Čínská věta o zbytcích
RSA

Matematické algoritmy (11MAG)

Jan Přikryl

4. přednáška 11MAG
pondělí 21. října 2013

verze: 2013-10-23 16:56

Obsah

1 Čínská věta o zbytcích

Více vzájemně ekvivalentních tvrzení z algebry a teorie čísel. Nejstarší zmínka z Číny ve 3. století našeho letopočtu.

Problém 1. *Jak najít x, jenž je řešením více kongruencí najednou, například*

$$x \equiv 2 \pmod 3,$$
$$x \equiv 3 \pmod 5,$$
$$x \equiv 2 \pmod 7?$$

Zbytkové třídy jsou $[2]_3$, $[3]_5$ a $[2]_7$, výsledné řešení musí spadat do všech tří z nich.

Výsledkem bude opět kongruence, jejíž modul je dán násobkem tří dílčích modulů soustavy kongruencí,

$$x = 3i + 2 = 5j + 3 = 7k + 2$$
$$x = \dots$$

Zbytková třída výsledku bude tedy $3 \cdot 5 \cdot 7 = 105$.

Pokud zvolíme řešení jako lineární kombinaci třech dílčích řešení pro jednotlivé kongruence, tedy

$$x \equiv 2\kappa_1 + 3\kappa_2 + 2\kappa_3 \pmod{105}$$

stačí pro dodržení ekvivalencí v soustavě kongruencí zaručit, aby se κ_1 chovalo jako 1 ve zbytkové třídě 3 a jako 0 ve zbytkových třídách 5 a 7, a aby se κ_2 chovalo jako 1 ve zbytkové třídě 5 a jako 0 ve zbytkových třídách 3 a 7, a obdobně aby κ_3 bylo ve zbytkových třídách $[1]_7$, $[0]_3$ a $[0]_5$. Potom bude platit

$$2\kappa_1 + 3\kappa_2 + 2\kappa_3 \equiv 2 \pmod 3,$$
$$2\kappa_1 + 3\kappa_2 + 2\kappa_3 \equiv 3 \pmod 5,$$
$$2\kappa_1 + 3\kappa_2 + 2\kappa_3 \equiv 2 \pmod 7.$$

V prvním kroku hledáme nulové a jednotkové zbytkové třídy pro kombinace původních modulů. V našem případě platí

$$\kappa_1 = 70 \equiv 0 \pmod{5 \cdot 7} \quad \wedge \quad \kappa_1 = 70 \equiv 1 \pmod 3,$$
$$\kappa_2 = 21 \equiv 0 \pmod{3 \cdot 7} \quad \wedge \quad \kappa_2 = 21 \equiv 1 \pmod 5,$$
$$\kappa_3 = 15 \equiv 0 \pmod{3 \cdot 5} \quad \wedge \quad \kappa_3 = 15 \equiv 1 \pmod 7.$$

Řešením dané soustavy kongruencí je v takovém případě číslo

$$\hat{x} = 2 \cdot 70 + 3 \cdot 21 + 2 \cdot 15 = 233.$$

Minimální hodnota x je dána třídou kongruence modulo $3 \cdot 5 \cdot 7 = 105$, tedy

$$x = 233 \bmod 105 = 23.$$

1.1 Vlastní tvrzení

Nechť $n_1, n_2, \dots, n_k$ jsou navzájem nesoudělná přirozená čísla, $n_i \geq 2$ pro všechna $i = 1, \dots, k$. Potom řešení soustavy rovnic

$$x \equiv a_1 \pmod{n_1}$$
$$x \equiv a_2 \pmod{n_2}$$
$$\vdots$$
$$x \equiv a_k \pmod{n_k}$$

existuje a je určeno jednoznačně v modulo $n = n_1 \cdot n_2 \cdot \ldots \cdot n_k$.

