů e_{ij} jednotlivých činitelů $(a_i + b_i m)$ tak, aby byl sudý. Jakmile se to podaří, číslo (A) je součinem sudých mocnin prvočísel, tedy je to čtverec, který můžeme odmocnit (k čemuž směřujeme). Podobně se postupuje i u polynomů. Hledání podmnožiny S vůči číslům a hledání podmnožiny S vůči polynomům musí být ale koordinované, protože vztahy (A) a (B) musí platit současně.

Výsledkem fáze prosévání je množina D; nyní si povšimněme, jak se zpracuje, tj. jak se nalezne její podmnožina S. Vysvětlíme to na číslech, tj. na vztahu (A). Z každého čísla $(a_i + b_i m) = \Pi_{pj \in F} p_j^{eij}$ vytvoříme jeden řádek výsledné matice, který obsahuje exponenty $(e_{i1}, e_{i2}, e_{i3} \ldots)$ prvočísel $p_1, p_2, p_3, \ldots$ Přitom ve skutečnosti nezapisujeme e_{ij}, ale jen $e_{ij} \bmod 2$, a později s maticí pracujeme také v modulu 2, protože nás koneckonců zajímá jen to, je-li e_j sudé nebo liché.

Pro každou dvojici čísel (a, b) z D tedy vznikne jeden řádek výsledné binární matice. V ní pak hledáme takovou podmnožinu řádků, aby součet jejich prvků po sloupcích byl vždy sudý (ve skutečnosti 0 modulo 2). Množinu S pak tvoří ty dvojice (a, b), které odpovídají vybraným řádkům. Poznamenejme, že toto úsilí musí být koordinováno s podobným postupem pro polynomy. Jak je vidět, třetí fáze GNFS je náročná na paměť, protože musíme vyhledávat lineární závislosti řádků a příslušné operace provádět v celé matici.

4. FÁZE – VÝPOČET FAKTORŮ

V poslední fázi se pak už jen objevené závislosti využijí k výpočtu hledaných prvočinitelů faktorizovaného čísla n. Jak právě uvedené myšlenky ukazují, faktorizace velkých čísel metodou GNFS je rozhodně netriviální záležitost. Jestliže jsme u Pollardových metod mohli napsat faktorizační program „na koleně“ a pro domácí počítač, tady to už nepůjde. Pokud máte zájem o podrobnosti, v infotipech naleznete souhrnnou monografii o GNFS.

PRAKTICKÉ MOŽNOSTI GNFS

Výhodou fáze prosévání je, že může být uskutečňována nezávisle a paralelně na různých strojích – například na internetu. Každý stroj pak nalezenou dvojici (a, b) posílá do centrálního počítače. Naproti tomu zpracování matice je většinou prováděno na jednom centrálním (super)počítači, protože vyžaduje velké množství operační paměti na uložení matice. Dnes je ještě nejužším hrdlem strojový čas na prosévání, v budoucnu – při faktorizaci větších čísel – se jím může stát paměť tohoto superpočítače, pokud nebudou nalezeny efektivní metody paralelního zpracování matice.

Možnosti faktorizace metodou GNFS, jak je odhaduje společnost RSASI, ukazuje tabulka 1. Vycházejí z předpokladu, že jsou využity **současné** schopnosti GNFS a že řešení by mělo být nalezeno **do jednoho roku,** a zcela se zde **zanedbává fáze zpracování matice** (tj. časové i paměťové nároky této fáze).

Délka čísla n, které má být faktorizováno metodou GNFS (v bitech)	Nezbytný počet PC Pentium 500 MHz nebo ekvivalentních strojů	Požadovaná paměť RAM každého stroje
430	1	minimální
760	215 000	4 GB
1020	342 000 000	170 GB
1620	1 600 000 000 000 000	120 TB

Tab. 1. Prognóza možností faktorizace společnosti RSASI, založená na současných znalostech, s požadovaným výsledkem do 1 roku

KVADRATICKÉ SÍTO

Metodou, která předcházela GNFS, a která je pro čísla do 110 až 120 cifer dokonce úspěšnější než GNFS, je metoda *Quadratic Sieve* (QS).

