Univerza v Ljubljani
Fakulteta za računalništvo in informatiko
Fakulteta za matematiko in fiziko

Janko Purgaj

Implementacija kriptosistema NTRUEncrypt

diplomsko delo
na interdisciplinarnem univerzitetnem študiju
računalništva in matematike

Mentor: prof. dr. Marko Petkovšek

Ljubljana 2016

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Janko Purgaj, z vpisno številko **63050210**, sem avtor diplomskega dela z naslovom:

Implementacija kriptosistema NTRUEncrypt

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom prof. dr. Marka Petkovška,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki "Dela FRI".

V Ljubljani, dne 24. avgusta 2016 Podpis avtorja:

Zahvaljujem se družini in prijateljem, ki so me podpirali na študijski poti. Posebej se zahvaljujem mentorju, prof. dr. Marku Petkovšku za vso pomoč in potrpežljivost pri izdelavi diplomske naloge.

Kazalo

Povzetek

Abstract

1 Uvod **1**

2 Matematično ozadje **3**
2.1 Celoštevilske rešetke 3
2.2 Rešetke in kratki vektorji 6
2.3 Algoritmi za redukcijo rešetk 7

3 Kriptosistem NTRUEncrypt **13**
3.1 Opis 13
3.2 Izbor parametrov 13
3.3 Generiranje ključev 14
3.4 Šifriranje 15
3.5 Dešifriranje 16
3.6 Varnost 18
3.7 Primer 21

4 Pohitritve **23**
4.1 Izbira privatnega polinoma 23

5 Implementacija **25**
5.1 Algoritem za hitro množenje polinomov 25
5.2 Inverz polinoma 28

6 Zaključek **31**

Povzetek

V svetu hitrega tehnološkega razvoja se kaže potreba po razvijanju novih, varnejših in hitrejših kriptosistemov. Varnost klasičnih asimetričnih kriptosistemov, kot so RSA, Diffie-Hellmanova izmenjava ključa, Elgamalov kriptosistem ali kriptosistemi, ki uporabljajo eliptične krivulje, temelji na težavnosti razcepa naravnih števil oziroma na težavnosti problema diskretnega logaritma. Ker za oba omenjena problema obstajajo učinkoviti kvantni algoritmi (Shor 1994 [10], Proos & Zalka 2003 [6]), bodo s prihodom kvantnih računalnikov ti kriptosistemi postali ranljivi. Varnost asimetričnega kriptosistema NTRUEncrypt pa temelji na problemih najkrajšega oziroma najbližjega vektorja v celoštevilskih rešetkah, za katera še ne poznamo učinkovitih kvantnih algoritmov. Zato NTRUEncrypt predstavlja zanimivo alternativo klasičnim asimetričnim kriptosistemom.

Ključne besede: kriptografija, NTRUEncrypt, varnost.

Abstract

In the world of rapid technological development there is a need to design new, safer and faster cryptosystems. Security of classical asymmetric cryptosystems such as RSA, Diffie-Hellman key exchange, Elgamal cryptosystem, or cryptosystems using elliptic curves is based on the difficulty of factoring integers and on the difficulty of the discrete logarithm problem respectively. Since for both of these problems there exist efficient quantum algorithms (Shor 1994 [10], Proos & Zalka 2003 [6]), the advent of quantum computers will render these systems unsafe. On the other hand, security of the NTRUEncrypt asymmetric cryptosystem is based on the difficulty of the shortest vector and of the closest vector problems in integer lattices, for which no efficient quantum algorithms are known to date. Therefore NTRUEncrypt represents an interesting alternative to the classical asymmetric cryptosystems.

Keywords: cryptography, NTUEncrypt, security.

Poglavje 1

Uvod

Kriptografija je veda, ki preučuje tehnike za zagotavljanje varne komunikacije. V splošnem temelji na gradnji in analizi protokolov, ki preprečujejo, da bi tretje osebe oziroma javnost bila zmožna prebirati privatna sporočila. Moderna kriptografija je močno povezana z matematičnimi teorijami in računalništvom. Kriptografski algoritmi so zgrajeni na matematičnih problemih, za katere predpostavljamo, da jih je težko rešiti v doglednem času.
V moderni kriptografiji ločimo kriptografijo s simetričnimi ter asimetričnimi ključi. Pri prvi se uporablja isti ključ tako za šifriranje kot za dešifriranje, medtem ko se pri asimetrični kriptografiji uporablja javni ključ za šifriranje in privatni ključ za dešifriranje sporočila. V praksi uporabljamo kombinacijo obeh, in sicer uporabimo asimetričen kriptografski sistem za izmenjavo šifrirnega ključa simetričnega kriptografskega sistema, pri katerem je šifriranje in dešifriranje zaradi bistveno krajših ključev mnogo hitrejše.
Kriptosistema DES (Data Encryption Standard) in AES (Advanced Encryption Standard), katerih zasnova so bločne šifre, sta primera simetričnih kriptosistemov. Whitfield Diffie in Martin Hellman sta prva objavila članek o kriptografiji, ki uporablja javni ključ. Njuno delo temelji na izmenjavi ključa preko javnega kanala. Primeri asimetričnih kriptosistemov, ki so jima sledili, so sistemi kot sta RSA (Rivest-Shamir-Adleman) in Rabinov kriptosistem, ki temeljita na množenju velikih praštevil, ter sistemi, ki uporabljajo eliptične krivulje.
V tem diplomskem delu je predstavljen kriptosistem NTRUEncrypt in njegova implementacija. NTRUEncrypt spada v vrsto asimetrične kriptografije in temelji na problemih najkrajšega oziroma najbližjega vektorja v celoštevilskih rešetkah. Prednost, ki jo ima pred trenutno uveljavljenimi asimetričnimi sistemi, kot so RSA, ElGamal in podobni, je njegova hitrost šifriranja in dešifriranja. Prav tako še ni znan kvantni algoritem, ki bi učinkovito reševal problem najkrajšega oziroma najbližjega vektorja.

Struktura dela je sledeča:

- V poglavju 2 so predstavljeni matematični pojmi in definicije, potrebne za razumevanje nadaljnjih poglavij.
- Poglavje 3 opiše delovanje kriptosistema NTRUEncrypt, njegovo kriptoanalizo in se zaključi s preprostim primerom.
- V poglavju 4 so opisane predlagane pohitritve kriptosistema NTRUEncrypt.
- Kompleksnejši algoritmi in njihova implementacija so opisani v poglavju 5, izvorna koda v Javi pa se nahaja na priloženem CD-ju.

Poglavje 2

Matematično ozadje

V tem poglavju so opisani osnovni matematični pojmi, ki so potrebni za razumevanje nadaljnjih poglavij.

2.1 Celoštevilske rešetke

Celoštevilska rešetka (ali kratko: rešetka) v m-razsežnem evklidskem prostoru $\mathbb{R}^m$ je množica

$$L = \mathcal{L}(v_1, \ldots, v_n) = \left\{ \sum_{i=1}^{n} a_i v_i : a_i \in \mathbb{Z} \right\} \tag{2.1}$$

vseh celoštevilskih linearnih kombinacij n linearno neodvisnih vektorjev $v_1, \ldots, v_n$ v $\mathbb{R}^m$ ($m \geq n$). Števili m in n imenujemo *rang* oziroma *dimenzija* rešetke. Množica $\{v_1, \ldots, v_n\}$ predstavlja bazo rešetke in jo lahko podamo z matriko

$$B = \begin{bmatrix} v_1 \\ \vdots \\ v_n \end{bmatrix} \in \mathbb{R}^{n \times m}, \tag{2.2}$$

