Algoritmus RSA

Vilém Vychodil

4. března 2002

Abstrakt

Následující podpůrný text stručně shrnuje základní problematiky při šifrování algoritmem RSA. Sem spadá nejen samotný princip algoritmu, ale i základní metody generování velkých prvočísel. Text po čtenáři nevyžaduje prakticky žádné znalosti, může být proto použit jako doplňkový materiál i v kursu Paradigmata programování I. Autor může být kontaktován prostřednictvím elektronické pošty na adrese <vilem.vychodil@upol.cz>.

Základní pojmy

V dalších úvahách vycházíme z jedné algebraická struktury – okruhu celých čísel $(\mathbb{Z}, +, \cdot)$. Zvolíme-li pevné číslo $m \in \mathbb{N}$, pak můžeme na množině $\mathbb{Z}$ definovat binární relaci θ_m předpisem

$$\theta_m = \{\langle a, b\rangle \,;\, a = b + t \cdot m, \text{ pro nějaké } t \in \mathbb{Z}\}\,. \tag{1}$$

Snadno lze ukázat, že relace θ_m je *ekvivalence* na $\mathbb{Z}$. Podrobně, θ_m je zcela jistě reflexivní, platí $a = a + 0 \cdot m$, odtud plyne $\langle a, a\rangle \in \theta_m$. Pokud $\langle a, b\rangle \in \theta_m$, pak lze psát $a = b + t \cdot m$ pro nějaké $t \in \mathbb{Z}$. Tento výraz lze upravit na $a - t \cdot m = b$. Odtud již z komutativity a z vlastností opačného prvku plyne $b = a + (-t) \cdot m$, to jest $\langle b, a\rangle \in \theta_m$ – relace je symetrická. Nyní stačí ověřit transitivitu. Uvažujme $\langle a, b\rangle \in \theta_m$, $\langle b, c\rangle \in \theta_m$. Existují tedy vyjádření $a = b + s \cdot m$, $b = c + t \cdot m$. Dosazením za b dostáváme $a = c + t \cdot m + s \cdot m$, to jest $a = c + (t + s) \cdot m$. Relace θ_m je vskutku ekvivalence na $\mathbb{Z}$.

Relace θ_m definovaná vztahem (1) je *kongruence* na $\mathbb{Z}$, to jest splňuje *substituční podmínky* vzhledem k operacím okruhu $(\mathbb{Z}, +, \cdot)$. To znamená, že pro libovolné $\langle a, b\rangle \in \theta_m$, $\langle c, d\rangle \in \theta_m$ platí $\langle a + c, b + d\rangle \in \theta_m$ a $\langle a \cdot c, b \cdot d\rangle \in \theta_m$. Platnost lze opět snadno ověřit, vyjdeme-li ze vztahů $a = b + t \cdot m$, $c = d + s \cdot m$, pak lze vyjádřit $a + c = b + d + (t + s) \cdot m$. Odtud plyne $\langle a + c, b + d\rangle \in \theta_m$. Dále $a \cdot c = b \cdot d + (b \cdot s + d \cdot t + t \cdot s \cdot m) \cdot m$, to jest i $\langle a \cdot c, b \cdot d\rangle \in \theta_m$.

Poznámka. V literatuře bývá kongruence θ_m nazývána obvykle *kongruence modulo* m. Fakt $\langle a, b\rangle \in \theta_m$ je obvykle značen $a \equiv b \pmod{m}$. Toto značení by však v dalším textu nebylo přehledné, proto jej nebudeme používat. Dále je dobré uvědomit si, co znamená předpis (1). Čísla a a b jsou kongruentní modulo m, právě když $m \mid (a - b)$, to plyne rovnou z definičního vztahu.

Vzhledem k tomu, že kongruence θ_m je relací ekvivalence, lze dle ní rozložit $\mathbb{Z}$ právě na m tříd rozkladu. Každá třída rozkladu representuje množinu čísel ze $\mathbb{Z}$, které dávají po vydělení číslem m stejný zbytek. Rozkladem okruhu $(\mathbb{Z}, +, \cdot)$ je vytvořen faktorový okruh $\mathbb{Z}/\theta_m$. Operace pro třídy rozkladu lze definovat přirozeně pomocí operací okruhu $(\mathbb{Z}, +, \cdot)$, díky substituční podmínce navíc nejsou výsledky operací ovlivněny výběrem prvku z třídy rozkladu.

Definice 1. Nechť $m \in \mathbb{N}$ a $\theta_m \subseteq \mathbb{Z} \times \mathbb{Z}$ označuje kongruenci definovanou vztahem (1). Označme $[a]_m = \{b;\, \langle a, b\rangle \in \theta_m\}$ třídu rozkladu do níž padne prvek $a \in \mathbb{Z}$. Dále nechť množina $\mathbb{Z}/\theta_m$ označuje systém všech tříd rozkladu dle θ_m, to jest $\mathbb{Z}/\theta_m = \{[a]_m\,;\, a \in \mathbb{Z}\}$. Na $\mathbb{Z}/\theta_m$ zavedeme operace následujícími předpisy,

$$[a]_m + [b]_m = [a+b]_m\,, \tag{2}$$
$$[a]_m \cdot [b]_m = [a \cdot b]_m\,. \tag{3}$$

Struktura $(\mathbb{Z}/\theta_m, +, \cdot)$ se nazývá *faktorový okruh modulo* m.