Díky nesoudělnosti existuje ve třídě operací modulo n_i ke každému $N_i = n/n_i$ jeho multiplikativní inverze M_i, tedy

$$M_i \cdot N_i \equiv 1 \pmod{n_i}$$

a platí

$$x = \sum_{i=1}^{k} a_i M_i N_i.$$

Ve výše uvedeném případě se zbytkovými třídami $[2]_3$, $[3]_5$ a $[2]_7$ je

$$x = 2 \cdot 2 \cdot 35 + 3 \cdot 1 \cdot 21 + 2 \cdot 1 \cdot 15 = 233.$$

Výpočty modulo velké M lze převést na výpočty modulo menší součinitelé čísla M – zrychlení výpočtu.

Lze generalizovat pro soudělná čísla.

Význam hlavně v šifrovacích systémech.

1.2 Problém nůše s vejci

V nůši je v vajec. Pokud z ní odebíráme vejce po dvou, třech a pěti najednou, v nůši nakonec zůstane 1, 2, respektive 4 vejce. Pokud odebíráme vejce po sedmi kusech, v nůši nakonec nezůstane vejce žádné.

Jaká je nejmenší hodnota v pro niž může uvedená situace nastat?

Zbytkové třídy jsou $[1]_2$, $[2]_3$, $[4]_5$ a $[0]_7$.

Hledáme řešení soustavy

$$\begin{aligned} v &\equiv 1 \pmod{2} \\ v &\equiv 2 \pmod{3} \\ v &\equiv 4 \pmod{5} \\ v &\equiv 0 \pmod{7} \end{aligned}$$

Výsledek bude nějaká třída kongruence modulo 210.

Pro jednotlivé ekvivalence máme

i	n_i	N_i	M_i	a_i
1	2	105	1	1
2	3	70	1	2
3	5	42	3	4
4	7	30	4	0

$$\begin{aligned} v &= (1 \cdot 1 \cdot 105 + 2 \cdot 1 \cdot 70 + 4 \cdot 3 \cdot 42 + 0 \cdot 4 \cdot 30) \bmod 210 \\ &= (105 + 140 + 504 + 0) \bmod 210 = 749 \bmod 210 = 119 \end{aligned}$$

2 Modulární mocnění

Neefektivně lze opakovaným násobením a redukcí:

$$c = \underbrace{b\,[\,b\,[\ldots[\,b\cdot b}_{r-\text{krát}}\mod n\,]\ldots]\,\mod n\,]\,\mod n$$

Jde to ale i lépe. Postup si ukážeme na následujících příkladech.

Příklad 2 (Mocnění opakovaným násobením). Mám-li počítat $a \equiv 21^{41} \pmod{43}$, mohu sice postupně vyčíslovat

$$\begin{aligned} a_1 &= 21,\\ a_2 &= a_1 \cdot 21 \bmod 43,\\ a_3 &= a_2 \cdot 21 \bmod 43,\\ &\vdots\\ a = a_{41} &= a_{40} \cdot 21 \bmod 43, \end{aligned}$$

půjde ale o poměrně pracný postup – představte si, jak dlouho takto budete počítat kongruenci $a \equiv 97011687217^{98764321261} \pmod{225898512559}$, a že takovýchto kongruencí potřebujete spočítat milióny.

Vylepšení celého algoritmu spočívá v jednoduchém pozorování: Každý mocnitel r z množiny přirozených čísel můžeme reprezentovat jako součet mocnin čísla 2 a každou vysokou mocninu čísla b lze poskládat z násobků čísel $b^{(2^i)}$. Opět je asi vhodnější si celý postup ukázat na konkrétním příkladu:

Příklad 3 (Mocnění opakovaným kvadrátem). V našem ilustrativním případě $a \equiv 21^{41} \pmod{43}$ je mocnitel $41 = 32 + 8 + 1$ a platí tedy $a \equiv 21^{41} \pmod{43} \equiv 21 \cdot 21^8 \cdot 21^{32} \pmod{43}$, kde pro výpočet dvojkových mocnin čísla 21 potřebujeme pouze opakovaně umocňovat na druhou:

$$\begin{aligned} 21^2 &\equiv (21^1)^2 \pmod{43} \equiv 11 \pmod{43},\\ 21^4 &\equiv (21^2)^2 \pmod{43} \equiv 11^2 \pmod{43} \equiv 35 \pmod{43},\\ 21^8 &\equiv (21^4)^2 \pmod{43} \equiv 35^2 \pmod{43} \equiv 21 \pmod{43},\\ 21^{16} &\equiv (21^8)^2 \pmod{43} \equiv 21^2 \pmod{43} \equiv 11 \pmod{43},\\ 21^{32} &\equiv (21^{16})^2 \pmod{43} \equiv 11^2 \pmod{43} \equiv 35 \pmod{43}. \end{aligned}$$

Výsledek obdržíme jako $a \equiv 21 \cdot 21^8 \cdot 21^{32} \pmod{43} \equiv 21 \cdot 21 \cdot 35 \pmod{43}$ a po úpravách $a \equiv 41 \pmod{43}$.

To vede na algoritmus opakovaného kvadrátu, jenž můžete shlédnout v Algoritmu 1.

Počet kroků nutných pro umocnění b^r opakovaným kvadrátem je $\log_2 r$ oproti r krokům potřebným pro opakované násobení.

Přibližme si celý postup, popsaný v Algoritmu 1, ještě jedním příkladem.

Příklad 4 (Spočtěte $c = 3^{17} \bmod 7$). Nejprve rozložíme $r = 17 = 10001_\text{b}$. Je $a_0 = 1$ a proto prvotní hodnota $c = b = 3$ a $b_0 = 3$. Potom

$$\begin{aligned} b_1 &= 3^2 \bmod 7 = 9 \bmod 7 = 2, a_1 = 0,\\ b_2 &= 2^2 \bmod 7 = 4 \bmod 7 = 4, a_2 = 0,\\ b_3 &= 4^2 \bmod 7 = 16 \bmod 7 = 2, a_3 = 0,\\ b_4 &= 2^2 \bmod 7 = 4 \bmod 7 = 4, a_4 = 1. \end{aligned}$$

Algoritmus 1 Efektivní algoritmus mocnění pomocí opakovaného kvadrátu.

Require: $b \in \mathbb{Z}$, $r, n \in \mathbb{N}$
Ensure: $c \equiv b^r \pmod n$
  Nechť $r = \sum_{j=0}^{k} a_j \cdot 2^j, a_j \in \{0, 1\}$
  $c \leftarrow 1 + a_0 \cdot (b - 1)$; $b_0 \leftarrow b$
  for $j = 1$ **to** k **do**
    $b_j \leftarrow b_{j-1}^2 \bmod n$
    if $a_j > 0$ **then**
      $c \leftarrow c \cdot b_j \bmod n$
    end if
  end for
  return $c \equiv b^r \pmod n$

Nyní přepočteme $c = 3 \cdot 4 \bmod 7 = 12 \bmod 7 = 5$. Další binární cifry už v r nejsou, výsledkem je proto $c = 5$.

Kontrola: $3^{17} = 129140163 \bmod 7 = 5$.

3 Eulerova funkce

Definice 5 (Eulerova věta)**.** Malou Fermatovu větu lze zobecnit na tvar

$$a^{\phi(n)} \equiv 1 \pmod n,$$

kde $\phi(n)$ je tak zvaná **Eulerova funkce**, která udává počet přirozených čísel $1 \leq x \leq n$, jež jsou s n nesoudělná.

Někdy $\phi(n)$ označuje názvem *totient*.