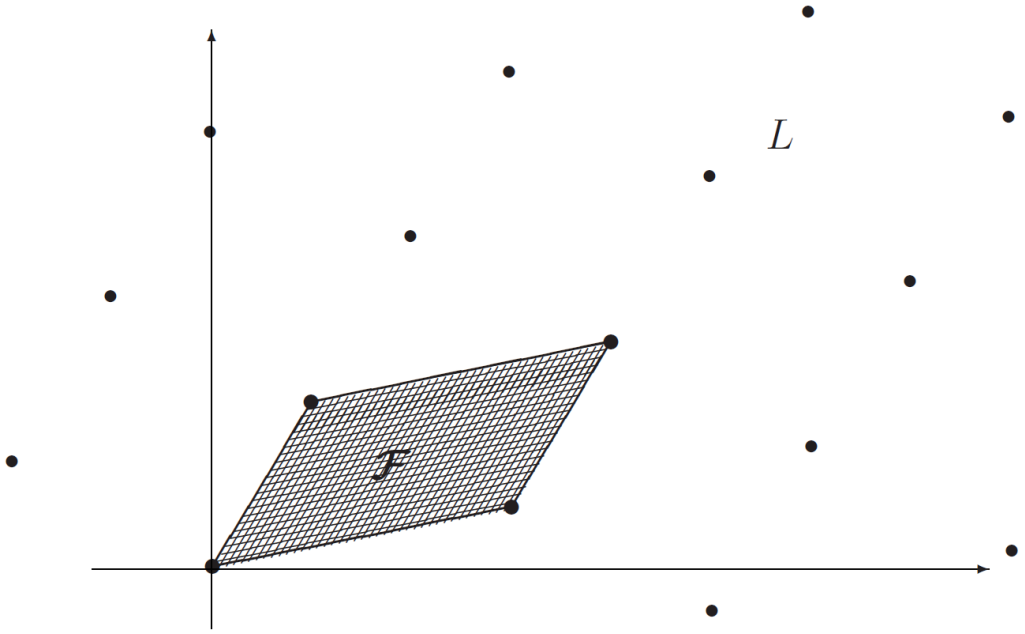
kjer bazni vektorji predstavljajo vrstice. Z uporabo matričnega zapisa lahko zapišemo (2.1) kot

$$\mathcal{L}(B) = \{xB : x \in \mathbb{Z}^n\}. \tag{2.3}$$

Izrek 2.1. *Naj bosta L_1 in L_2 n-razsežni rešetki ranga m z baznima matrikama B_1 in B_2. Potem je $L_1 = L_2$ natanko tedaj, ko obstaja matrika $U \in \mathbb{Z}^{n \times n}$ z $\det U = \pm 1$, tako da je $B_2 = UB_1$.*

Dokaz. ($\Rightarrow$) Če je $L_1 = L_2$, so vrstice matrike B_2 celoštevilske linearne kombinacije vrstic matrike B_1 in obratno, torej obstajata matriki

$U, V \in \mathbb{Z}^{n\times n}$, tako da je $B_2 = UB_1$ in $B_1 = VB_2$. Od tod sledi, da je $B_1 = VUB_1$ oziroma $(I - VU)B_1 = 0$. Ker so vrstice matrike B_1 linearno neodvisne, je matrika B_1 obrnljiva, torej je $VU = I$ in $\det U \det V = 1$. Matriki U in V sta celoštevilski, zato sta takšni tudi njeni determinanti, torej je $\det U = \det V = \pm 1$.
($\Leftarrow$) Naj bo $B_2 = UB_1$, kjer je $U \in \mathbb{Z}^{n\times n}$ in $\det U = \pm 1$. Potem je $B_1 = U^{-1}B_2$, kjer je $U^{-1} \in \mathbb{Z}^{n\times n}$. Torej so vrstice matrike B_1 celoštevilske linearne kombinacije vrstic matrike B_2 in pripadajo rešetki L_2, vrstice matrike B_2 pa so celoštevilske linearne kombinacije vrstic matrike B_1 in pripadajo rešetki L_1. Od tod sledi, da je $L_1 \subseteq L_2$ in $L_2 \subseteq L_1$, torej je $L_1 = L_2$. □
Rešetke so, podobno kot vektorski prostori, generirane z vsemi linearnimi kombinacijami svojih baznih vektorjev, vendar se za razliko od vektorskih prostorov uporabljajo celoštevilski koeficienti namesto realnih. Predstavljamo si jih lahko kot urejeno razporeditev točk v $\mathbb{R}^m$, kjer postavimo točko na koncu vsakega vektorja. Primer rešetke v $\mathbb{R}^2$ prikazuje slika 2.1.



Slika 2.1: Rešetka L in njena temeljna domena $\mathcal{F}$

Definicija. *Naj bo L rešetka dimenzije n in $B = \{v_1, v_2, \ldots, v_n\}$ njena baza.* Temeljna domena *(ali* temeljni paralelepiped*) rešetke L, ki ustreza bazi B, je množica*

$$\mathcal{F}(v_1, \ldots, v_n) = \{t_1v_1 + t_2v_2 + \cdots + t_nv_n : 0 \leq t_i < 1\}. \tag{2.4}$$

Temeljna domena rešetke v dimenziji 2 je prikazana na sliki 2.1.

Izrek 2.2. *Naj bo $L \subset \mathbb{R}^n$ rešetka dimenzije n in naj bo $\mathcal{F}$ njena temeljna domena. Potem lahko vsak vektor $w \in \mathbb{R}^n$ zapišemo v obliki*

$$w = t + v \text{ za enolično } t \in \mathcal{F} \text{ in } v \in L.$$

Prav tako unija translacij temeljnih domen

$$\mathcal{F} + v = \{t + v : t \in \mathcal{F}\}$$

kjer so v vektorji v rešetki L, pokriva celoten prostor $\mathbb{R}^n$ (glej sliko 2.2).

Dokaz. Naj bo $v_1, \ldots, v_n$ baza rešetke L, ki ji ustreza temeljna domena $\mathcal{F}$. Potem so $v_1, \ldots, v_n$ linearno neodvisni v $\mathbb{R}^n$, torej so baza $\mathbb{R}^n$. Zato lahko $w \in \mathbb{R}^n$ zapišemo kot

$$w = \alpha_1 v_1 + \alpha_2 v_2 + \cdots + \alpha_n v_n \text{ za neke } \alpha_1, \ldots, \alpha_n \in \mathbb{R}.$$

Vsak α_i zapišemo kot

$$\alpha_i = t_i + a_i, \text{ kjer je } 0 \leq t_i < 1 \text{ in } a_i \in \mathbb{Z}.$$

Potem

$$w = \overbrace{t_1 v_1 + t_2 v_2 + \cdots + t_n v_n}^{\text{vektor } t \in \mathcal{F}} + \overbrace{a_1 v_1 + a_2 v_2 + \cdots + a_n v_n}^{\text{vektor } v \in L},$$

kar pomeni, da lahko zapišemo w v želeni obliki.

Recimo, da lahko vektor $w = t + v = t' + v'$ predstavimo z vsoto vektorja iz $\mathcal{F}$ in vektorja iz L na dva načina. Potem

$$\begin{aligned}&(t_1 + a_1)v_1 + (t_2 + a_2)v_2 + \cdots + (t_n + a_n)v_n = \\ &= (t'_1 + a'_1)v_1 + (t'_2 + a'_2)v_2 + \cdots + (t'_n + a'_n)v_n.\end{aligned}$$