Vstup: složené číslo n (které není mocninou prvočísla)
Výstup: odpověď netriviální prvočinitel d čísla n

1. Vyber faktorovou bázi F = $\{p_1, p_2, ..., p_t\}$, kde $p_1 = -1$ (to děláme proto, že čísla b v dalším mohou být i záporná) a p_j ($j \geq 2$) je (j-1)ní prvočíslo p, pro něž je n kvadratické residuum modulo p (tato podmínka na prvočísla z faktorové báze je důsledkem podstaty úlohy).
2. Vypočti m = $[n^{1/2}]$, kde [] je celá část čísla.
3. Nasbírej t+1 párů čísel (a_i, b_i) podle následujícího postupu nebo metodou popsanou v textu. Nastav i = 1 a dokud i ≤ t+1, prováděj následující:
   - 3.1. Vypočti $b = q(x) = (x + m)^2$ – n a testuj, zda b je p_t – hladké (tj. hladké vzhledem k množině S). Pokud ne, zvol další x (jsou volena v pořadí ±1, ±2, ±3 ...) a opakuj krok 3.1.
   - 3.2. Jestliže b je p_t – hladké, lze ho vyjádřit ve tvaru $\Pi_{j=1...t}\ p_j^{e_{ij}}$, a proto se volí $a_i = x + m$, $b_i = b$ a $v_i = (v_{i1}, v_{i2}, ..., v_{it})$, kde $v_{ij} = e_{ij} \bmod 2$ pro j = 1 ... t.
   - 3.3. i = i + 1.
4. Nalezni neprázdnou podmnožinu $T \subseteq \{1, 2, ..., t + 1\}$ tak, že (součet modulo 2) $\Sigma_{i \in T}\ v_i = 0$.
5. Vypočti $x = \Pi_{i \in T}\ a_i \bmod n$, $y = \Pi_{j=1...t}\ p_j^{k}$, kde $l_j = \Sigma_{i \in T}\ e_{ij} / 2$ pro každé j = 0...t.
6. Jestliže $x \equiv \pm y \pmod n$, pak se vrať na krok 4 a najdi jinou podmnožinu T. Pokud by se jiná podmnožina už nenašla, nahraď několik párů (a_i, b_i) novými páry v kroku 3 a pokračuj dále.
7. Vypočti d = gcd(x – y, n).
8. Return(d).

Obr. 1. Pseudokód algoritmu kvadratického síta pro faktorizaci

1. Vybereme faktorovou bázi F = {-1, 2, 3, 5, 13, 23} o velikosti t = 6.
2. Vypočteme m = $[\ 24961^{1/2}] = 157$.
3. V tabulce jsou zaznamenány hodnoty pro prvních t + 1 hodnot x, pro něž je q(x) 23-hladké.

i	x	q(x)	faktorizace q(x)	a_i	v_i
1	0	-312	-2^3*3*13	157	(1; 1; 1; 0; 1; 0)
2	1	3	3	158	(0; 0; 1; 0; 0; 0)
3	-1	-625	-5^4	156	(1; 0; 0; 0; 0; 0)
4	2	320	2^6*5	159	(0; 0; 0; 1; 0; 0)
5	-2	-936	-2^3*3^2*13	155	(1; 1; 0; 0; 1; 0)
6	4	960	2^6*3*5	161	(0; 0; 1; 1; 0; 0)
7	6	-2160	-2^4*3^3*5	151	(1; 0; 1; 1; 0; 0)

Příklad faktorizace čísla 24961 metodou kvadratického síta

4. Nalezneme závislost: $v_1 + v_2 + v_5 = 0$, tj. T = {1, 2, 5}.
5. Vypočteme $x = a_1a_2a_5 \bmod n = 936$.
6. Vypočteme $l_1 = 1$, $l_2 = 3$, $l_3 = 2$, $l_4 = 0$, $l_5 = 1$, $l_6 = 0$.
7. Vypočteme $y = -2^{2}*3^{2}*13 \bmod n = 24025$.
8. Protože $936 \equiv -24025 \pmod n$, musíme nalézt jinou lineární závislost.
9. Nalezneme $v_3 + v_6 + v_7 = 0$, tj. T = {3, 6, 7}.
10. Vypočteme $x = a_3a_6a_7 \bmod n = 23405$.
11. Vypočteme $l_1 = 1$, $l_2 = 5$, $l_3 = 2$, $l_4 = 3$, $l_5 = 0$, $l_6 = 0$.
12. Vypočteme $y = -2^{5}*3^{2}*5^{3} \bmod n = 13922$.
13. Nyní nalezneme d = gcd(x – y, n) = gcd(23405 – 13922, 24961) = gcd(9483, 24961) = 109, čili 24961 = 109 * 229 je hledaná faktorizace.