Pro prvočísla je

$$\phi(p) = p - 1,$$

pro nesoudělná x a y platí

$$\phi(x \cdot y) = \phi(x) \cdot \phi(y)$$

a proto pro prvočísla p a q také

$$\phi(p)\phi(q) = (p - 1)(q - 1).$$

Pro libovolné přirozené n platí také

$$\phi(n) = n \prod_{p|n} \left(1 - \frac{1}{p}\right).$$

Pozorování 6. *Zvolíme-li $n = p \cdot q$, kde p i q jsou prvočísla, bude podle Eulerovy věty*

$$a^{\phi(n)} = a^{\phi(p \cdot q)} = a^{(p-1)(q-1)} \equiv 1 \pmod{p \cdot q},$$

a tedy také

$$\left(a^{p-1}\right)^{q-1} \equiv \left(a^{q-1}\right)^{p-1} \equiv 1 \pmod{p \cdot q}.$$

a protože p a q jsou nesoudělná, tak také

$$a^{\phi(n)} \equiv 1 \pmod p \equiv 1 \pmod q.$$

4 Šifrování

4.1 Symetrické a asymetrické šifry

Existují dvě základní skupiny šifrovacích algoritmů:

- **Symetrické šifry** u nichž se ten samý klíč používá jak k šifrování, tak i k dešifrování zprávy. Odesílatel i příjemce musí mít k dispozici identické klíče. Příkladem je DES, 3DES, AES.
- **Asymetrické šifry** u nichž se šifruje jiným klíčem, než je klíč určený k dešifrování. Odesílatel po zašifrování již nemá možnost zprávu dešifrovat. Příkladem je RSA (PGP), GnuPG, ElGamal.

Symetrické šifry jsou při stejné délce šifrovacího klíče výrazně bezpečnější, než šifry asymetrické ...

4.2 Výměna klíčů

... ale symetrické šifrování má *základní problém*: distribuci klíčů.

Diffie a Hellman, 1976

Alice a Bob se na klíči mohou dohodnout přes nezabezpečený komunikační kanál. Je pouze třeba zajistit, aby operace, jež Alice a Bob provádějí, *nebyly výpočetně snadno invertovatelné.*

Diffieho-Hellmanova výměna klíčů

Veřejně známé prvočíslo p a $\alpha \in \{2, \ldots, p-2\}$. Oba jako klíč použijí $\alpha^{xy} \bmod p$ – Alice si vymyslí veliké $x \in \mathbb{N}$ a Bobovi pošle $\alpha^x \bmod p$, Bob pošle Alici $\alpha^y \bmod p$. Alice pak provede $(\alpha^y)^x \bmod p$, Bob obdobně.

Hodnota α se v praxi volí 2 nebo 5.

Výměna klíčů je založena na faktu, že v modulární aritmetice modulo p se velmi těžko hledá **diskrétní logaritmus** celého čísla, tedy číslo $x = \log_g(h)$[1] takové, že $g^x \equiv h \pmod p$.

Příklad 7 (Diskrétní logaritmus). Uvažujme kongruenci $3^x \equiv 15 \pmod{19}$. Jedním z možných řešení je $x = 5$, neboť $3^5 \equiv 15 \pmod{19}$, není to ale řešení jediné. Z Malé Fermatovy věty totiž plyne $3^18 \equiv 1 \pmod{19}$, a proto má řešená kongruence nekonečně mnoho řešení ve tvaru $3^{5+18k} \equiv 15 \pmod{19}$ pro $k \in \mathbb{N}$. Zadanou kongruenci tedy splňují všechna x, jež jsou řešeními kongruence $x \equiv 5 \pmod{18}$ a ze zveřejněné hodnoty $3^x \bmod 19$ lze jen s velkou náhodou určit jedno konkrétní x, zvolené při výměně klíče Alicí.

Vzhledem k tomu, že $[a]_p \cdot [b]_p = [ab]_p$ platí pro Alicí přijaté Bobovo $\alpha^y \bmod p$ následující:

$$\alpha^y \bmod p \equiv [\underbrace{\alpha \cdot \alpha \cdots \alpha}_{y-\text{krát}}]_p,$$