Ker so $v_1, \ldots, v_2$ med sabo linearno neodvisni, sledi

$$t_i + a_i = t'_i + a'_i \text{ za vse } i = 1, 2, \ldots, n.$$

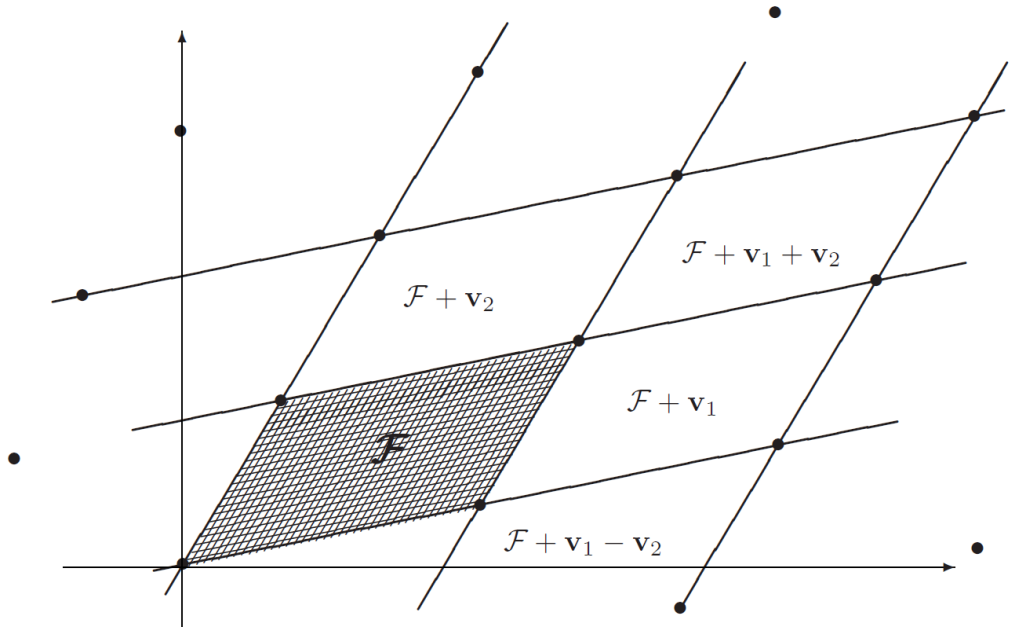
Torej je

$$t_i - t'_i = a'_i - a_i \in \mathbb{Z}$$

celo število. Ker vemo, da sta števili t_i in t'_i večji ali enaki 0 in strogo manjši od 1, je $t_i - t'_i$ celo število natanko tedaj, kadar $t_i = t'_i$. Torej $t = t'$, zato tudi

$$v = w - t = w - t' = v'.$$

To dokazuje, da sta $t \in \mathcal{F}$ in $v \in L$ enolično določena z w. □



Slika 2.2: Translacije $\mathcal{F}$ z vektorji iz L pokrivajo $\mathbb{R}^n$

2.2 Rešetke in kratki vektorji

Osnovna računska problema, povezana z rešetkami, sta:

- **Problem najkrajšega vektorja** (angl. *The Shortest Vector Problem* - SVP): poišči najkrajši neničelni vektor v rešetki L. Npr. poišči neničelni vektor $v \in L$, ki minimizira evklidsko normo $\|v\|$.
- **Problem najbližjega vektorja** (angl. *The Closest Vector Problem* - CVP): Za dani vektor $w \in \mathbb{R}^m$, ki ni vsebovan v L, poišči vektor $v \in L$, ki mu je najbližji. Npr. poišči vektor $v \in L$, ki minimizira evklidsko normo $\|w - v\|$.

2.2.1 Babai-jev algoritem za iskanje približka CVP

Če ima rešetka $L \subset \mathbb{R}^n$ bazo vektorjev $v_1, \ldots, v_n$, ki so paroma pravokotni:

$$v_i \cdot v_j = 0 \text{ za vse } i \neq j,$$

potem sta SVP in CVP lahko rešljiva. Za rešitev SVP opazimo, da je dolžina kateregakoli vektorja v L podana s formulo

$$\|a_1 v_1 + a_2 v_2 + \cdots + a_n v_n\|^2 = a_1^2 \|v_1\|^2 + a_2^2 \|v_2\|^2 + \cdots + a_n^2 \|v_n\|^2.$$

Ker so $a_1, \ldots, a_n \in \mathbb{Z}$, vidimo, da je najkrajši neničelni vektor v L najkrajši vektor v množici $\{\pm v_1, \ldots, \pm v_n\}$.
Podobno velja, če želimo najti danemu vektorju $w \in \mathbb{R}^n$ najbližji vektor v L. Najprej zapišemo

$$w = t_1 v_1 + t_2 v_2 + \cdots + t_n v_n,\ t_1, \ldots, t_n \in \mathbb{R}.$$

Potem za $v = a_1 v_1 + \cdots + a_n v_n \in L$ velja

$$\|v - w\|^2 = (a_1 - t_1)^2 \|v_1\|^2 + (a_2 - t_2)^2 \|v_2\|^2 + \cdots + (a_n - t_n)^2 \|v_n\|^2. \quad (2.5)$$

Ker so a_i cela števila, bo vrednost (2.5) najmanjša, če za a_i vzamemo celo število, najbližje pripadajočemu t_i. Vendar se v primeru, ko bazni vektorji med sabo niso dovolj pravokotni, algoritem ne obnese najbolje.

2.3 Algoritmi za redukcijo rešetk

V tem razdelku bomo najprej opisali Gaussov algoritem za redukcijo rešetk v dimenziji 2, nato bomo predstavili algoritem LLL, ki vrne približno pravokotno bazo s kratkimi vektorji.

2.3.1 Gaussov algoritem za redukcijo rešetk v dimenziji 2

Glavna ideja algoritma, ki ga prikazuje slika 1, je izmenično odštevanje enega baznega vektorja od drugega, dokler to ni več mogoče.
Naj bosta v_1 in v_2 bazna vektorja 2-razsežne rešetke $L \subset \mathbb{R}^2$. Pod predpostavko, da je $\|v_1\| \leq \|v_2\|$ (če ni izpolnjena, vektorja v_1 in v_2 zamenjamo med seboj), želimo skrajšati vektor v_2 tako, da od njega odštejemo ustrezen večkratnik vektorja v_1,

$$v_2^* = v_2 - \alpha v_1.$$

Število α izberemo tako, da bo vektor v_2^* najkrajši, torej da bo pri tem α kvadratna funkcija

$$f(\alpha) = \|v_2^*\|^2 = \|v_2\|^2 - 2\alpha(v_1 \cdot v_2) + \alpha^2 \|v_1\|^2$$

dosegla minimum. To se zgodi, ko je $f'(\alpha) = -2(v_2 \cdot v_1) + 2\alpha\|v_1\|^2 = 0$ oziroma ko je $\alpha = \frac{v_1 \cdot v_2}{\|v_1\|^2}$. Tedaj je vektor $v_2^* = v_2 - \frac{v_1 \cdot v_2}{\|v_1\|^2} v_1$ pravokoten na v_1. Seveda $\frac{v_1 \cdot v_2}{\|v_1\|^2}$ v splošnem ne bo celo število, zato za novi v_2 vzamemo

$$v_2^* = v_2 - \left\lfloor \frac{v_1 \cdot v_2}{\|v_1\|^2} \right\rceil v_1,$$

kjer smo z $\lfloor x \rceil$ označili celo število, za katero je $|x - \lfloor x \rceil| \leq \frac{1}{2}$.

```
vhod: bazna vektorja v_1, v_2 rešetke L
izhod: skrajšana bazna vektorja rešetke L
while True do
    if ||v_2|| < ||v_1|| then
        zamenjaj v_1 in v_2
    end
    m = ⌊(v_1 · v_2)/||v_1||^2⌉
    if m = 0 then
        return v_1, v_2
    end
    v_2 = v_2 − m v_1
end
```

Slika 1: Gaussov algoritem za redukcijo baze

Ko se algoritem ustavi, je v_1 najkrajši neničelni vektor v L, kar pomeni, da algoritem rešuje SVP.

Dokaz. Dokažimo, da je v_1 najkrajši neničelni vektor v rešetki. Algoritem vrne vektorja v_1 in v_2, torej velja $\|v_2\| \geq \|v_1\|$ in $m = 0$ oziroma

$$\frac{|v_1 \cdot v_2|}{\|v_1\|^2} \leq \frac{1}{2}. \tag{2.6}$$

Recimo, da je $v \in L$ katerikoli neničelni vektor v L. Če zapišemo

$$v = a_1 v_1 + a_2 v_2,\ a_1, a_2 \in \mathbb{Z},$$

dobimo

$$\begin{aligned}
\|v\|^2 &= \|a_1 v_1 + a_2 v_2\|^2 \\
&= a_1^2\|v_1\|^2 + 2a_1 a_2 (v_1 \cdot v_2) + a_2^2\|v_2\|^2 \\
&\geq a_1^2\|v_1\|^2 - 2|a_1 a_2||v_1 \cdot v_2| + a_2^2\|v_2\|^2 \\
&\geq a_1^2\|v_1\|^2 - |a_1 a_2|\|v_1\|^2 + a_2^2\|v_2\|^2 && \text{iz (2.6)} \\
&\geq a_1^2\|v_1\|^2 - |a_1 a_2|\|v_1\|^2 + a_2^2\|v_1\|^2 && \text{saj } \|v_2\| \geq \|v_1\| \\
&= (|a_1|^2 - |a_1||a_2| + |a_2|^2)\|v_1\|^2.
\end{aligned}$$