Poznámka. Nulou okruhu $(\mathbb{Z}/\theta_m, +, \cdot)$ je prvek $[0]_m$. Opačným prvkem k $[a]_m$ je $[m-a]_m$, neboť $[a]_m + [m-a]_m = [a+m-a]_m = [m]_m = [0]_m$. Jedničkou okruhu je $[1]_m$. V literatuře se může značení opět odlišovat. Okruh $(\mathbb{Z}/\theta_m, +, \cdot)$ je obvykle nazýván *okruh zbytkových tříd modulo* m a značen $(\mathbb{Z}_m, +, \cdot)$, jeho prvky bývají značeny C_0 až C_{m-1}. V našem případě toto značení koresponduje s třídami $[0]_m$ až $[m-1]_m$.

Definice 2. (Eulerova φ-funkce) Zobrazení $\varphi\colon \mathbb{N} \to \mathbb{N}$, které každé $n \in \mathbb{N}$ zobrazuje na počet menších s ním nesoudělných čísel, to jest

$$\varphi(n) = \mathrm{Card}\,\{k;\, k \in \mathbb{N},\, k < n,\, \mathrm{nsd}\,(k,n) = 1\}, \tag{4}$$

se nazývá *Eulerova φ-funkce.*

V některých případech lze hodnotu Eulerovy φ-funkce stanovit velmi rychle. Metoda šifrování RSA je založena právě na vhodných vlastnostech Eulerovy φ-funkce. Je-li n prvočíslo, nebo je ve tvaru součinu dvou prvočísel, pak lze hodnotu $\varphi(n)$ stanovit ihned.

Lemma 1. *Nechť $p, q \in \mathbb{N}$ jsou prvočísla. Pak platí $\varphi(p) = p-1$ a $\varphi(pq) = (p-1)(q-1)$.*

Důkaz. Je-li p prvočíslo, potom je každé číslo $k \in \mathbb{N}$, $k < n$ s číslem n nesoudělné. Těchto čísel je právě $p-1$. Nyní uvažujme číslo n ve tvaru $n = pq$, kde p, q jsou prvočísla. Všechna čísla soudělná s n jsou buďto ve tvaru $s \cdot p$, nebo $t \cdot q$. Soudělných čísel ve tvaru $s \cdot p$ menších jak n je ale právě $q-1$. Stejně tak soudělných čísel ve tvaru $t \cdot q$ menších jak n je právě $p-1$. Dohromady dostáváme $\varphi(pq) = (pq-1) - (p-1) - (q-1) = pq - p - q + 1 = (p-1)(q-1)$. □

Lemma 2. *Nechť $a \in \mathbb{Z}$, $n \in \mathbb{N}$ jsou nesoudělná čísla. Pak platí,*

(i) *Všechna čísla $p \in [a]_n$ jsou s n nesoudělná.*

(ii) *Je-li navíc $q \in \mathbb{Z}$ nesoudělné s n, pak je i číslo $q \cdot a$ nesoudělné s n.*

(iii) *Všechna čísla nesoudělná s n tvoří právě $\varphi(n)$ tříd rozkladu podle θ_n.*

Důkaz. (i) Uvažujme třídu rozkladu $[a]_n$ a libovolné číslo $p \in [a]_n$. Platí $p = a + tn$. Předpokládejme, že $m \mid n$, $m \mid p$. Z předpokladu platí $m \mid (a + tn)$. Jelikož $m \mid n$, platí i $m \mid tn$. Rovněž platí i $m \mid (p - tn)$. Podrobněji, z $tn = m \cdot s_1$, $p = m \cdot s_2$ plyne $p - tn = m(s_2 - s_1)$, to jest $m \mid (p - tn)$. Z předchozích zjištění dále plyne $m \mid a$. Odtud dostáváme $m = 1$.

(ii) Uvažujme dvě čísla $q, a \in \mathbb{Z}$ nesoudělná s n. To jest platí $\mathrm{nsd}\,(q,n) = 1$, $\mathrm{nsd}\,(a,n) = 1$. Chceme ukázat, že i $\mathrm{nsd}\,(a \cdot q, n) = 1$. K úplně přesnému důkazu by bylo třeba definovat pojem asociované prvky, my se však pro zjednodušení omezíme pouze na přirozené největší společné dělitele. Bez újmy lze vyjádřit $\mathrm{nsd}\,(a \cdot q, n) = \mathrm{nsd}\,(a \cdot q, \mathrm{nsd}\,(n, n \cdot q))$, z asociativity

dále plyne $\operatorname{nsd}(a \cdot q, n) = \operatorname{nsd}(\operatorname{nsd}(a \cdot q, n \cdot q), n)$. Platí $\operatorname{nsd}(a \cdot q, n \cdot q) = q \cdot \operatorname{nsd}(a, n)$. Odtud dostáváme $\operatorname{nsd}(a \cdot q, n) = \operatorname{nsd}(q \cdot \operatorname{nsd}(a, n), n) = \operatorname{nsd}(q, n) = 1$, což bylo dokázat. Důkaz bodu (ii) by správně potřebovat detailnější rozbor vlastností dělitelnosti v oboru integrity, to ale není cílem tohoto textu.