Obr. 2. Příklad kvadratického síta pro faktorizaci čísla n = 24961

Byla v Chipu už popsána v článcích o TWINKLE (viz infotipy, [ROSA]). Nebudeme ji už proto znovu celou popisovat, ale přiblížíme si ji ve formě algoritmu a příkladu.

Vlastní algoritmus sledujeme na obrázku 1, přičemž na obrázku 2 vidíme postup na konkrétním čísle. Podstatný je krok 3, v němž hledáme t + 1 (co nejvíce) párů čísel (a_i, b_i) tak, aby $a_i^2 \equiv b_i \pmod n$. V kroku 4 z nich vybíráme takovou podmnožinu T, aby součin čísel b_i obsahoval jen sudé mocniny prvočísel, tj. aby to byl nějaký čtverec (y^2). Součin odpovídajících čísel a_i^2 bude samozřejmě také čtverec (x^2), viz krok 5, a bude platit hledaná rovnost $x^2 \equiv y^2 \pmod n$. V kroku 7 pak pomocí x, y určíme faktor čísla n. V kroku 3.1 algoritmu nejprve vytváříme čísla b a poté zjišťujeme, zda jsou hladká. Metoda QS to dělá trochu rafinovaněji – čísla b s touto vlastností nám zůstanou jako výsledek prosévání.

QS je trochu podobná vyhledávání prvočísel pomocí tzv. *Eratosthenova síta*, které ukazuje obrázek 3; možná si na ně vzpomenete ještě z dětství. Napsali jsme prostě do řádku přirozená čísla a vyškrtli jsme z nich všechna dělitelná dvojkou (kromě dvojky samé, která je nejmenším prvočíslem). První zbylé číslo (trojka) je další nejmenší prvočíslo. Ze zbývající množiny jsme tedy vyškrtli všechna čísla dělitelná trojkou, poté pětkou atd. Čísla, která zůstala nepřeškrtnuta, jsou prvočísla.

U QS nevyšetřujeme pochopitelně všechna přirozená čísla, ale jen omezenou množinu (přesněji posloupnost) čísel, kterou označíme Q = {q(x); x = ±1, ±2, ±3, ... ±M}, kde q(x) je zvolený kvadratický polynom a M je vhodná hranice. Účelem je zjistit, která čísla z Q jsou hladká, tj. rozložitelná

2	3	~~4~~	5	~~6~~	7	~~8~~	9	~~10~~	11	~~12~~	13	~~14~~	15	~~16~~	17	~~18~~	19	~~20~~	21	~~22~~	23
	3	~~4~~	5	~~6~~	7	~~8~~	~~9~~	~~10~~	11	~~12~~	13	~~14~~	~~15~~	~~16~~	17	~~18~~	19	~~20~~	~~21~~	~~22~~	23
			5	~~6~~	7	~~8~~	~~9~~	~~10~~	11	~~12~~	13	~~14~~	~~15~~	~~16~~	17	~~18~~	19	~~20~~	~~21~~	~~22~~	23
					7	~~8~~	~~9~~	~~10~~	11	~~12~~	13	~~14~~	~~15~~	~~16~~	17	~~18~~	19	~~20~~	~~21~~	~~22~~	23
									11	~~12~~	13	~~14~~	~~15~~	~~16~~	17	~~18~~	19	~~20~~	~~21~~	~~22~~	23
											13	~~14~~	~~15~~	~~16~~	17	~~18~~	19	~~20~~	~~21~~	~~22~~	23
															17	~~18~~	19	~~20~~	~~21~~	~~22~~	23
																	19	~~20~~	~~21~~	~~22~~	23
																					23

Obr. 3. Eratosthenovo síto pro určování prvočísel

na součin prvků z určené faktorové báze F. Princip QS si můžeme vysvětlit jako analogii Eratosthenova síta následovně. Prvky z Q nevyškrtáváme, ale postupně dělíme nalezenými prvočíselnými děliteli. F je předem určená množina prvočísel, opět menších než nějaká stanovená hranice.