Za katerikoli realni števili t_1 in t_2 je količina

$$t_1^2 - t_1 t_2 + t_2^2 = \left(t_1 - \frac{1}{2}t_2\right)^2 + \frac{3}{4}t_2^2$$

lahko nič le v primeru, ko je $t_1 = t_2 = 0$. Ker sta a_1 in a_2 celi števili in ne obe 0, velja torej $a_1^2 - |a_1||a_2| + a_2^2 \geq 1$, kar dokazuje, da je v_1 najmanjši neničelni vektor v rešetki L. □

2.3.2 Algoritem LLL

Gaussov algoritem, opisan v prejšnjem podpoglavju, v 2-razsežnem prostoru učinkovito najde najkrajši neničelni vektor v rešetki, vendar iskanje takšnega vektorja postane težje, ko dodajamo dimenzije. Naš cilj je, za dano bazo $\{v_1, v_2, \ldots, v_n\}$ v rešetki L najti boljšo bazo, ki sestoji iz čim krajših, med seboj čim bolj pravokotnih vektorjev. Hadamardova neenakost pravi

$$\det L = \mathrm{Vol}(\mathcal{F}) \leq \|v_1\|\|v_2\| \cdots \|v_n\|.$$

Bolj kot so vektorji med sabo pravokotni, bolj se zgornja neenakost približuje enakosti. Pri grajenju nove, izboljšane baze si pomagamo z Gram-Schmidtovo pravokotno bazo. Začnemo z $v_1^* = v_1$ in za $i = 2, 3, \ldots, n$ računamo

$$v_i^* = v_i - \sum_{j=1}^{i-1} \mu_{i,j} v_j^*, \text{ kjer je } \mu_{i,j} = \frac{v_i \cdot v_j^*}{\|v_j^*\|^2} \text{ za } 1 \leq j \leq i-1. \tag{2.7}$$

Izrek 2.3. *Če je $\mathcal{B} = \{v_1, v_2, \ldots, v_n\}$ baza rešetke L in $\mathcal{B}^* = \{v_1^*, v_2^*, \ldots, v_n^*\}$ njena pravokotna Gram-Schmidtova baza, potem velja*

$$\det(L) = \prod_{i=1}^{n} \|v_i^*\|.$$

Opomba: $\mathcal{B}^*$ ni nujno baza rešetke L, saj Gram-Schmidtov algoritem izračuna linearno kombinacijo vektorjev z realnimi koeficienti.

Dokaz. Naj bo $F = F(v_1, \ldots, v_n)$ matrika, kjer so koordinate i-tega baznega vektorja v i-ti vrstici:

$$v_i = (r_{i1}, r_{i2}, \ldots, r_{in}),$$

$$F = F(v_1, \ldots, v_n) = \begin{pmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ r_{21} & r_{22} & \cdots & r_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ r_{n1} & r_{n2} & \cdots & r_{nn} \end{pmatrix}.$$

Iz linearne algebre je znano, da je volumen paralelepipeda, ki ga razpenjajo vektorji $v_1, v_2, \ldots, v_n$, enak absolutni vrednosti determinante matrike $F = F(v_1, \ldots, v_2)$, torej velja $\det(L) = |\det F|$.
Naj bo $F^* = F(v_1^*, v_2^*, \ldots, v_n^*)$ matrika, v kateri so vrstice koordinate vektorjev $v_1^*, \ldots, v_n^*$. Matriki F in F^* sta povezani z enačbo

$$MF^* = F,$$

kjer je M matrika za spremembo baze

$$M = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 & 0 \\ \mu_{2,1} & 1 & 0 & \cdots & 0 & 0 \\ \mu_{3,1} & \mu_{3,2} & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \mu_{n-1,1} & \mu_{n-1,2} & \mu_{n-1,3} & \cdots & 1 & 0 \\ \mu_{n,1} & \mu_{n,2} & \mu_{n,3} & \cdots & \mu_{n,n-1} & 1 \end{pmatrix}.$$

Opazimo, da je $\det(M) = 1$, zato

$$\det(L) = |\det F| = |\det(MF^*)| = |(\det M)(\det F^*)| = |\det F^*| = \prod_{i=1}^{n} \|v_i^*\|.$$

□

Definicija. *Naj bo $\mathcal{B} = \{v_1, v_2, \ldots, v_n\}$ baza rešetke L in $\mathcal{B}^* = \{v_1^*, v_2^*, \ldots, v_n^*\}$ njena Gram-Schmidtova pravokotna baza. Rečemo, da je baza $\mathcal{B}$* LLL-reducirana*, če veljata pogoja:*

$$(\textbf{\textit{pogoj velikosti}})\ |\mu_{i,j}| = \frac{|v_i \cdot v_j^*|}{\|v_j^*\|^2} \leq \frac{1}{2},\ \textit{za vse}\ 1 \leq j < i \leq n,$$

$$(\textbf{\textit{Lovászov pogoj}})\ \|v_i^*\|^2 \geq \left(\frac{3}{4} - \mu_{i,i-1}^2\right) \|v_{i-1}^*\|^2,\ \textit{za vse}\ 1 < i \leq n.$$

Izrek 2.4. *Naj bo L rešetka dimenzije n. Vsaka LLL-reducirana baza $\{v_1, v_2, \ldots, v_n\}$ za L ima naslednji lastnosti:*

$$\prod_{i=1}^{n} \|v_i\| \leq 2^{n(n-1)/4} \det L, \tag{2.8}$$

$$\|v_j\| \leq 2^{(i-1)/2} \|v_i^*\|\ \textit{za vse}\ 1 \leq j \leq i \leq n. \tag{2.9}$$

Za prvi vektor LLL-reducirane baze velja:

$$\|v_1\| \leq 2^{(n-1)/4} |\det L|^{1/n}, \tag{2.10}$$

$$\|v_1\| \leq 2^{(n-1)/2} \min_{0 \neq v \in L} \|v\|. \tag{2.11}$$

To pomeni, da LLL-reducirana baza rešuje približek SVP do faktorja $2^{(n-1)/2}$ natanko.

Dokaz. Lovászov pogoj in dejstvo, da je $|\mu_{i,i-1}| \leq \frac{1}{2}$, nam povesta, da je

$$\|v_i^*\|^2 \geq \left(\frac{3}{4} - \mu_{i,i-1}^2\right) \|v_{i-1}^*\|^2 \geq \frac{1}{2} \|v_{i-1}^*\|^2.$$

Iz zgornje neenačbe za $1 \leq j \leq i \leq n$ dobimo uporaben približek

$$\|v_j^*\|^2 \leq 2^{i-j}\|v_i^*\|^2. \tag{2.12}$$

Nato izračunamo

$$\begin{aligned}
\|v_i\|^2 &= \left\| v_i^* + \sum_{j=1}^{i-1} \mu_{i,j} v_j^* \right\|^2 && \text{iz (2.7)} \\
&= \|v_i^*\|^2 + \sum_{j=1}^{i-1} \mu_{i,j}^2 \|v_j^*\|^2 && \text{saj so } v_1^*, \ldots, v_n^* \text{ pravokotni} \\
&\leq \|v_i^*\|^2 + \sum_{j=1}^{i-1} \frac{1}{4}\|v_j^*\|^2 && \text{ker } |\mu_{i,j}| \leq \frac{1}{2} \\
&\leq \|v_i^*\|^2 + \sum_{j=1}^{i-1} 2^{i-j-2}\|v_i^*\|^2 && \text{iz (2.12)} \\
&= \frac{1 + 2^{i-1}}{2}\|v_i^*\|^2 \\
&\leq 2^{i-1}\|v_i^*\|^2 && \text{saj } 1 \leq 2^{i-1} \text{ za vse } i \geq 1 \qquad (2.13)
\end{aligned}$$