(iii) Existuje právě $\varphi(n)$ vzájemně různých přirozených čísel ostře menších než n a zároveň nesoudělných s n. Označme tato čísla $a_1, \ldots, a_{\varphi(n)}$. Všechny prvky tříd $[a_1]_n, \ldots, \left[a_{\varphi(n)}\right]_n$ jsou čísla s n nesoudělná, to jest $\cup\{[a_i]_n\,;\, 1 \leq i \leq \varphi(n)\}$ je množinou všech čísel nesoudělných s n. Navíc pro dvě třídy rozkladu $[a_i]_n = [a_j]_n$ lze $p \in [a_i]_n$ vyjádřit jako $p = a_i + tn = a_j + sn$. Odtud dostáváme $a_i - a_j = (s-t)n$. Jinak řečeno $n \mid a_i - a_j$, ale $a_i, a_j \leq n-1$, tedy platí $a_i = a_j$. Třídy rozkladu jsou vzájemně různé a je jich právě $\varphi(n)$. □

Věta 1. (Fermatova-Eulerova) *Nechť $q \in \mathbb{Z}$, $n \in \mathbb{N}$ jsou nesoudělná čísla. Pak platí*

$$\left[q^{\varphi(n)}\right]_n = [1]_n\,. \tag{5}$$

Důkaz. Označme $a_1, \ldots, a_{\varphi(n)}$ všechna různá čísla nesoudělná s n, pro která platí $a_i < n$ pro libovolné $i = 1, \ldots, \varphi(n)$. Podle předchozí lemmy jsou $[a_1]_n, \ldots, \left[a_{\varphi(n)}\right]_n$ právě všechny třídy čísel nesoudělných s n. Je-li q číslo nesoudělné s n, pak jsou rovněž i $qa_1, \ldots, qa_{\varphi(n)}$ čísla nesoudělná s n, to opět plyne z předcházející lemmy.

Třídy $[qa_i]_n$, $[qa_j]_n$ jsou pro libovolná $1 \leq i, j \leq \varphi(n)$, $i \neq j$ vzájemně různé. Předpokládáme-li $[qa_i]_n = [qa_j]_n$, pak lze libovolný $p \in [qa_i]_n$ vyjádřit ve tvarech $p = qa_i + sn$ a $p = qa_j + tn$. To jest platí $qa_i + sn = qa_j + tn$, odtud $q(a_i - a_j) = (t-s)n$. Odtud dostáváme, že $n \mid (a_i - a_j)$, protože q je s n nesoudělné. Ale čísla $a_i, a_j \leq n-1$, to jest $a_i = a_j$.

Z předchozího faktu plyne, že třídy $[a_1]_n, \ldots, \left[a_{\varphi(n)}\right]_n$ a $[qa_1]_n, \ldots, \left[qa_{\varphi(n)}\right]_n$ jsou až na pořadí totožné. Označme $[a]_n = [a_1]_n \cdot [a_2]_n \cdots \left[a_{\varphi(n)}\right]_n = \left[a_1 \cdot a_2 \cdots a_{\varphi(n)}\right]_n$. Pak platí,

$$[a]_n = [a_1]_n \cdot [a_2]_n \cdots \left[a_{\varphi(n)}\right]_n = [qa_1]_n \cdot [qa_2]_n \cdots \left[qa_{\varphi(n)}\right]_n = \left[q^{\varphi(n)}a\right]_n = \left[q^{\varphi(n)}\right]_n \cdot [a]_n\,.$$

Jelikož je a nesoudělné s n, platí $[a]_n \neq [0]_n$. Tím pádem lze předchozí vztah vykrátit $[a]_n$ a dostáváme požadované tvrzení $\left[q^{\varphi(n)}\right]_n = [1]_n$. □

Z předcházejícího tvrzení mimo jiné plyne, že pro nesoudělná čísla $q \in \mathbb{Z}$, $n \in \mathbb{N}$ dělí n číslo $q^{\varphi(n)} - 1$ beze zbytku, to jest $n \mid q^{\varphi(n)} - 1$. Další části textu se věnují využití vlastnosti Eulerovy φ-funkce při asymetrickém šifrování metodou RSA.

