Public-key matematika

Invertování čísla v okruhu

dělení čísel, případně výpočet inverzní hodnoty v rámci okruhu (tělesa) – nelze samozřejmě pro tzv. dělitele nuly
používá se rozšířený Euclidův algoritmus (viz. níže), uvažte, že $ax \equiv 1 \pmod{d}$

Rozšířený Euclidův algoritmus

vychází z Bezoutovy rovnosti: **gcd (*a*, *b*) =** $\alpha a + \beta b$ *a*, *b* přirozená, α,β celá
vstup: nezáporná čísla *a* a *b*, a ≥
výstup: *d* = **gcd**(*a*, *b*) a celá čísla *x*, *y* tž. $ax + by = d$

if (*b* = 0) do
d←*a*, *x*←1, *y*←0, return (*d*, *x*, *y*)
enddo
$x_2 \leftarrow 1$, $x_1 \leftarrow 0$, $y_2 \leftarrow 0$, $y_1 \leftarrow 1$
while *b* > 0 do
$q \leftarrow \lfloor a/b \rfloor$, $r \leftarrow a - qb$, $x \leftarrow x_2 - q\,x_1$, $y \leftarrow y_2 - q\,y_1$
$a \leftarrow b$, $b \leftarrow r$, $x_2 \leftarrow x_1$, $x_1 \leftarrow x$, $y_2 \leftarrow y_1$, $y_1 \leftarrow y$
enddo
$d \leftarrow a$, $x \leftarrow x_2$, $y \leftarrow y_2$
return (*d*, *x*, *y*)

takže položíme $b = \varphi(n)$, $x = e$ kde *e* nesoudělné s $\varphi(n)$ a počítáme

Volba prvočísel

Vygenerujeme náhodné liché číslo zvoleného řádu
Otestujeme prvočíselnost
Není-li prvočíslo, pokračujeme bodem 1.

Testy prvočíselnosti

Pro každé liché přirozené číslo *n* definujeme množinu $W(n) \subset \mathbf{Z}_n$:
pro $a \in \mathbf{Z}_n$ lze v polynomiálním čase ověřit, zda $a \in W(n)$
pokud je *n* prvočíslo, $W(n) = \theta$
pokud je *n* složené, $|W(n)| >= n/2$

Prvky množiny $W(n)$ nazýváme *svědky* toho, že číslo n je složené, ostatním číslům v $\mathbf{Z}_n$ - $W(n)$ říkáme *lháři*.

Solovay-Strassenův test prvočíselnosti

Základem je tzv. Eulerovo kritérium:
Pro liché prvočíslo p platí $r^{(p-1)/2} \equiv J(r,p) (\mathrm{mod}\, n)$ pro všechna celá čísla r splňující **nsd**(p, r) = 1

Ze skutečnosti, že pro p liché existuje maximálně $\phi(p)/2$ lhářů pro čísla 1, 2, ... p-1 lze odvodit následující algoritmus:

p číslo, které zkoumáme, r libv. číslo, pak nutně

nsd(p, r) = 1

a zároveň

$$\mathbf{J}(r,p) \equiv r^{p-1/2} \bmod p$$

kde **J**(r, p) je Jacobiho funkce, definovaná následovně

$$\mathbf{J}(r,p) \equiv \begin{cases} 1 & \text{pro } r = 1 \\ \mathbf{J}(r/2,p) * (-1)^{(p^2-1)/8} & \text{pro } r \text{ sudé} \\ \mathbf{J}(p \bmod r, r) * (-1)^{(r-1)(p-1)/4} & \text{pro } r \text{ liché }, r \neq 1 \end{cases}$$

1. zvolíme náhodně r tž. $2 \le r \le p-2$
2. spočítáme $n = r^{(p-1)/2} \bmod (p)$
3. pokud $n \neq 1$ a $n \neq p-1$ konec, je složené
4. spočítáme s = **J**(r, p), pokud není $n \equiv s$ konec, je složené
5. asi prvočíslo

Opakováním testu pro různé hodnoty r lze docílit požadované jistoty, že p je skutečně prvočíslo.

Miller-Rabinův test prvočíselnosti

Založen na následující skutečnosti:
Je-li p prvočíslo, potom jediná dvě řešení vztahu $x^2 \equiv 1 (mod\, p)$ jsou 1 a -1.