Če za $1 \leq i \leq n$ množimo enačbo (2.13) samo s sabo, dobimo

$$\prod_{i=1}^{n} \|v_i\|^2 \leq \prod_{i=1}^{n} 2^{i-1}\|v_i^*\|^2 = 2^{n(n-1)/2} \prod_{i=1}^{n} \|v_i^*\|^2 = 2^{n(n-1)/2} (\det L)^2,$$

pri čemer smo zadnji dve enačbi izpeljali z uporabo izreka 2.3. Ko neenačbo korenimo, smo dokazali lastnost (2.8).
Za katerikoli $j \leq i$ uporabimo (2.13) (pri $i = j$) in (2.12), da dobimo približek

$$\|v_j\|^2 \leq 2^{j-1}\|v_j^*\|^2 \leq 2^{j-1} \cdot 2^{i-j}\|v_i^*\|^2 = 2^{i-1}\|v_i^*\|^2.$$

Ko neenačbo korenimo, smo dokazali lastnost (2.9).
Če v neenačbi (2.9) postavimo $j = 1$ in jo zmnožimo za $1 \leq i \leq n$, dobimo

$$\|v_i\|^n \leq \prod_{i=1}^{n} 2^{(i-1)/2}\|v_i^*\| = 2^{n(n-1)/4} \prod_{i=1}^{n} \|v_i^*\| = 2^{n(n-1)/4} \det L$$

Ko na obeh straneh izračunamo n-ti koren, dobimo približek (2.10).
Naj bo vektor $v \in L$ neničelni vektor rešetke

$$v = \sum_{j=1}^{i} a_j v_j = \sum_{j=1}^{i} b_j v_j^*,$$

kjer $a_i \neq 0$, $a_1, \ldots, a_i \in \mathbb{Z}$ in $b_1, \ldots, b_i \in \mathbb{R}$. Velja $|a_i| \geq 1$.
Ker vemo, da so vektorji $v_1^*, \ldots, v_k^*$ paroma pravokotni za vsak k in razpenjajo isti prostor, kot vektorji $v_1, \ldots, v_k$, nas enačbe

$$v \cdot v_i^* = a_i v_i \cdot v_i^* = b_i v_i^* \cdot v_i^* \text{ in } v_i \cdot v_i^* = v_i^* \cdot v_i^*$$

privedejo do zaključka, da je $a_i = b_i$, kar pomeni, da je $|b_i| = |a_i| \geq 1$. S tem in z uporabo (2.9) (za $j = 1$) dobimo približek

$$\|v\|^2 = \sum_{j=1}^{i} b_j^2 \|v_j^*\|^2 \geq b_i^2 \|v_i^*\|^2 \geq \|v_i^*\|^2 \geq 2^{-(i-1)} \|v_1\|^2 \geq 2^{-(n-1)} \|v_1\|^2.$$

Ko neenačbo korenimo, smo dokazali približek (2.11). □

Izrek 2.5. *Naj bo $\{v_1, \ldots, v_n\}$ baza za rešetko L. Algoritem LLL, prikazan na sliki 2, se zaključi v končno mnogo korakih in vrne LLL-reducirano bazo za L.*
Če je $B = \max \|v_i\|$, algoritem izvede glavno zanko $\mathcal{O}(n^2 \log n + n^2 \log B)$ krat.

```
1  vhod: baza {v_1, v_2, ..., v_n} rešetke L
2  izhod: LLL-reducirana baza rešetke L
3  k = 2
4  while k ≤ n do
5      for j = k − 1, k − 2, ..., 1 do
6          v_k = v_k − ⌊μ_{k,j}⌉ v_j /* Manjšanje velikosti */
7      end
       /* Lovászev pogoj */
8      if ‖v_k^*‖^2 ≥ (3/4 − μ_{k,k−1}^2) ‖v_{k−1}^*‖^2 then
9          k = k + 1
10     else
11         zamenjaj v_{k−1} in v_k
12         k = max(k − 1, 2)
13     end
14 end
15 return {v_1, ..., v_n}
```

Slika 2: LLL Algoritem

Skico dokaza algoritma iz slike 2 lahko bralec najde v [4] na straneh 445-446.

Poglavje 3

Kriptosistem NTRUEncrypt

3.1 Opis

Kriptosistem NTRUEncrypt je asimetrični kriptosistem, ki temelji na celoštevilskih rešetkah. Razvili so ga Jeffrey Hoffstein, Jill Pipher in Joseph H. Silverman leta 1996. V tem poglavju bomo predstavili kriptosistem NTRUEncrypt, kot je opisan v [4]. Najprej bomo predstavili glavne sestavine kriptosistema, nato pa še prevedbo na obliko z uporabo celoštevilskih rešetk.

3.2 Izbor parametrov

Kriptosistem NTRUEncrypt je odvisen od izbire treh celoštevilskih parametrov (N, p, q) in štirih množic polinomov stopnje $N - 1$, s celoštevilskimi koeficienti:

- N – določa polinomsko stopnjo $N - 1$,
- p – mali modul,
- q – veliki modul,
- $\mathcal{L}_f$ – množica privatnih polinomov f,
- $\mathcal{L}_g$ – množica začasnih polinomov g,
- $\mathcal{L}_\phi$ – množica slepilnih polinomov,
- $\mathcal{L}_m$ – množica sporočil v polinomski obliki

Števili N in p sta praštevili, število q pa jima je tuje (glej poglavje (3.6.5)) ter veliko večje od p (glej neenačbo (3.5)). Operacije potekajo v kolobarju $R = \mathbb{Z}[x]/(x^N - 1)$.
Ker se pri šifriranju in dešifriranju uporabljajo tudi kolobarji s koeficienti, skrčenimi po modulu p in q, definiramo še kolobarja

$$R_p = \frac{(\mathbb{Z}/p\mathbb{Z})[x]}{(x^N - 1)}, R_q = \frac{(\mathbb{Z}/q\mathbb{Z})[x]}{(x^N - 1)}.$$

Element $F \in R$ je zapisan kot polinom, ki ga predstavimo z vektorjem koeficientov

$$F = \sum_{i=0}^{N-1} F_i x^i = [F_0, F_1, \ldots, F_{N-1}].$$

Množica sporočil $\mathcal{L}_m$ vsebuje vse polinome stopnje največ $N-1$ s koeficienti iz $\mathbb{Z}_p$:

$$\mathcal{L}_m = \left\{ m \in R : \text{koeficienti } m \text{ ležijo med } -\frac{p-1}{2} \text{ in } \frac{p-1}{2} \right\}.$$

Ostale množice opišemo s pomočjo oznake

$$\mathcal{T}(d_1, d_2) = \left\{ F \in R : \begin{array}{c} d_1 \text{ koeficientov enakih } 1 \\ d_2 \text{ koeficientov enakih } -1 \\ \text{ostali enaki } 0 \end{array} \right\}$$

Izberemo tri pozitivna cela števila d_f, d_g, d_ϕ in vzamemo

$$\mathcal{L}_f = \mathcal{T}(d_f + 1, d_f),\ \mathcal{L}_g = \mathcal{T}(d_g, d_g) \text{ in } \mathcal{L}_\phi = \mathcal{T}(d_\phi, d_\phi).$$

Omejili se bomo na primer, ko velja $d := d_f = d_g = d_\phi$.

3.3 Generiranje ključev

3.3.1 Privatni ključ

Privatni del ključa predstavljata naključno izbrana polinoma $f, g \in R$ z majhnimi koeficienti (iz množice $\{-1, 0, 1\}$). Polinom f mora biti obrnljiv po modulu p in po modulu q. Inverza, ki ju izračunamo z razširjenim Evklidovim algoritmom, označimo s F_p in F_q

$$F_p * f \equiv 1 \pmod{p}, \tag{3.1}$$
$$F_q * f \equiv 1 \pmod{q}. \tag{3.2}$$

3.3.2 Javni ključ

Javni ključ h je, kot pove ime, objavljen javno in dostopen vsem, ki želijo šifrirati sporočilo:

$$h \equiv F_q * g \pmod{q}. \tag{3.3}$$

3.4 Šifriranje

Sporočilo $m \in \mathcal{L}_m$ je polinom s koeficienti, skrčenimi po modulu p. Pošiljatelj sporočila si izbere naključen slepilni polinom $\phi \in \mathcal{L}_\phi$ in uporabi javni ključ h za izračun šifriranega sporočila e

$$e \equiv p\phi * h + m \pmod{q}.$$

3.4.1 Prevod besedila v polinomsko obliko

Sporočilo $m = z_1 z_2 \dots z_n$, kjer so z_i znaki iz izvorne abecede $\sum = \{c_0, c_1, \dots, c_{t-1}\}$, je potrebno pred šifriranjem pretvoriti v zaporedje polinomov, ki jih nato šifriramo. To v primeru $p = 2k + 1$ storimo npr. takole:

1. Naj $s \in \mathbb{N}$ zadošča pogoju $p^{s-1} < t \leq p^s$. Vsak znak $z_i = c_j$ v sporočilu m zamenjamo s p-iškim zapisom števila j, na levi dopolnjenim z ničlami do dolžine s. Tako dobimo niz $m' \in \{0, 1, \dots, 2k\}^{ns}$.