Princip algoritmu RSA

Algoritmus RSA je asi nejznámějším representantem asymetrických metod šifrování. V roce 1977 jej navrhli Ron Rivest, Adi Shamir a Leonard Adelman, algoritmus nese jméno právě podle nich. Asymetrická kryptografie je založena na dvou od sebe různých klíčích. Jeden z klíčů je používán výhradně k šifrování a nelze jej použít k dešifrování. Druhý klíč je využíván výlučně pro dešifrování a naopak jej nelze použít k šifrování. V praxi je pouze jeden z dvojice klíčů veřejně dostupný, druhý je utajen. Aby byla asymetrická kryptografie skutečně robustní, musí být zaručena praktická nemožnost výpočtu jednoho klíče z druhého. Asymetrické šifrovací algoritmy jsou aplikací teorie složitosti, jedné ze stěžejních disciplín informatiky. Dvojice klíčů je volena tak, aby výpočet jednoho klíče z druhého byl sice algoritmicky řešitelný, ale neúnosný z hlediska výpočetního výkonu soudobých i budoucích počítačů.

Celý algoritmus je zhruba následující. Nejprve jsou zvolena dvě prvočísla p, q, jejich součin označme $n = pq$. Jelikož je n ve tvaru součinu prvočísel platí $\varphi(n) = (p-1)(q-1)$. Dále zvolíme čísla e, d aby platilo $[e \cdot d]_{\varphi(n)} = [1]_{\varphi(n)}$. Dvojice čísel $\langle e, n\rangle$, $\langle d, n\rangle$ tvoří veřejný a soukromý klíč. Označme zdrojovou zprávu v a zakódovanou zprávu w. Pokud chce odesílatel zakódovat zdrojovou zprávu v, vychází ze vztahu $[w]_n = [v^e]_n$. Příjemce zprávy obdrží w a rozkóduje jej pomocí vztahu $[v]_n = \left[w^d\right]_n$.

Při použití algoritmu je nutné vyřešit několik zásadních problémů. Nejprve je nutné dokázat správnost metody uvedené v předchozím odstavci. Navíc je potřeba z vhodně zvolených e a n pomocí vztahu $[e \cdot d]_{\varphi(n)} = [1]_{\varphi(n)}$ vypočítat číslo d. Dále je potřeba mít k disposici jednoduchý algoritmus generování velkých prvočísel s přijatelnou časovou složitostí. Dalším problémem je efektivní implementace mocnění modulo m, které se používá při šifrování a dešifrování. V neposlední řadě hraje roli i volba kódování zprávy. Slova v, w uvedená v předchozím odstavci representují čísla $0 \leq v, w \leq n-1$. Přenášené zprávy však mívají většinou charakter „sekvence znaků“. Na jednotlivé problémy se pokusíme odpovědět v dalším textu.

Následující tvrzení ukazuje správnost algoritmu.

Věta 2. (Správnost algoritmu RSA) *Nechť p, q jsou prvočísla, $n = pq$. A nechť e, d jsou libovolná čísla splňující podmínku $[ed]_{\varphi(n)} = [1]_{\varphi(n)}$. Dále označme v, w čísla $0 \leq v \leq n-1$ a nechť platí $[w]_n = [v^e]_n$. Potom platí $[v]_n = \left[w^d\right]_n$.*

Důkaz. V důkazu správnosti algoritmu vycházíme z $[ed]_{\varphi(n)} = [1]_{\varphi(n)}$ a $[w]_n = [v^e]_n$. Dokazujeme platnost $[v]_n = \left[w^d\right]_n$. Ze vztahu $[ed]_{\varphi(n)} = [1]_{\varphi(n)}$ plyne, že $\langle ed, 1\rangle \in \theta_{\varphi(n)}$. Jinak řečeno součin ed dává po dělení $\varphi(n)$ zbytek jedna, to jest platí $ed = 1 + r\varphi(n)$. Pokud dále umocníme vztah $[w]_n = [v^e]_n$ do tvaru $\left[w^d\right]_n = \left[v^{ed}\right]_n$, lze místo dokazovaného vztahu $[v]_n = \left[w^d\right]_n$ psát $[v]_n = \left[v^{ed}\right]_n$. Pro důkaz správnosti algoritmu tedy stačí ověřit platnost

$$[v]_n = \left[v^{1+r\varphi(n)}\right]_n . \tag{6}$$

Důkaz je dále veden rozborem případů. Konkrétně uvažujeme číslo v buďto soudělné s n, nebo nesoudělné. V případě, že v je nesoudělné s n lze psát $\left[v^{1+r\varphi(n)}\right]_n = [v]_n \cdot \left[v^{r\varphi(n)}\right]_n$, to jest stačí ověřit $\left[v^{r\varphi(n)}\right]_n = [1]_n$. Dále z Fermatovy-Eulerovy věty plyne $\left[v^{\varphi(n)}\right]_n = [1]_n$. Umocněním tohoto vztahu a z vlastností kongruence dostáváme

$$\left[v^{r\varphi(n)}\right]_n = \left[v^{\varphi(n)}\right]_n^r = [1]_n^r = [1^r]_n = [1]_n . \tag{7}$$