Prosévame tak, že pro každé prvočíslo $p \in F$ vytvoříme síto $S_p = \{x_{1,2} + k*p; |x_{1,2} + k*p| \le M, k = \pm1, \pm2, \pm3, ...\}$, kde $x_{1,2}$ jsou kořeny rovnice $q(x) \equiv 0 \pmod p$. Pointa je v tom, že pro prvky síta $x \in S_p$ jsou odpovídající prvky $q(x) \in Q$ dělitelné číslem p. Je tomu tak proto, že q je kvadratický polynom, a pokud ke kořenu $x_{1,2}$ rovnice $q(x) \equiv 0 \pmod p$ připočteme násobek čísla p, dostaneme opět $q(x_{1,2} + k*p) \equiv 0 \pmod p$, neboť zde přibudou navíc jen členy dělitelné p a p^2. Čísla $q(x_{1,2} + k*p)$ – ještě nemodulovaná – jsou tedy dělitelná číslem p.

Vlastní prosévání pak konkrétně provedeme tak, že ta čísla q(x) z Q, jejichž index x je ze síta S_p, vydělíme číslem p (resp. jeho nejvyšší možnou mocninou p, kterou je dané číslo dělitelné). Pak pokračujeme

n	Proporcionální složitost L(N)	Odpovídající složitost 2^x, tj. odpovídající délka klíče symetrické šifry	Časová náročnost úlohy vůči n = 512	Paměťová náročnost úlohy vůči n = 512
512	1,76E+19	64	1	1
576	1,91E+20	67	11	3
640	1,79E+21	71	102	10
704	1,47E+22	74	836	29
768	1,08E+23	77	6135	78
1024	**1,32E+26**	**87**	**7491286**	**2737**
1536	1,31E+31	103	7,47E+11	8,64E+05
2048	1,53E+35	117	8,73E+15	9,34E+07
2560	4,74E+38	128	2,70E+19	5,19E+09
3072	5,80E+41	139	3,30E+22	1,82E+11
3584	3,58E+44	148	2,04E+25	4,51E+12
4096	1,29E+47	156	7,34E+27	8,57E+13

Tab. 2. Složitost úlohy faktorizace metodou GNFS

v „prosévání“ dalším prvočíslem z faktorové báze F. Nakonec nám z některých čísel množiny Q zůstane pouze jednička, což znamená, že původní číslo, které bylo na tomto místě, bylo dělitelné prvočísly z faktorové báze neboli je to hladké číslo.

KDYBY VŠECHNY POČÍTAČE SVĚTA...

Nyní se pokusme odpovědět na otázku z minulého dílu, jak velké číslo bychom právě teď byli schopni faktorizovat, kdybychom měli všechny znalosti současné vědy, hodně peněz a veškerou výpočetní techniku na Zemi. Abychom mohli odpovědět, musíme si ujasnit složitost této úlohy.

SLOŽITOST ÚLOHY FAKTORIZACE

Složitost úlohy faktorizace je shora omezená číslem $L(N) = \exp((c + o(1))*(\log N)^{1/3}*(\log \log N)^{2/3})$, kde $c = 1{,}923$ a $o(1)$ je veličina jdoucí k nule, když N jde do nekonečna. Nároky na čas jsou proporcionální číslu L(N) a na paměť odmocnině z L(N). Tyto nároky na paměť i čas se týkají jak fáze prosévání, tak fáze řešení matice. Čísla L(N) pro různá $N = 2^n$, která nás zajímají, ukazuje tabulka 2. Z ní je možné odvodit, že faktorizace 768bitového modulu je cca 6135krát časově a 78krát paměťově náročnější než faktorizace 512bitového modulu.

Zabývejme se nyní úlohou **faktorizace 768bitového modulu**. V nejnovější práci Lenstry a Shamira (konstruktér TWINKLE), přednesené společně s výsledky faktorizace 512bitového modulu na konferenci Eurocrypt v minulém roce, se odhaduje, že k řešení úlohy **prosévání** by bylo potřeba cca **90 000 PC** (každý s 5 GB RAM) a **po roce** jejich práce bychom obdrželi asi jednoterabajtovou matici, řešitelnou na jednom PC s odpovídajícím 1 TB paměti RAM asi 4000 let...