2. Vsako p-iško števko v nizu m' zmanjšamo za k. Tako dobimo niz $m'' \in \{-k, -(k-1), \dots, k-1, k\}^{ns}$.

3. Niz m'' dopolnimo do dolžine, ki je večkratnik N, takole:
   Na koncu mu dodamo enojko, nato pa še toliko ničel (nič ali več), da dobimo niz m''' dolžine rN, kjer je $r \in \mathbb{N}$.

4. Niz m''' zapišemo v obliki $m''' = p_1 p_2 \dots p_r$, kjer so p_i podnizi dolžine N. Podniz $p_i = a_0 a_1 \dots a_{N-1}$ pretvorimo v polinom $p_i(x) = a_0 + a_1 x + \dots + a_{N-1} x^{N-1}$. Tako dobimo iskano zaporedje polinomov $p_1(x), p_2(x), \dots, p_r(x)$.

Primer prevoda črke "A" v polinomsko obliko na opisani način, če abecedo $\sum$ predstavlja prvih 128 ASCII znakov, $p = 3$ in $N = 5$:
Tu je $n = 1, t = 128, k = 1$ in $s = 5$.

1. ASCII vrednost črke "A" je $65_{(10)} = 2102_{(3)}$, zato je $m' = 02102$.

2. $m'' = \overline{1}10\overline{1}1$, kjer $\overline{1}$ predstavlja vrednost -1.

3. $m''' = \overline{1}10\overline{1}110000$, $r = 2$.

4. $p_1(x) = -1 + x - x^3 + x^4$, $p_2(x) = 1$.

3.5 Dešifriranje

Šifrirano sporočilo dešifriramo z uporabo privatnega ključa f. Najprej izračunamo

$$a(x) \equiv f(x) * e(x) \pmod{q}.$$

Nato $a(x)$ preuredimo tako, da so koeficienti na intervalu $(-q/2, q/2]$, ter izračunamo

$$b(x) \equiv F_p(x) * a(x) \pmod{p}. \tag{3.4}$$

Če so bili parametri pravilno izbrani, lahko preverimo, da je $b(x)$ enak šifriranemu čistopisu $m(x)$.

Izrek 3.1. *Če so parametri (N,p,q,d) sistema NTRUEncrypt izbrani tako, da velja*

$$q > (6d+1)p, \tag{3.5}$$

je polinom $b(x)$ iz enačbe (3.4) enak šifriranemu čistopisu $m(x)$.

Dokaz.

$$\begin{aligned} a &\equiv f * e \pmod{q} \\ &\equiv f * (p\phi * h + m) \pmod{q} \\ &\equiv f * p\phi * h + f * m \pmod{q} \\ &\equiv f * p\phi * F_q * g + f * m \pmod{q} \text{ iz (3.3)} \\ &\equiv p\phi * g + f * m \pmod{q}, \end{aligned} \tag{3.6}$$

kjer so koeficienti a na intervalu $(-q/2, q/2]$.
Poglejmo, kakšna je največja možna vrednost koeficientov polinoma a, če gledamo enačbo (3.6) v kolobarju R namesto v R_q. Polinoma $g(x)$ in $r(x)$ sta v $\mathcal{T}(d,d)$, zato je največji možni koeficient njunega zmnožka lahko $2d$. Podobno je $f(x) \in \mathcal{T}(d+1,d)$, koeficienti $m(x)$ pa so med $-\frac{1}{2}p$ in $\frac{1}{2}p$, torej največji možni koeficient produkta $f(x) * m(x)$ ne presega $(2d+1) \cdot \frac{1}{2}p$. Zato največji možni koeficient desne strani enačbe (3.6) v R ne presega

$$p \cdot 2d + (2d+1) \cdot \tfrac{1}{2}p = (3d + \tfrac{1}{2})p.$$

Enačba (3.5) nam zdaj zagotavlja, da je vsak koeficient v (3.6) strogo manjši od $\frac{1}{2}q$. Iz tega sledi, da pri računanju v R_q ne pride do izgub informacije, saj so koeficienti enaki tudi v R.
Dešifriranje končamo s tem, da izračunamo še

$$\begin{aligned} b &\equiv F_p * a \pmod{p} \\ &\equiv p\phi * g * F_p + F_p * f * m \pmod{p} \text{ iz (3.1)} \\ &\equiv m \pmod{p} \end{aligned} \tag{3.7}$$

in s tem dobimo originalno sporočilo m. □

3.5.1 NTRU kot kriptosistem, ki temelji na celoštevilskih rešetkah

Za javni ključ

$$h(x) = h_0 + h_1 x + \cdots + h_{N-1}x^{N-1}$$

definiramo pripadajočo $2N$-razsežno NTRU rešetko L_h^{NTRU} z bazno matriko

$$M_h^{NTRU} = \left(\begin{array}{cccc|cccc} 1 & 0 & \cdots & 0 & h_0 & h_1 & \cdots & h_{N-1} \\ 0 & 1 & \cdots & 0 & h_{N-1} & h_0 & \cdots & h_{N-2} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & h_1 & h_2 & \cdots & h_0 \\ \hline 0 & 0 & \cdots & 0 & q & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & 0 & q & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & q \end{array}\right).$$

Opazimo, da matrika sestoji iz štirih blokov velikosti $N \times N$, kjer je:

zgornji levi blok - identična matrika,

spodnji levi blok - matrika, s samimi ničlami,

zgornji desni blok - ciklične permutacije koeficientov $h(x)$,

spodnji desni blok - identična matrika, pomnožena s q.

Krajše

$$M_h^{NTRU} = \begin{pmatrix} I & H, \\ 0 & qI \end{pmatrix}, \tag{3.8}$$

kjer smo s H označili matriko velikosti $N \times N$, katere vrstice so ciklične permutacije koeficientov polinoma $h(x)$. Poljuben par polinomov

$$a(x) = a_0 + a_1 x + \cdots + a_{N-1}x^{N-1} \text{ in } b(x) = b_0 + b_1 x + \cdots + b_{N-1}x^{N-1}$$

v kolobarju R identificirajmo z $2N$-razsežnim vektorjem

$$(a, b) = (a_0, a_1, \ldots, a_{N-1}, b_0, b_1, \ldots, b_{N-1}) \in \mathbb{Z}^{2N}.$$

Izrek 3.2. *Naj bo $f(x) * h(x) \equiv g(x) \pmod q$ in naj bo $u(x) \in R$ polinom, za katerega velja*

$$f(x) * h(x) = g(x) + qu(x). \tag{3.9}$$

Potem je

$$(f, -u)M_h^{NTRU} = (f, g), \tag{3.10}$$

kar pomeni, da je vektor (f, g) v rešetki L_h^{NTRU}.