V případě, že že v je soudělné s $n = pq$ musí být v buďto ve tvaru $v = ap$, nebo ve tvaru $v = bq$. Bez újmy lze předpokládat, že $v = ap$. V tomto případě je ale v nesoudělné s q, jelikož q je prvočíslo. V okruhu $(Z/\theta_q, +, \cdot)$ můžeme vyjádřit

$$\left[v^{r\varphi(n)}\right]_q = \left[v^{\varphi(n)}\right]_q^r = \left[v^{(p-1)(q-1)}\right]_q^r = \left[v^{(q-1)}\right]_q^{(p-1)r} = [1]_q . \tag{8}$$

Poslední rovnost plyne z faktu, že q je prvočíslo a proto $\varphi(q) = q-1$. Navíc podle Fermatovy-Eulerovy věty je $\left[v^{(q-1)}\right]_q = \left[v^{\varphi(q)}\right]_q = [1]_q$. Jelikož platí i $\left[v^{r\varphi(q)}\right]_q = [1]_q$, lze tento fakt vyjádřit ve tvaru $v^{r\varphi(n)} = 1^r + tq$, po vynásobení této rovnosti v obdržíme $v^{1+r\varphi(n)} = v + vtq$, to jest $v^{1+r\varphi(n)} = v + atpq$. Zároveň platí $n = pq$, tedy $v^{1+r\varphi(n)} = v + (at)n$, jinými slovy

$$[v]_n = \left[v^{1+r\varphi(n)}\right]_n ,$$

což bylo dokázat. □

Síla algoritmu RSA je patrná právě z tohoto důkazu. Odesílatel pošty nemůže ze znalosti v, w, e a n jednoduše stanovit hodnotu d. K tomu by musel vyřešit kongruenci $[ed]_{\varphi(n)} = [1]_{\varphi(n)}$. Pro správné řešení je nejprve nutné znát hodnotu $\varphi(n)$. Pro velké číslo n nelze hodnotu $\varphi(n)$ v dohledné době vypočítat. Rovněž prvočíselný rozklad velkého čísla n trvá neúnosnou dobu. Dosud není znám efektivní algoritmus rozkladu na prvočinitele.

Příklad 1. Zvolme například prvočísla $p = 5437$, $q = 7331$. Dále stanovíme,

$$\begin{aligned} n = pq &= 39858647, & \varphi(n) = (p-1)(q-1) &= 39845880, \\ e &= 25634761, & d &= 37458481. \end{aligned}$$

Dvojice $\langle e, n\rangle$, $\langle d, n\rangle$ tvoří veřejný a tajný klíč. Nyní můžeme zakódovat například $v = 1234$ pomocí vztahu $[w]_n = [v^e]_n$, dostáváme $w = 14807834$. Naopak při rozkódování vycházíme ze vztahu $[v]_n = \left[w^d\right]_n$, kde $w = 14807834$ a obdržíme $v = 1234$. □

Poznámka. Na předchozím příkladu jsou dobře vidět dvě úskalí. V první řadě není jasné, jak byla čísla e, d stanovena. Tato čísla vskutku splňují podmínku $[e \cdot d]_{\varphi(n)} = [1]_{\varphi(n)}$. V případě malých čísel lze jejich volbu provést ad hoc. V případě velkých čísel je nutné vypočítat jeden klíč pomocí druhého a čísla $\varphi(n)$. To jest na počátku je vhodně zvolen například pouze klíč e a pomocí hodnoty $\varphi(n)$ je vypočten příslušný klíč d.

Dalším, spíš technickým problémem, je samotný výpočet w z v a obráceně. Na první pohled se nabízí umocnit například v^e a zjistit zbytek po dělení číslem n. To by v předchozím případě ale znamenalo vypočítat nejprve číslo o $\log_{10}(v^e) = e\log_{10}(v) \doteq 79245126$ cifrách, což je naprosto neúnosné. Navíc tento fakt je ještě umocněn „velmi neprozřetelnou“ volbou prvočísel p, q. Čísla byla z demonstračních důvodů zvolena velmi malá – zlomení klíče by v jejich případě nebylo příliš časově náročné.

Věta 3. *Nechť p, q jsou prvočísla, $n = pq$. Dále zvolme prvočíslo $2 \le e \le n-1$ nesoudělné s číslem $\varphi(n)$. Předpokládejme, že pro čísla e, d platí $[ed]_{\varphi(n)} = [1]_{\varphi(n)}$. Potom d je ve tvaru*

$$d = \frac{1 + r\varphi(n)}{e}, \quad \text{kde} \quad [r]_e = \left[(e-1)\varphi(n)^{e-2}\right]_e . \tag{9}$$

Důkaz. Poznamenejme, že $[ed]_{\varphi(n)} = [1]_{\varphi(n)}$ platí právě když platí $ed = 1 + r\varphi(n)$. To jest fakticky stačí ověřit, zdali je číslo r voleno tak, aby platilo $e \mid 1 + r\varphi(n)$. Pak lze stanovit d a platnost vztahů je tím dokázána.