[1] Měli bychom ještě správně uvádět, že hledané číslo x je prvkem tzv. okruhu celých čísel modulo p, zapisujeme $x \in \mathbb{Z}/p\mathbb{Z}$

a po umocnění na x-tou:

$$\left([\underbrace{\alpha \cdot \alpha \cdots \alpha}_{y-\text{krát}}]_p \right)^x = \underbrace{[\underbrace{\alpha \cdot \alpha \cdots \alpha}_{y-\text{krát}}]_p \cdot [\underbrace{\alpha \cdot \alpha \cdots \alpha}_{y-\text{krát}}]_p \cdots [\underbrace{\alpha \cdot \alpha \cdots \alpha}_{y-\text{krát}}]_p}_{x-\text{krát}} \equiv$$
$$\equiv [\underbrace{\alpha \cdot \alpha \cdots \alpha}_{xy-\text{krát}}]_p.$$

Reciprocně to platí i pro Bobem přijaté Aličino $\alpha^x \bmod p$.

Příklad výměny pro $p = 17$ *a* $\alpha = 5$

Alice si zvolí $x = 1039$.

Bob si zvolí $x = 1271$.

Po nezašifrovaném spojení pošle Alice Bobovi $5^{1039} \bmod 17 = 7$ a Bob pošle Alici $5^{1271} \bmod 17 = 10$.

Bob si spočte svůj klíč jako $7^{1271} \bmod 17 = 12$, Alice jako $10^{1039} \bmod 17 = 12$.

Ve skutečnosti budou p, α, x, y mnohem větší čísla (proč)?

Pro ilustraci situace útočníka si povšimněte, že platí $5^7 \equiv 5^{23} \equiv 5^{39} \equiv 5^{7+k\cdot 16} \equiv 10 \pmod{17}$ pro libovolné $k \in \mathbb{N}$ a že $1271 = 7 + 79 \cdot 16$. Stejně tak je $5^{15} \equiv 5^{31} \equiv 5^{47} \equiv 5^{15+k\cdot 16} \equiv 7 \pmod{17}$ a $1039 = 15 + 64 \cdot 16$. V našem ilustračním případě může útočník celkem lehce vyzkoušet všechny možné kombinace klíčů (bude jich jenom sedmnáct), ale v případě velkých prvočísel to už nebude praktické: Bude-li $p = 429183283$ a dokážeme-li otestovat 100 klíčů za vteřinu, bude prohledávání celého prostoru možných klíčů trvat přibližně 1192 hodin. Pro p z oblasti 64-bitových prvočísel by prohledávání celého prostoru možných klíčů rychlostí 10^6 klíčů/s mohlo trvat až 585 tisíc let (vyzkoušejte si to).

4.3 RSA (Rivest, Shamir a Adelman 1977)

V dnešní době asi nejznámějším algoritmem pro asymetrické šifrování (tedy pro šifrování veřejným klíčem) je algoritmus vyvinutý Ronem Rivestem, Adi Shamirem a Leonardem Adelmanem na MIT v roce 1977, označovaný zkratkou **RSA**. Až o dvacet let později vyšlo najevo, že britský matematik Clifford Cocks, zaměstnanec GHCQ,[2] navrhl v podstatě stejný systém už okolo roku 1973. GHCQ ovšem nemělo pro jeho vynález využití, a jako utajovaná informace zůstal 25 let archivován.

Šifra RSA vychází z předpokladu, že faktorizace součinu prvočísel p a q je časově náročná – všichni proto mohou znát šifrovací klíč e a šifrovací modul $n = p \cdot q$, ale nepomůže jim to ke zjištění dešifrovacího klíče, založeného na p a q.

V praxi je klíč $n = p \cdot q \in \{0,1\}^{1024}$ až $\{0,1\}^{4096}$.

Poslední faktorizovaný RSA klíč je RSA-768 ($n = \{0,1\}^{768}$, 232 dekadických číslic) za necelé 3 roky na až 618 pracovních stanicích v roce 2009.

[2] Anglicky *Government Communications Headquarters*. Je to britská obdoba NSA, vládní bezpečnostní a zpravodajská agentura zaměřená na ochranu datových komunikací a dešifrování zpráv.