Dokaz. Z bločnim množenjem dobimo

$$(f, -u)\begin{pmatrix} I & H \\ 0 & qI \end{pmatrix} = (f, f * H - qu) = (f, g).$$

□

3.6 Varnost

3.6.1 Napad z grobo silo

Koeficienti javnega ključa so videti kot naključna cela števila po modulu q, vendar obstaja zveza

$$f(x) * h(x) \equiv g(x) \pmod{q}, \tag{3.11}$$

kjer imata $f(x)$ in $g(x)$ majhne koeficiente. Za dani $h(x)$ želi torej napadalec poiskati takšna polinoma s koeficienti iz množice {-1, 0, 1}, da velja enačba (3.12).
Iskanje ključa na takšen način nima samo ene rešitve, ker je za vsako rešitev $(f(x), g(x))$ tudi $(x^k * f(x), x^k * g(x)), 0 \leq k < N$, veljavna rešitev. Polinomu $x^k * f(x)$ pravimo *zasuk polinoma* $f(x)$, ker so koeficienti zasukani za k mest. Z dešifriranjem s takšnim polinomom dobimo zasukani čistopis $x^k * m(x)$.
Število možnih polinomov v $\mathcal{T}(d_1, d_2)$, kjer ima d_1 koeficientov vrednost 1, d_2 koeficientov vrednost -1, preostalih $N - d_1 - d_2$ pa 0, je:

$$\#\mathcal{T}(d_1, d_2) = \binom{N}{d_1}\binom{N - d_1}{d_2} = \frac{N!}{d_1!d_2!(N - d_1 - d_2)!}$$

Pri izčrpnem iskanju privatnega ključa mora napadalec preveriti vsak polinom v $\mathcal{T}(d + 1, d)$. A ker je vsak zasuk polinoma $f(x)$ tudi dešifrirni ključ, imamo N možnih rešitev, zato je potrebnih približno $\#\mathcal{T}(d+1, d)/N$ poskusov, da najdemo enega izmed zasukov $f(x)$.

Zgled:
Če izberemo NTRU parametre

$$(N, p, q, d) = (251, 3, 257, 83),$$

je tako potrebno preiskati približno

$$\frac{\#\mathcal{T}(84, 83)}{251} = \frac{1}{251}\binom{251}{84}\binom{167}{83} \approx 2^{381.6}$$

polinomov, preden najdemo dešifrirni ključ.

3.6.2 Napad s srečanjem na sredini

Howgrave-Graham, Silverman in Whyte v svojem članku [5] opisujejo napad s srečanjem na sredini (meet-in-the-middle attack). Za lažjo predstavitev napada predpostavimo, da velja:

- N je sodo število,
- d je sodo število,
- $f(x)$ in $g(x)$ imata dvojiške koeficiente z d enicami: $f(x), g(x) \in \mathcal{T}(d, 0)$,
- k je takšno celo število, ki ga izbere napadalec, da je 2^k večje od $\binom{N/2}{d/2}$ (recimo za faktor 100).

Ideja je poiskati takšen $f(x) = f_1(x) + f_2(x) \in \mathcal{T}(d, 0)$, da imata polinoma

$$f_1(x) = \sum_{0 \le i < N/2} a_i x^i \text{ in } f_2(x) = \sum_{N/2 \le i < N} a_i x^i$$

vsak po $d/2$ koeficientov enakih 1. Velja:

$$\begin{aligned} f(x) * h(x) &= g(x) && (\text{mod } q) \\ \Rightarrow (f_1(x) + f_2(x)) * h(x) &= g(x) && (\text{mod } q) \\ \Rightarrow f_1(x) * h(x) &= g(x) - f_2(x) * h(x) && (\text{mod } q) \\ \Rightarrow (f_1(x) * h(x))_i &= \{0, 1\} - (f_2(x) * h(x))_i \text{ za vsak } i && (\text{mod } q) \end{aligned} \tag{3.12}$$

kjer zapis u_i pomeni i-ti koeficient polinoma $u(x)$. Napadalec izvede naslednje korake:

- Najprej pregledamo vse možne polinome $f_1(x)$ dolžine $N/2$, ki imajo $d/2$ enic. Za to potrebujemo $\binom{N/2}{d/2}$ korakov. Vsak $f_1(x)$ vstavimo v "predalček", katerega naslov sestavljajo vodilni biti prvih k koeficientov polinoma $f_1(x) * h(x) \pmod q$. Od 2^k možnih predalčkov jih bo približno $\binom{N/2}{d/2}$ zasedenih.
- Nato pregledamo vse možne polinome $f_2(x)$, za kar prav tako potrebujemo $\binom{N/2}{d/2}$ korakov. Pri tem polinom $f_2(x)$ pripada vsem tistim predalčkom, ki jih določajo vodilni biti prvih k koeficientov množice polinomov, ki jih dobimo, če nekaterim koeficientom polinoma $-f_2(x) * h(x)$ prištejemo 1 (mod q). Če $f_2(x)$ pripada kateremu od zasedenih predalčkov, rečemo, da je prišlo do trka. V tem primeru smo našli polinoma $f_1(x)$ in $f_2(x)$, za katera velja enačba (3.12). Če polinom

$(f_1(x) + f_2(x)) * h(x) \pmod q$ pripada $\mathcal{T}(d, 0)$, vrnemo $f(x) = f_1(x) + f_2(x)$, sicer nadaljujemo z naslednjim $f_2(x)$. Če zasedeni predalček vsebuje več kandidatov za $f_1(x)$, opisani postopek opravimo z vsemi.

Zgled ($N = 4, q = 8, k = 4$):

- če $f_1(x) * h(x) \pmod q = [7, 2, 3, 5]$ (oziroma, če števila zapišemo dvojiško $[\mathbf{1}11, \mathbf{0}10, \mathbf{0}11, \mathbf{1}01]$), potem je $f_1(x)$ shranjen v predalček z naslovom [1001].
- če $-f_2(x) * h(x) \pmod q = [6, 2, 1, 5]$ (dvojiško [**1**10, **0**10, **0**01, **1**01]), potem $f_2(x)$ pripada predalčku [1001].
- če $-f_2(x) * h(x) \pmod q = [7, 2, 3, 5]$ (dvojiško [**1**11, **0**10, **0**11, **1**01]), potem $f_2(x)$ pripada predalčkom [1001], [0001], [1011], [0011], saj se številoma 7 in 3 v primeru, da jima prištejemo 1, spremeni vodilni bit.

Število potrebnih korakov za pridobitev ključa pri algoritmih, ki uporabljajo trke, dobimo tako, da korenimo število potrebnih iskanj z grobo silo. Pri zgledu iz prejšnjega poglavja bi tako z uporabo algoritma s srečanjem na sredini potrebovali $\sqrt{2^{381.6}} \approx 2^{190.8}$ korakov.

3.6.3 Napad z izbranimi tajnopisi

Kot smo že predpostavili v prejšnjih poglavjih, morata biti modula p in q izbrana tako, da zadoščata neenačbi (3.5), sicer se lahko zgodi, da pri dešifriranju ne dobimo originalnega sporočila m. Priporočeni parametri so se od prve verzije kriptosistema leta 1998 [2] preko dodatnih optimizacij leta 2000 [3], ki so opisane v poglavju 4, do standardizacije leta 2003 [7] spreminjali. Priporočeni parametri po standardu IEEE P1363.1 ne omogočajo več dešifrirnih napak, zato na tem mestu bralca napotimo h kriptoanalizi napada z izbranimi tajnopisi, ki je opisana v [1].

3.6.4 Napadi na rešetke

Če ima rešetka L pravokotno bazo, potem je zelo lahko rešiti problema SVP in CVP. Algoritem LLL, opisan v 2.3.2, sicer ne vrne pravokotne baze, vendar dobimo bazo, v kateri so vektorji dokaj pravokotni med seboj. Če združimo algoritem LLL z Babai-jevim algoritmom (poglavje 2.2.1), dobimo približno rešitev problema CVP.