Pro číslo e platí $e \mid 1 + r\varphi(n)$ právě když $\langle 1 + r\varphi(n), 0\rangle \in \theta_e$, to jest právě když zbytek po dělení výrazu $1 + r\varphi(n)$ číslem e je nula. Tuto podmínku lze ekvivalentně vyjádřit ve tvaru $[1 + r\varphi(n)]_e = [0]_e$, nebo jinak $[1]_e + [r]_e \cdot [\varphi(n)]_e = [0]_e$, dále jako $[r]_e \cdot [\varphi(n)]_e = -[1]_e$. Opačný prvek k $[1]_e$ má tvar $[e-1]_e$. Pomocí inverse lze vyjádřit $[r]_e = [e-1]_e \cdot [\varphi(n)]_e^{-1}$. Vztah (9) je tedy platný právě když je $[\varphi(n)]_e^{-1}$ roven $\left[\varphi(n)^{e-2}\right]_e$. Platí,

$$[\varphi(n)]_e \cdot [\varphi(n)]_e^{-1} = [\varphi(n)]_e \cdot \left[\varphi(n)^{e-2}\right]_e = \left[\varphi(n)^{e-1}\right]_e = \left[\varphi(n)^{\varphi(e)}\right]_e = [1]_e . \tag{10}$$

Poslední vztah plyne z předpokladu nesoudělnosti $\varphi(n)$ s e a užitím Fermatovy-Eulerovy věty. Samozřejmě platí $\varphi(e) = e - 1$, protože e bylo voleno jako prvočíslo. Tím je ale důkaz hotov. Volba čísla r implikuje platnost $e \mid 1 + r\varphi(n)$ a podmínka $ed = 1 + r\varphi(n)$ je splněna. □

Příklad 2. Předchozí tvrzení dává návod, jak jednoduše stanovit klíče. Uvažujme hodnoty p, q z minulého příkladu. Dále předpokládejme, že jsme zvolili číslo e. Číslo e je v našem případě prvočíslo a evidentně nedělí $\varphi(n)$, to jest má požadované vlastnosti. Ze vztahů (9) vypočítáme pomocné číslo $r = 24098833$, a potom i $d = 37458481$. □

Stanovením klíčů se opět otevírá problém efektivního výpočtu zbytku po dělení umocněného čísla. Tímto problémem se budeme zabývat nyní. Označme pro přehlednost $\mathrm{Res}\,(a, m)$ zbytek po dělení čísla a číslem m, to jest pro $a = bm + r$ je $\mathrm{Res}\,(a, m) = r$. Pro mocnění přirozených čísel a, n je znám algoritmus s logaritmickou časovou složitostí. Platí totiž,

$$a^n = \begin{cases} a^{\frac{n}{2}} \cdot a^{\frac{n}{2}} & \text{pro } n \text{ sudé, } n > 1, \\ a \cdot a^{n-1} & \text{pro } n \text{ liché, } n > 1, \\ a & \text{pro } n = 1. \end{cases} \tag{11}$$

Pokud bychom vyšli z tohoto algoritmu a počítali $\mathrm{Res}\,(a^n, m)$, při mocnění velkých čísel velkými čísly bychom se záhy dostali do potíží. Například výsledkem umocnění dvou stociferných čísel je víc jak (10^{100})-ciferné číslo. Výpočet takové mocniny je hluboko za hranicemi možností jakéhokoliv byť i hypotetického výpočetního systému. Díky vlastnostem kongruence však můžeme provádět operaci $\mathrm{Res}\,(a, m)$ v každém kroku výpočtu. Intuitivně řečeno, při výpočtu je využito vlastnosti $[a^n]_m = [a]_m^n$ a v paměti počítače jsou udržována čísla se stále stejným počtem cifer. Následující kód v jazyku Scheme demonstruje implementaci funkce `expmod` využívající tohoto principu.

```
(define (expmod x n m)
  (define (even? n) (= (modulo n 2) 0))
  (define (sqr x) (* x x))
  (cond ((= n 0) 1)
        ((even? n) (modulo (sqr (expmod x (/ n 2) m)) m))
        (else (modulo (* x (expmod x (- n 1) m)) m))))
```

Výše uvedenou procedurou lze efektivně kódovat, dekódovat i vypočítat klíče. Při výpočtu klíče je opět nutné využít funkci `expmod` při stanovení pomocné hodnoty r. Předpokládejme, že máme definovány hodnoty pro p, q, n a stanovenu hodnoty $\varphi(n)$ a rovněž klíč e. Klíč d lze efektivně vypočítat následujícím kódem.

```
(define r (modulo (* (- e 1) (expmod phi-n (- e 2) e)) e))
(define d (/ (+ 1 (* r phi-n)) e))
(= (expmod (expmod 1234 e n) d n) 1234)
```

Symbol `phi-n` representuje hodnotu $\varphi(n) = (p-1)(q-1)$. Výsledkem vyhodnocení výrazu na třetím řádku je `#t` značící pravdu, to jest zkušební zakódované slovo 1234 bylo správně dekódováno.