Jediným východiskem je proto paralelizovat také tuto úlohu **zpracování matice**, k čemuž slouží tzv. *blokový Lanczosův paralelizační algoritmus* (BLPA). Podle něj se matice rozdělí na *k* částí, které paralelně zpracovávají různé stroje, a ty pak díky vzájemné komunikaci mohou ve velké matici nalézt lineární vztahy (znamená to tedy propojení všech těchto počítačů). Dosud nebyl BLPA vyzkoušen pro k › 16, zatímco v naší úloze potřebujeme **k = 80 000**, abychom mohli úlohu řešit do **tří měsíců** (zde použijeme pochopitelně 80 000 PC z první fáze – nemusíme tedy nakupovat další).

Zda je něco takového vůbec uskutečnitelné, je předmětem velkého sporu Silvermana na jedné straně (BLPA není vyzkoušen, nebude fungovat pro k = 80 000, problém rychlého vzájemného propojení 80 000 PC vytvářejících ve skutečnosti jeden stroj s prostorově oddělenou pamětí a procesory...) a Lenstry na straně druhé, který žádnou z uvedených námitek za problém nepovažuje. (Fakt ale je, že oba pánové jsou sice odborníky na faktorizaci, ovšem nikoli na paralelní architektury.)

Ale dejme tomu, že by se to podařilo. Pak bychom k vyřešení úlohy potřebovali 15 měsíců (12 měsíců prosévání + 3 měsíce řešení matice) a cca (90 000 PC)*(2000 USD/PC) = 180 000 000 USD. Pokud se vám nelíbí odhad 2000 USD za PC s 5GB RAM, dosaďte si sem své číslo; v tomto okamžiku jde však především o to, zda obdržená čísla jsou vůbec („pro lidstvo“) dosažitelná nebo ne.

DRUHÁ CESTA ŘEŠENÍ

Další možností řešení jsou specializovaná zařízení. Zatím známe jednoho představitele – TWINKLE, není však vyloučeno, že tajné služby mají něco lepšího, a kdyby lidstvu opravdu „teklo do bot“, nenechaly by si to pro sebe...

Ale zanechme „kdyby“ a dejme slovo prof. Shamirovi. Společně s Lenstrou odhadují na fázi prosévání při faktorizaci 768bitového modulu potře-

bu 5000 zařízení TWINKLE podporovaných 80 000 počítači (Pentium II s minimální pamětí RAM, cena cca 100 USD), které ve fázi prosévání (6 měsíců) pro TWINKLE připravují data (na jeden TWINKLE cca 16 PC). Cena za upravený TWINKLE je cca 5000 USD, takže za fázi prosévání zaplatíme 5000*5000 + 80 000*100, tj. 33 milionů dolarů (6 měsíců výpočtů). Jak vidíte, TWINKLE může fázi prosévání zlevnit, což je výborné. K druhé fázi můžeme využít počítače z první fáze, ale problém, zda algoritmus BLPA pro k = 80 000 je realizovatelný, zůstává otevřený jako v předchozím případě. Jinak řečeno – zařízení TWINKLE nám trochu zlevnilo první fázi, ale s hlavním problémem nám nijak nepomohlo.

JE OHROŽEN 1024BITOVÝ MODUL?

Z tabulky 2 lze odvodit, že faktorizace 1024bitového modulu je 7491286/6135 = 1221krát časově a 35krát paměťově náročnější než faktorizace čísla 768bitového. Tentokrát však TWINKLE na fázi prosévání použít nepůjde, protože úloha přesahuje jeho technické možnosti. Se zvyšováním modulu totiž roste i velikost faktorové báze neboli počet diod. Ten je u TWINKLE omezen na 200 000 (střízlivěji, vzhledem k chybovosti, se uvažuje 100 000 funkčních diod), což je pro 1024bitový modul málo. Nemusí nás to ale nijak zvlášť mrzet – jak dovozují Lenstra se Shamirem, TWINKLE oproti použití PC může fázi prosévání nanejvýše zlevnit. Zlevnění by bylo maximálně o jeden řád a peníze nás v tomto případě tak nezajímají, takže můžeme tuto cestu opustit.