Pro liché přirozené číslo p tž. $p - 1 = 2^l s$, kde s je liché buď a celé číslo takové, že **nsd**$(a,p) = 1$. Potom $a^s \equiv 1 (\mathrm{mod}\, p)$, nebo $a^{2^j s} \equiv -1 (\mathrm{mod}\, p)$ pro nějaké j tž. $0 \le j \le l-1$ … pokud obě kongruence neplatí, je a silný svědek.
Lhářů je max $\frac{\varphi(p)}{4}$, tzn. test mnohem rychleji konverguje.

Odtud odvodíme následující algoritmus:
Dáno testované číslo p.
Nechť $p - 1 = 2^l s$, pro nějaké liché s
Náhodně zvolíme $a \in \{ 2, \dots , p - 2\}$
Spočítáme $q \equiv a^s \bmod p$. Pokud $q \equiv 1 \bmod p$ konec, asi prvočíslo.
Počítáme q^2, q^{2^2}, … , $q^{2^l} = a^{p-1}$, vše **mod** p. Pokud není $a^{p-1} \equiv 1 \bmod p$, konec - nemůže být prvočíslo.
Nalezeme největší k tak, že $q^{2^k} \not\equiv 1 \bmod n$. Pokud $q^{2^k} \equiv -1 \bmod p$, konec - asi prvočíslo, jinak nemůže být prvočíslo.

Výtah z teorie kódů

u a v jsou kódová slova
Hammingova vzdálenost $\delta(u,v) = \left|\{ i \vee 1 \le i \le n ; u_i \ne v_i \}\right|$
minimální vzdálenost kódu $d(C) = min\{ \delta(u,v) \vee u,v \in C \wedge u \ne v \}$
u' obsahuje t chyb pokud $\delta(u,u') = t$
kód detekuje d – 1 chyb a opravuje max $\frac{d-1}{2}$ chyb
Lineární binární blokový kód je takový, že $\forall u,v \in C \quad u+v \in C$
Hammingova váha slova … #jedniček
Věta: Minimální váha netriviálního kódu je $d(C)$

Kontrolní matice *H* kódu $C \quad \forall x \in C ; x = (x_1 \dots x_n)$

$$H x^T = H \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}$$

… tedy $C = \{ x \in \{0,1\}^n \vee H x^T = 0^T \}$
Množina vektorů *U* je lineárně nezávislá

$$\nexists u \in U \in t.ž.u\ je\ lin.kombinací\ U \setminus \{u\}$$

Báze je lineárně nezávislá podmnožina U prostoru V t.ž. obal(U) = V (tedy generuje V)

Dimenze prostoru dim(V) = $|U|$

Nechť $U = \{e_1, \ldots, e_n\}$ je báze V, potom

$$\forall u \in V\ \exists!\ a_1 \ldots a_n \in F\ t.ž.\ u = a_1 e_1 + \ldots + a_n e_n$$

Nechť F je konečné těleso. **Lineární (n, k) kód** je k-dimenzionální podprostor F^n
… k informačních bitů, n-k kontrolních bitů, … báze e_1 … e_k

$$\forall v \in C\ v = \sum_1^k u_i e_i \text{ kde } u_i \in F$$

tzn. každé slovo $u = u_1 \ldots u_n$ definuje kódové slovo v

Generující matice kódu *C*:

$$G = \begin{pmatrix} e_1 \\ \vdots \\ e_k \end{pmatrix} = \begin{pmatrix} e_{11} & \cdots & e_{1n} \\ \vdots & \ddots & \vdots \\ e_{k1} & \cdots & e_{kn} \end{pmatrix}$$

kódování slova u

$$e(u) = uG$$

Systematický kód … G má tvar

$$G = (E \mid B)$$

kde E je jednotková matice k x k a B je k x n-k matice

Ekvivalentní kód : C_1 a C_2 jsou ekvivalentní pokud existuje permutace Π nad {1..n} taková, že

$$u_1, \ldots, u_n \in C_1 \quad u_{\Pi(1)}, \ldots, u_{\Pi(n)} \in C_2$$

Kontrolní matice H ... $\forall u \in C\ Hu^T = 0^T$

Pro systematický kód C s generující maticí $G = (E_1 \mid B)$ platí

$$H = (-B^T \mid E_2)$$

je kontrolní matice, přičemž E_1 je jednotková matice k x k a $E2$ je jednotková matice n-k x n-k