Poznámka. Obrácenou technikou k šifrování s veřejným klíčem je elektronický podpis. V podstatě se jedná o modifikaci problému. Pouze jeden účastník může zprávu zašifrovat, ostatní ji mohou pouze rozšifrovat – tím se přesvědčí o „pravosti zprávy“. Vzhledem k tomu, že dvojice klíčů $\langle e, n\rangle$, $\langle d, n\rangle$ je komplementární, lze ji vzájemně použít nejen k šifrování ve smyslu utajení dat, ale právě i k elektronickému podpisu – šifrování ve smyslu ověření pravosti. Navíc je použit týž šifrovací a dešifrovací algoritmus, pouze se zaměněnými klíči.

Probabilistické testování prvočíselnosti

Při implementaci metody je potřeba vhodně volit počáteční prvočísla p, q. Složitost zlomení klíče je úměrná jejich velikosti. Otázkou je, jak efektivně generovat velká prvočísla s řádově stovkami cifer. Jelikož je exaktní testování prvočíselnosti pro velká čísla neúnosné, v aplikacích často stačí mít k disposici číslo, které je téměř jistě prvočíslem. Zcela dostačující pravděpodobnost prvočíselnosti je například $1 - 10^{-20}$. Mezi základní probabilistické metody testování prvočíselnosti patří metoda založená na platnosti Fermatovy-Eulerovy[1] věty.

Je-li n prvočíslo, pak pro každé a nesoudělné s n platí $\left[a^{n-1}\right]_n = [1]_n$. Čísla soudělná s n mohou být pouze ve tvaru $t \cdot n$, pro $t \in \mathbb{Z}$. Odtud dostáváme, že pro $1 \leq a \leq n-1$ je za předpokladu prvočíselnosti n podmínka $\left[a^{n-1}\right]_n = [1]_n$ vždy splněna. Pokud není podmínka $\left[a^{n-1}\right]_n = [1]_n$ pro nějaké $1 \leq a \leq n-1$ splněna, číslo n nemůže být prvočíslo. U velkých čísel není možné testovat všechna čísla splňující $1 \leq a \leq n-1$. Test je prováděn jen konstantně mnoha pokusy při nichž je volena hodnota a náhodně. Pokud je pokusů dostatečně mnoho a podmínka je při nich splněna, je vysoce pravděpodobné, že n je prvočíslo. Existují však čísla, pro které heuristický test založený na předchozích úvahách částečně selhává.

Definice 3. (Carmichaelovo číslo) Nechť m je složené číslo a nechť pro libovolné $a \in \mathbb{Z}$ nesoudělné s m platí $\left[a^{m-1}\right]_m = [1]_m$. Složené číslo m se nazývá *Carmichaelovo číslo*.

To jest pokud je provedeno k náhodných voleb čísla a a ani v jednom případě nedělí Carmichaelovo číslo m, potom heuristický test prohlásí číslo m za „prvočíslo“, což není pravda. Nejmenším Carmichaelovým číslem je 561, dále 1105, 1729, ... Pouze šestnáct Carmichaelových čísel je menších než 10^5 a pouze 2163 je menších než $25 \cdot 10^9$. To je vzhledem k počtu 10^7 prvočísel menších než $25 \cdot 10^9$ zanedbatelné množství. Ačkoliv jsou Carmichaelova čísla mezi ostatními přirozenými čísly rozložena velmi řídce, v roce 1994 bylo dokázáno, že je jich nekonečně mnoho, viz [1]. Naštěstí při testování dostatečně velkých kandidátů na prvočíslo lze „zásah“ Carmichaelova čísla prakticky vyloučit.

Poznámka. Algoritmus generování velkých prvočísel je přímočarý. Nejprve je vygenerováno velké číslo. V dalším kroku je číslo zpracováno heuristickým testem. Pokud test selže, je vygenerováno nové velké číslo a testovací procedura se opakuje. Generování nových čísel probíhá dokud není nalezeno číslo, pro které je heuristický test splněn. Ve většině programovacích jazyků nelze generovat velká čísla přímo a je nutné vytvářet je po jednotlivých cifrách.

Následující funkce generuje velké číslo po cifrách.

```
(define (select-number digits)
  (define (iter i accum)
    (if (>= i (- digits 1))
        accum
        (iter (+ i 1) (+ (* accum 10) (random 10)))))
  (iter 0 (+ (random 9) 1)))
```

Funkce `select-number` vrací náhodně vygenerované číslo o zvoleném počtu cifer. Při generování je nutné vhodně volit hodnotu první cifry. První cifra nemůže být nulová, jinak by nebyl zachován počet požadovaných platných cifer. Pro testování prvočíselnosti a generování velkých prvočísel lze vytvořit další dvě funkce.