Co budeme potřebovat, kdybychom nasadili PC, ukazuje tabulka 3. Nejprve si řekněme, jak tabulka vznikla. První řádek tabulky je převzat z originálního článku *Factorization of a 512-Bit RSA Modulus, Eurocrypt 2000, pp. 1 – 17* kolektivu autorů (Lenstra, Montgomery a další), kteří faktorizovali 512bitový modul RSA-155. Všechna ostatní čísla vznikla pouhým násobením koeficientem složitosti, odvozeným z čísel L(N) a L(512) jako L(N)/L(512) u časové náročnosti a jako odmocnina z tohoto koeficientu u paměťové náročnosti. Číslo L(N) i uvedené koeficienty složitosti jsou uznávány v táborech optimistů i pesimistů. Zdůrazňujeme to, protože v různých výpočtech různých autorů se berou v úvahu různé předpoklady. Například údaje o počtu PC z tabulky 1 (Silverman) byly vypočteny na základě úlohy RSA-155, kde počítače nebyly využity pro faktorizaci na 100 %. Proto (a také z důvodu jiného taktování) je jich více, než je uváděno v tabulce 3, kde se předpokládá plné využití a vyšší výkon.

Dále poznamenejme, že jako referenční počítač uvádíme PC/450MHz, který je sice už obstarožní, ale slouží jako etalon v různých pracích. Jak tedy vypadá 1024bitový modul podle tabulky 3? Postačí vyrobit desítky milionů PC (například 139 milionů PC/450MHz nebo ekvivalentně 13,9 milionu PC/4500MHz), každý disponující 175 GB RAM. Poznamenejme jen, že to musí být už PC se 64bitovými procesory, aby takovou paměť mohly přímo adresovat. S tímto vybavením pak budeme za rok hotovi s fází prosévání. Nebo investujeme dvojnásobek a výsledek bude za půl roku atd.

Pak ale opět přijde na řadu problém umístění a zpracování matice, tentokrát zabírající 5500 GB RAM. Pokud bychom tuto paměť soustředili do jednoho PC, řešení by nám na něm trvalo 75 milionů dní. Takže budeme muset úlohu paralelizovat. Jenže jak by fungoval algoritmus BLPA pro k = 139 000 000 paralelních počítačů, když jsme ještě nevyzkoušeli ani k = 16? Jak by asi fungovalo propojení takového množství počítačů zpracovávajících matici paralelně po blocích? To nevíme a pro tak velké počty nikdo o takovém postupu neuvažuje.

Jsou tu sice čistě teoretické předpoklady, že by mohl existovat pokrok, který tyto problémy překoná, ale pokud zůstaneme v realitě, nemáme po ruce nic. Proto se osobně domnívám, že v tento okamžik na 1024bitový

n	Prosévání Počet PC/450MHz, řešících úlohu nepřetržitě 1 rok (*)	Prosévání Nutně vynaložený skutečný výpočetní výkon (v MIPS letech)	Prosévání Nutná paměť každého jednotlivého PC pro řešení úlohy prosévání (v MB)	Zpracování matice Nutná paměť superpočítače pro zpracování matice (v GB)	Zpracování matice Čas superpočítače ke zpracování matice (dní)
512	18,7	8400	64	2	10
576	203	91547	211	7	109
640	1901	855541	646	20	1019
704	15602	7020966	1850	58	8358
768	114513	51530771	5013	157	61346
1024	**139837347**	**62926806060**	**175169**	**5474**	**74912864**
1536	1,39E+13	6,27E+15	5,53E+07	1,73E+06	7,47E+12
2048	1,63E+17	7,33E+19	5,98E+09	1,87E+08	8,73E+16
2560	5,03E+20	2,26E+23	3,32E+11	1,04E+10	2,70E+20
3072	6,17E+23	2,78E+26	1,16E+13	3,64E+11	3,30E+23
3584	3,80E+26	1,71E+29	2,89E+14	9,02E+12	2,04E+26
4096	1,37E+29	6,16E+31	5,48E+15	1,71E+14	7,34E+28

Tab. 3. Složitost faktorizace metodou GNFS pro různé délky čísel a nároky na výkon, čas a paměť
(*) PC/450MHz je často volený referenční stroj, viz též [LV]

modul prostě nestačíme a případní luštitelé 1024bitového modulu RSA budou muset přijít s nějakým nápadem, jak se vyrovnat s druhou fází, nebo s něčím úplně jiným, než je GNFS. Třeba s kvantovými počítači, viz například [VESMIR] v infotipech.