Ale pozor: Už v květnu 2007 padlo $M_{1039} = 2^{1039} - 1$ za 11 měsíců v laboratořích EPFL, Uni Bonn a NTT.

Faktorizovat 1024-bitový klíč, jenž se dnes stále běžně používá v každodenním šifrovaném provozu) by podle Kleinjunga a kolegů [1] bylo asi 1000× náročnější, než jejich faktorizace 768-bitového klíče. Vzhledem k tomu, je 768-bitový klíč je na faktorizaci několik tisíckrát složitější, než 512-bitový klíč, a že mezi faktorizacemi 512 a 768-bitového klíče uplynulo přibližně 10 let, lze očekávat, že kilobitový klíč bude považován za faktorizovatelný (a tedy prolomitelný) někdy okolo roku 2015.

Poslední faktorizovaný klíč je momentálně RSA-704 (212 dekadických číslic), faktorizovat jej trvalo necelý rok na několika sítích univerzitních počítačů ve Francii a Austrálii [2].

4.3.1 Generování veřejného a soukromého klíče

Proces tvorby veřejného a soukromého šifrovacícho klíče je poměrně jednoduchý:

1. Zvolíme nepříliš si blízká prvočísla p a q. Ideální délka každého prvočísla je v dnešní době alespoň 1024 bitů.
2. Spočteme **modul** šifrovací a dešifrovací transformace, $n = p \cdot q$. Tento modul bude mít tedy ideální cca 2048 bitů.
3. Vypočteme Eulerovu funkci pro n, $\phi(n) = (p-1)(q-1)$.
4. Zvolíme **šifrovací exponent** e takový, že $1 < e < \phi(n)$ a $\gcd(e, \phi(n)) = 1$.
5. Dopočteme **dešifrovací exponent** d tak, aby d bylo multiplikativní inverzí k e modulo $\phi(n)$, $d \cdot e \equiv 1 \pmod{\phi(n)}$.

Veřejný klíč pro zašifrování zprávy je (n, e), **soukromý klíč** pro dešifrování je (n, d).

Princip přenosu zprávy X je primitivní:

Šifrování

Po lince přenášíme šifrovaný text c, jenž vznikne jako

$$c = X^e \bmod n.$$

Dešifrování

Příjemce si z přijatého šifrovaného textu spočítá původní zprávu jako

$$X = c^d \bmod n.$$

Trik celého postupu spočívá v tom, že *z pouhé znalosti* (n, e) *nelze v rozumném čase určit* d. Nejde to z toho důvodu, že pro výpočet hodnoty d potřebujeme znát $\phi(n) = (p-1)(q-1)$, je tedy třeba nejprve faktorizovat šifrovací modul n. To je ovšem pro vhodně zvolená p a q časově velmi náročná úloha.

4.3.2 Důkaz RSA

Základní otázkou, kterou bychom si nyní měli položit, je: „Obdržíme dešifrováním opravdu původní text?“ Ač se to na první pohled nezdá z výše uvedených vzorců pravděpodobné, RSA vskutku funguje.

Při dešifrování $c \equiv X^e \pmod n$ máme

$$c^d \equiv (X^e)^d \equiv X^{ed} \pmod n \equiv X^{ed} \pmod{pq}.$$

Prozkoumáme vlastnosti $c^d \equiv X^{ed} \pmod p$ a $c^d \equiv X^{ed} \pmod q$ a zobecníme je na operace modulo n.

Z definice součinu ed v algoritmu RSA plyne

$$ed \equiv 1 \pmod{\phi(n)} \Rightarrow \exists g \in \mathbb{Z} : ed = 1 + g(p-1)(q-1),$$

což můžeme dále upravit na

$$ed = 1 + f(p-1)(q-1) = 1 + g(q-1) = 1 + h(p-1)$$

a tedy

$$ed \equiv 1 \pmod{\phi(n)} \equiv 1 \pmod{\phi(p)} \equiv 1 \pmod{\phi(q)}.$$

Důkaz. $ed \equiv 1 \pmod{\phi(n)} \equiv 1 \pmod{(p-1)(q-1)} \Leftrightarrow ed = 1 + g(p-1)(q-1)$ a tedy $ed = 1 + g_p(q-1)$ a $ed = 1 + g_q(p-1)$. □