[1] V literatuře je možné setkat se s označením „Malá Fermatova věta“. Pierre de Fermat ji vyslovil poprvé v roce 1640, avšak ke tvrzení nepodal důkaz. Důkaz byl sestaven až Leonhardem Eulerem v roce 1736.

```
(define (fast-prime? n times digits)
  (define (fermat-test n)
    (define a (+ 2 (select-number (+ (random digits) 1))))
    (= (expmod a (- n 1) n) 1))
  (cond ((= times 0) #t)
        ((fermat-test n) (fast-prime? n (- times 1) digits))
        (else #f)))

(define (select-prime digits)
  (define (iter candidate)
    (if (fast-prime? candidate 50 (- digits 2))
        candidate
        (iter (select-number digits))))
  (iter (select-number digits)))
```

Predikát `fast-prime?` je pravdivý právě když vstupní číslo ◁n▷ o ◁$digit$▷ cifrách splňuje podmínku heuristického testu. Počet opakování testu je dán argumentem ◁$times$▷. Funkce `select-prime` postupně generuje velká čísla a požadovaném počtu cifer dokud není nalezeno číslo splňující podmínku heuristického testu.

Kódování vstupní abecedy

Až doposud byly kódované zprávy uvažovány jako čísla. V praxi se však kódují především zprávy složené ze znaků. Jednotlivé znaky jsou v počítači representovány rovněž čísly zpravidla s osmibitovým rozsahem, to jest čísly $0, \ldots, 255$. Algoritmus RSA šifruje čísla s rozsahem $0 \leq v \leq n-1$. Obvykle bývá n výrazně větším číslem než 255, nebylo by tedy úsporné kódovat vstupní zprávu „po znacích“. Z hlediska bezpečnostního je to rovněž nevyhovující – obsah zprávy by bylo možné dekódovat pomocí frekvenční analýzy.

Jedním z možných řešení je kódovat vždy skupinu znaků. Zdrojová zpráva je nejprve rozdělena do bloků o konstantní délce l. Každý blok je samostatně zakódován. Uvažujeme ji znaky kódované osmi bity, můžeme jednotlivé znaky chápat jako cifry čísla zapsaného v číselné soustavě o základu 256. Problém kódování bloku je tedy problémem převedení l-ciferného čísla ze soustavy o základu 256 do dekadické soustavy. Číslo v tomto tvaru již není problém zašifrovat pomocí algoritmu RSA. Po dešifrování je potřeba provést zpětné dekódování, to jest převod do soustavy o základu 256 a převod na znaky.

Otázkou zůstává, jak volit délku bloku l. Délka bloku musí respektovat velikost čísla $n-1$. Pokud by některé l-ciferné číslo zapsané v soustavě o základu 256 mělo dekadické vyjádření větší jako $n-1$, dekódování zprávy po rozšifrování by nebylo jednoznačné. Maximální možná délka je celočíselná hodnota $l = \log_{256}(n-1)$. Například pro $n = 39858647$ je $\log_{256}(n-1) = 3.612$, to jest bezpečná velikost bloku je $l = 3$. Následující funkce provádějí kódování a dekódování vstupní a výstupní zprávy.

```
(define (encode-block block base)
  (define (iter block aux)
    (if (null? block)
        aux
        (iter (cdr block) (+ (* base aux) (car block)))))
  (iter (map char->integer (string->list block)) 0))
```

```
(define (decode-block number base)
  (define (decode number)
    (if (< number base)
        (list number)
        (append (decode (quotient number base))
                (list (modulo number base)))))
  (list->string (map integer->char (decode number))))
```

Argumenty ◁*block*▷, ◁*number*▷ representují vstupní řetězec znaků a vstupní číslo. Argument ◁*base*▷ je číselný základ. V drtivé většině případů by měl být roven 256. Pokud by ale uživatel chtěl kódovat například pouze sedmibitové texty, může použít hodnotu 128. Je-li ku příkladu zakódován řetězec „Ahoj“ se základem 128, jeho kódem je číslo $a = 134836514$, to jest ekvivalentní zápis v desítkové soustavě. Naopak převedením čísla a zpět do soustavy o základu 256 a převodem čísel na znaky obdržíme řetězec „Ahoj“.

Poznámka. Zvolené řešení je spíš demonstrativní. Při efektivní implementaci se používají odlišné metody kódování. Především šifrovat lze přímo ve dvojkové soustavě, a klíče mají zpravidla délku dělitelnou osmi – kódování se tak výrazně zjednodušuje. Před samotným šifrováním jsou obvykle data komprimována nějakým slovníkovým kompresním algoritmem, například algoritmem LZW.

Reference

[1] Alford, W. R. — Granville, A. — Pomerance, C. *There are infinitely many Carmichael numbers.* Ann. Math., 140 (1994) 703-722.

[2] Granville, A. — Pomerance, C. *Two contradictory conjectures concerning Carmichael numbers.* Math. Comp., (2001) to appear in print.

[3] Pinch, R. *The Carmichael numbers up to 1015.* Math. Comp., 61:203 (1993) 381-391.

[4] Rivest, R. L. — Shamir, A. — Adelman, L. *A method for obtaining digital signatures and public-key cryptosystems.* Communications of the ACM, 21(2):120-126, February 1978.