JAK TO VIDÍ OPTIMISTÉ (NEBO IGNORANTI?)

Zcela jiný dojem musí člověk získat při čtení práce [LV], jejíž závěry jsou v zásadním rozporu se Silvermanovými úvahami a také s úvahami v článku, který čtete. Podle [LV] 1024bitový modul RSA nebude – zjednodušeně řečeno – poskytovat příliš velkou bezpečnost už v příštím roce (i když přesná interpretace vyslovených závěrů je trochu složitější). [LV] je velmi sugestivně napsána, takže pokud ji člověk čte jen letmo, snadno připustí několik dílčích rafinovaných extrapolací. Pokud se zkombinují dohromady, celkový závěr práce je dost zdrcující.

Na malém prostoru není možné rozvíjet hlubší kritiku, ale za závažnou považuji Silvermanovu výhradu, že [LV] nebere v úvahu paměťové nároky faktorizace. S těmi se [LV] vyrovnává na straně 26 takto: *„...protože současné procesory mají dostatek paměti pro problémy řešené metodou NFS v současnosti, můžeme předpokládat, že budoucí procesory budou mít více než dost paměti k řešení budoucích problémů...“*. Vážným zájemcům ale doporučuji seznámit se s oběma názory, vzhledem k tomu, že autoři [LV] jsou uznávaní odborníci.

ZÁVĚR

Jak velké číslo bychom tedy nyní dokázali faktorizovat, kdybychom měli všechny znalosti současné vědy, hodně peněz a veškerou výpočetní techniku na Zemi? Tabulka 2 dává každému dost možností na vlastní odpověď. Pokroky ve faktorizaci jsou zatím velmi skromné. Faktem zůstává, že nejlepší současnou metodou faktorizace je metoda GNFS, jejíž pomocí bylo v roce 1999 faktorizováno 512bitové číslo (155 desítkových číslic). Další pokroky lze sice očekávat, ale pokud nebude učiněn zásadní objev, praktickým limitem GNFS se zdá být 768 bitů (což je také moje osobní odpověď na vznesenou otázku). Názory na možnosti se ovšem různí – paradoxně přímo diametrálně u největších odborníků pracujících v oblasti teorie čísel.

Vlastimil Klíma | vlastimil.klima@i.cz

INFOTIPY

Monografie o GNFS:
[LL] Lenstra, A. K., Lenstra, H. W.: The development of the number field sieve, Springer-Verlag, Berlin, 1993

O faktorizačních metodách:
Menezes, A. J., Oorschot, P. C., Vanstone, S. A.: Handbook of Applied Cryptography, CRC Press, New York, 1997

O problematice těles polynomů:
Rosa, T.: V klidu a bezpečí, díl 8, Chip 6/00, str. 178 – 180

Vše o nové soutěži:
- *http://www.rsasecurity.com/rsalabs/challenges/factoring/index.html*

O faktorizaci a zařízení TWINKLE:
[ROSA] Rosa, T.: Na to vezmi LED!, Chip 8/99 a 9/99; Jde to i bez Twinklu, Chip 10/99

Bezpečnost a faktorizace podle Silvermana:
- *http://www.rsasecurity.com/rsalabs/bulletins/bulletin13.html*

Bezpečnost a faktorizace podle Lenstry a Verheula:
[LV] Lenstra, A. K., Verheul, E. R.: Selecting Cryptographic Key Sizes, PKC2000, Australia, January 2000, nyní aktualizováno na
- *http://www.cryptosavvy.com/joc.pdf*

O kvantových počítačích:
[VESMIR] Biskup, M., Cejnar, P., Kotecký, R.: Kvantové počítače, Vesmír, roč. 76, květen 1997, str. 250 – 254

Archiv článků:
Zmíněné články z Chipu naleznete také v elektronické podobě na
- *http://www.decros.cz/bezpecnost/_kryptografie.html*