Dokazujeme nadále p a q odděleně:

Pro $p \nmid X$ je podle Malé Fermatovy věty $X^{p-1} \equiv 1 \pmod p$ a tedy

$$X^{ed} = X^{1+h(p-1)} = X \cdot X^{h(p-1)} = X \cdot \left(X^{(p-1)}\right)^h \equiv X \cdot 1^h \equiv X \pmod p.$$

Pro $p \mid X$ je

$$X^{ed} \equiv 0^{ed} \pmod p \equiv X \pmod p$$

To samé platí pro q a tedy

$$X^{ed} \equiv X \pmod p$$
$$X^{ed} \equiv X \pmod q$$

Jedním z důsledků CRT je pro nesoudělná x a y ekvivalence

$$\begin{array}{rcl} a & \equiv & b \pmod x \\ a & \equiv & b \pmod y \end{array} \quad \Leftrightarrow \quad a \equiv b \pmod{xy}. \tag{1}$$

Proto také z

$$X^{ed} \equiv X \pmod p$$
$$X^{ed} \equiv X \pmod q$$

plyne

$$X^{ed} \equiv X \pmod{pq}.$$

Důkaz. Platnost ekvivalence (1) lze dokázat za předpokladu nesoudělnosti x a y tak, že uvážíme nejprve

$$\begin{aligned} a &= k_1 x + b, & a &= k_2 y + b, \\ a - b &= k_1 x, & a - b &= k_2 y. \end{aligned}$$

Čísla x a y musí dělit výraz $a - b$ beze zbytku,

$$a - b = k_1 x = k_2 y,$$

a protože jde o nesoudělná čísla, musí zároveň platit, že $y \mid k_1$ a $x \mid k_2$,

$$k_1 x = k_2 y = (\kappa y)x = (\kappa x)y.$$

Můžeme tedy psát

$$a = \kappa \cdot xy + b$$

a tedy také

$$a \equiv b \pmod{xy}$$

□

4.4 CRT-RSA

V úvodu přednášky zmíněná Čínská věta o zbytcích má velmi zajímavé využití právě při výpočtech dešifrovací části šifry RSA.

Problém 8 (Dešifrování RSA)**.** *Jak modul n, tak i dešifrovací exponent d jsou hodně velká čísla, a proces dešifrování*

$$X \equiv c^d \pmod{n}$$

trvá dlouho.

K urychlení dešifrovací transformace lze použít rozklad na výpočet s menšími moduly pomocí Čínské věty o zbytcích.

Pro $n = pq$ použijme již jednou provedený trik

$$\begin{aligned} X &\equiv X_p \pmod{p} \equiv c^{d_p} \pmod{p}, \\ X &\equiv X_q \pmod{q} \equiv c^{d_q} \pmod{q}, \end{aligned}$$

přičemž

$$\begin{aligned} d_p &\equiv d \pmod{\phi(p)} \equiv d \pmod{p-1} \Leftrightarrow d = d_p + j \cdot (p-1), \\ d_q &\equiv d \pmod{\phi(q)} \equiv d \pmod{q-1} \Leftrightarrow d = d_q + k \cdot (q-1), \end{aligned}$$

a tedy

$$\begin{aligned} c^d &\equiv c^{d_p + j(p-1)} \pmod{p} \equiv c^{d_p} 1^j \pmod{p} \equiv c^{d_p} \pmod{p}, \\ c^d &\equiv c^{d_q + k(q-1)} \pmod{q} \equiv c^{d_q} 1^k \pmod{q} \equiv c^{d_q} \pmod{q}. \end{aligned}$$

A	B	C	D	E	F	G	H	I	J	K	L	M
0	1	2	3	4	5	6	7	8	9	10	11	12
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
13	14	15	16	17	18	19	20	21	22	23	24	25

Tabulka 1: Konverzní tabulka pro převod znaků anglické abecedy do 26 zbytkových tříd

Zpráva X je tedy řešením soustavy dvou kongruencí sestavených pro c:

$$X \equiv c^{d_p} \pmod{p},$$
$$X \equiv c^{d_q} \pmod{q}.$$

Řešením je

$$X = \left[c^{d_p} M_p q + c^{d_q} M_q p\right] \bmod pq,$$

kde $M_p = q^{-1} \bmod p$ a $M_q = p^{-1} \bmod q$.