Syndrom *S*… nechť $v = u + e$, e buď chybové slovo, H kontrolní matice, potom

$$S^T = Hv^T$$

… zřejmě $Hv^T = He^T$

Buď C lineární (n, k) kód, H kontrolní matice, G generující matice, potom **třídou slova *e*** nazýváme

$$\forall e \in F^n \; e+C = \{e+u \vee u \in C\}$$

…tzn všechna slova, která mohou přijít pokud nastane chyba e
všechna $e+C$ mají stejný syndrom
Reprezentant *r* je slovo z $e+C$ s minimální Hamiltonovou váhou

Odtud již plyne postup dekódování lineárního kódu:

1. přijmeme v, spočteme syndrom $S^T = Hv^T$
2. Najdeme reprezentanta $s+C = r$
3. $u=v-r$ (předpokládáme minimální chybu)

Goppa kódy
jsou speciální příklad lineárních kódů
g polynom nad tělesem $F = GF(2^n)$ stupně t
$L = (L_1, \ldots, L_n)$, kde $L_i \in F$ & $L_i \neq L_j$ & $L_i \neq 0$ … L je báze kódu

$$\Gamma(gL) = \left\{ c \in GF(2^n) \vee \sum_{i=1}^{n} \frac{c_i}{x - L_i} \equiv 0 \, mod(g(x)) \right\}$$

Eliptická aritmetika

nahrazuje počítání v „obyčejných“ tělesech a okruzích za trochu složitější operace ve specifických podmnožinách (nebo dokonce podmnožinách podmnožin)

vezmu těleso – nad ním zadefinuji křivku – nemá-li provočíselný řád, vyberu podkřivku provčíselného řádu a počítáme

pokud věc opravdu brutálně zjednoduším:

pokud pro body P *a* Q nějaké eliptické křivky platí $Q = kP$ pro nějaké k, není problém spočítat Q na základě k a P (pomocí sčítání a zdvojování) ale pro dané Q a P neznáme rychlý způsob, jak určit k

v současné době si proto myslíme, že vystačíme s čísly o velikosti 150 – 250 bitů

Weirstrassova forma

je obecná forma definice eliptické křivky

eliptickou křivkou nad tělesem K rozumíme množinu všech bodů (x, y) pro které platí

$$y^2 + a\,y = x^3 + b\,x^2 + c\,x\,y + d\,x + e,$$

kde a, b, c, d, e jsou prvky K
ne úplně snadno se nad touto reprezentací implementuje, navíc potřebujete doplnit neutrální prvek / identitu … tzv. bod v nekonečnu

najít vhodnou EC je netriviální

(přechýlené) Edwardsovy křivky

Křivkou je množina bodů vyhovující rovnici

$$x^2 + y^2 = a^2 + a^2x^2y^2$$

určitým rozšířením jsou tzv. přechýlené Edwardsovy křivky (twisted EC)

$$ax^2 + y^2 = 1 + dx^2y^2$$

Sčítání

nechť (x_1, y_1) a (x_2, y_2) jsou body na edwardsově křivce

bod (x_3, y_3) je na téže křivce

$x_3 = (x_1y_2 + x_2y_1)/(a.(1+x_1y_1x_2y_2))$

$y_3 = (y_1y_2 – x_1x_2)/(a.(1 – x_1y_1x_2y_2))$

(důkaz není složitý ale trochu pracný – dosadíme a roznásobíme)
položme $(x_3, y_3) = (x_1, y_1) + (x_2, y_2)$

Zdvojování

nahradíme-li bod (x_2, y_2) bodem (x_1, y_1) dostaneme jednodušší tvar sčítací formule

$(x_1, y_1) + (x_1, y_1) = (x_3, y_3)$

$x_3 = (x_1y_1 + x_1y_1)/(a.(1+x_1y_1x_1y_1)) = (x_1y_1 + x_1y_1)/(a.(1+x_1^2y_1^2))$

$y_3 = (y_1y_1 – x_1x_1)/(a.(1 – x_1y_1x_1y_1)) = (y_1^2 – x_1^2)/(a.(1 – x_1^2y_1^2))$

...a zbývá jediné – toto implementovat (jako výpočet v konstantním čase)

Montgomeryho žebřík

```
R0 ← 0
R1 ← P
for i from m downto 0 do
    if di = 0 then
        R1 ← point_add(R0, R1)
        R0 ← point_double(R0)
    else
        R0 ← point_add(R0, R1)
        R1 ← point_double(R1)

    // invariant property to maintain correctness
    assert R1 == point_add(R0, P)
return R0
```