V tomto případě lze většinu hodnot předpočítat. Dešifrovací klíč je potom šestice $(p, q, d_p, d_q, M_p, M_q)$.

4.5 Prolomení RSA při nevhodné volbě p a q

Pokud zvolíme p a q nevhodně (blízko sebe, příliš malá, atd.), útočník využije znalosti (n, e):

1. Faktorizuje n na p a q.
2. Vypočte Eulerovu funkci pro n, $\phi(n) = (p-1)(q-1)$.
3. Dopočte **dešifrovací exponent** d tak, aby $d \cdot e \equiv 1 \pmod{\phi(n)}$.

Náš **soukromý klíč** pro dešifrování (n, d) v ten okamžik zná i útočník a může moje zprávy dešifrovat.

Příklad 9 (Příklad šifrování a dešifrování RSA). Chceme pomocí RSA zašifrovat a dešifrovat zprávu ABORT v anglické abecedě kódované podle Tabulky 1, přičemž pro generování RSA modulu použijeme $p = 43$ a $q = 31$.

Nejprve vypočteme RSA modul a jeho totient,

$$n = p \cdot q = 1333,$$
$$\phi(n) = \phi(1333) = 42 \cdot 30 = 1260.$$

Nyní musíme zvolit šifrovací exponent e tak, aby $1 < e < 1260$ a zároveň aby exponent nebyl soudělný s 1260 (připomeňme si, že v opačném případě by nebyl exponent invertovatelný). Tato volba je na nás, nejčastěji se setkáváme s nízkými exponenty e – pro náš příklad si zvolíme

$$e = 13,$$
$$d \equiv e^{-1} \pmod{\phi(n)} \equiv 13^{-1} \pmod{1260} \equiv 1153 \pmod{1260}.$$

Naši zprávu můžeme nyní například rozsekáme na bloky tak, aby celková bitová délka bloku byla kratší, než počet bitů v šifrovacím modulu n (ten má 11 bitů, naše reprezentace znaků anglické abecedy má 5 bitů na znak, do jednoho bloku tedy vložíme vedle sebe dva znaky).

text	A	B	O	R	T	
třída	0	1	14	17	19	
binárně	00000	00001	01110	10001	10011	
bloky	00000 00001		01110 10001		10011 00000	
číselně	1		465		608	

Další podobné příklady lze nalézt na různých interentových stránkách, například [3].

Reference

[1] Kleinjung, T. – Aoki, K. – Franke, J. – Lenstra, A. K. – Thomé, E. – Bos, J. W. – Gaudry, P. – Kruppa, A. – Montgomery, P. L. – Osvik, D. A. – te Riele, H. – Timofeev, A. – Zimmermann, P.: *Factorization of a 768-bit RSA modulus* [online]. Cryptology ePrint Archive, Report 2010/006. Dostupné na WWW: http://eprint.iacr.org/2010/006.pdf (staženo 16.10.2012).

[2] Bai, S. – Thomé, E. – Zimmermann, P.: *Factorisation of RSA-704 with CADO-NFS* [online]. Cryptology ePrint Archive, Report 2012/369. Dostupné na WWW: http://eprint.iacr.org/2012/369.pdf (staženo 16.10.2012).

[3] Wikipeda, the free encyclopedia: *RSA (algorithm) – A working example* [online]. Dostupné na WWW: http://en.wikipedia.org/wiki/RSA_(algorithm)#A_working_example (staženo 16.10.2012).