3.6.5 Priporočeni izbor parametrov

Tabela 3.2 prikazuje priporočeni izbor parametrov po standardu IEEE P1363.1. Parametri so bili izračunani tako, da so varni proti trenutno najboljšim in najhitrejšim znanim napadom. Priporočljivo je tudi, da sta modula p in q izbrana tako, kot prikazuje tabela 3.1.

p	q
2	praštevilo
3	2^m
$2+x$	2^m

Tabela 3.1: Izbor p[1] in q

	n	**p**	**q**	$\mathbf{d}_f$	$\mathbf{d}_g$	$\mathbf{d}_m$	$\mathbf{d}_\phi$	**Stopnja varnosti**
ees401ep1	401	3	2048	113	133	113	113	112
ees541ep1	541	3	2048	49	180	49	49	112
ees659ep1	659	3	2048	38	219	38	38	112
ees449ep1	449	3	2048	134	149	134	134	128
ees613ep1	613	3	2048	55	204	55	55	128
ees761ep1	761	3	2048	42	253	42	42	192
ees653ep1	653	3	2048	194	217	194	134	192
ees887ep1	887	3	2048	81	295	81	81	192
ees1087ep1	1087	3	2048	63	362	63	63	192
ees853ep1	853	3	2048	268	289	268	268	256
ees1171ep1	1171	3	2048	106	390	106	106	256
ees1449ep1	1499	3	2048	79	499	79	79	256

Tabela 3.2: Priporočljivi parametri po standardu IEEE P1363.1

3.7 Primer

Za primer smo izbrali parametre
$(N, p, q, d) = (11, 3, 32, 3)$
Privatni del ključa, naključno izbrana f in g:

$$f = -1 + x - x^3 + x^5 + x^8 + x^9 - x^{10}$$
$$g = -1 + x + x^6 - x^8 - x^9 + x^{10}$$

[1]Opis izbora parametra p kot polinom $2 + x$ lahko bralec najde v [9]

Izračunamo inverza f tako, da velja

$$F_p * f \equiv 1 \pmod{p},$$
$$F_q * f \equiv 1 \pmod{q}.$$

$F_p \equiv 2 + x + 2x^2 + 2x^3 + x^4 + x^5 + x^6 + 2x^7 + 2x^8 + x^9 + x^{10} \pmod 3$
$F_q \equiv 23x + x^2 + 14x^3 + 2x^4 + 28x^5 + 3x^6 + 24x^7 + 6x^8 + 16x^9 + 12x^{10} \pmod{32}$

Izračunamo javni ključ h:

$$h \equiv F_q * g \pmod q$$
$$h \equiv 16 + 29x + 30x^2 + 28x^3 + 29x^4 + 23x^5 + 27x^6 + 12x^7 + \\ + 23x^8 + 25x^9 + 14x^{10} \pmod{32}$$

Šifrirati želimo sporočilo $m =$ "AB".
ASCII vrednosti za "A" $= 65_{(10)}$ in "B" $= 66_{(10)}$.
Števili pretvorimo v trojiški sistem:

$$65_{(10)} = 2102_{(3)} = 02102_{(3)}$$
$$66_{(10)} = 2110_{(3)} = 02110_{(3)}$$

Dobljena trojiška zapisa staknemo in od posameznih števk odštejemo 1:

$$0210202110 \to \overline{1}10\overline{1}1\overline{1}100\overline{1},$$

kjer $\overline{1}$ predstavlja vrednost -1.
Sporočilo "AB" v polinomski obliki:

$$m = -1 + x - x^3 + x^4 - x^5 + x^6 - x^9$$

Slepilni polinom:

$$\phi = -1 + x + x^3 - x^4 - x^6 + x^{10}$$

Šifriranje:

$e \equiv p\phi * h + m \pmod q$
$e \equiv 12 + 9x + 12x^2 - 3x^3 - x^4 - 5x^5 + 3x^6 - 2x^7 - 2x^8 - 5x^9 + 13x^{10} \pmod{32}$

Dešifriranje:

$a \equiv f * e \pmod q$
$a \equiv 6 - 10x + 11x^2 - 3x^3 + 8x^4 - 3x^5 - x^6 - 4x^7 + 7x^9 - 12x^{10} \pmod{32}$

$b \equiv a * F_p \pmod p$
$b \equiv -1 + x - x^3 + x^4 - x^5 + x^6 - x^9 \pmod 3$

S tem dobimo prvotno sporočilo m.

Poglavje 4

Pohitritve

V tem poglavju so opisane izboljšave, ki še dodatno doprinesejo k hitrosti šifriranja in dešifriranja.

4.1 Izbira privatnega polinoma

Največ časa pri šifriranju in dešifriranju zahteva množenje polinomov v kolobarju R. Pri šifriranju je potrebno izračunati produkt $\phi * h \pmod q$, pri dešifriranju pa produkta $f * e \pmod q$ in $F_p * a \pmod p$. Podobno je najdražja časovna operacija pri generaciji ključev računanje inverzov F_p in F_q. Za pohitritev kreacije ključev in dešifriranja pri Security Innovations Inc. predlagajo [3], da se privatni ključ f izbere kot $f = 1 + p \cdot F \in R$.
Če izberemo f v tej obliki, potem:

$$f = 1 + p * F \equiv 1 \pmod p$$

$$F_p \equiv 1 \pmod p$$

Izračun inverza po modulu p tako ni več potreben.
Prav tako pri dešifriranju izračun drugega produkta ni več potreben:

$$\begin{aligned}
a &\equiv f * e \pmod q \\
&\equiv p \cdot \phi * g + f * m \pmod q \text{ iz (3.6)} \\
&\equiv p \cdot \phi * g + (1 + p \cdot F) * m \pmod q \\
b &\equiv a \pmod p \\
&\equiv p \cdot \phi * g + (1 + p \cdot F) * m \pmod p \\
&\equiv p \cdot (\phi * g + F * m) + m \pmod p \\
&\equiv m \pmod p
\end{aligned}$$

Poglavje 5

Implementacija

V tem poglavju so opisani nekateri algoritmi, ki so bili uporabljeni pri implementaciji kriptosistema NTRUEncrypt. Za implementacijo kriptosistema smo se odločili uporabiti programski jezik Java. Izvorna koda se nahaja na priloženem CD-ju.

5.1 Algoritem za hitro množenje polinomov

Koliko osnovnih aritmetičnih operacij (množenj in seštevanj) potrebujemo za izračun produkta dveh polinomov stopnje $n-1$? Naj bo $a(x) = \sum_{k=0}^{2n-1} a_k x^k$ produkt polinomov $b(x) = \sum_{i=0}^{n-1} b_i x^i$ in $c(x) = \sum_{j=0}^{n-1} c_j x^j$. Potem je:

$$a(x) = \sum_{i=0}^{n-1}\sum_{j=0}^{n-1} b_i c_j x^{i+j} = \sum_{k=0}^{2n-2} x^k \sum_{i=max(0,k-n+1)}^{min(k,n-1)} b_i c_{k-i}, \tag{5.1}$$

torej je

$$a_k = \begin{cases} \sum_{i=0}^{k} b_i c_{k-i}, \text{ za } k = 0, 1, \ldots, n-1, \\ \sum_{i=k-n+1}^{n-1} b_i c_{k-i}, \text{ za } k = n, n+1, 2n-2. \end{cases}$$

Pri običajnem algoritmu za "dolgo množenje" polinomov računamo koeficiente a_k po formuli (5.1) in opravimo

$$\sum_{k=0}^{n-1}(k+1) + \sum_{k=n}^{2n-2}(2n-k-1) = n^2 \text{ množenj in}$$

$$\sum_{k=0}^{n-1} k + \sum_{k=n}^{2n-2} (2n-k-2) = (n-1)^2 \text{ seštevanj.}$$

V tehničnem poročilu [13] je bilo pokazano, da je z razbitjem posameznega polinoma na dva dela mogoče število operacij zmanjšati na $\frac{3}{4}n^2$. Z uporabo rekurzije se da to število še zmanjšati.
Ideja v razbitju polinomov je v tem, da zapišemo b in c kot vsoti. Naj bosta $n_1 = \lfloor \frac{n}{2} \rfloor$ in $n_2 = \lceil \frac{n}{2} \rceil$:

$$b = b_1 + b_2 x^{n_1},$$
$$c = c_1 + c_2 x^{n_1},$$

kjer je

$$deg(b_1) = deg(c_1) = n_1 - 1,$$
$$deg(b_2) = deg(c_2) = n_2 - 1.$$

Produkt bc potem zapišemo kot:

$$bc = b_1c_1 + (b_1c_2 + b_2c_1)x^{n_1} + b_2c_2x^{2n_1} \tag{5.2}$$

Potrebno je izračunati štiri zmnožke $b_1c_1, b_1c_2, b_2c_1, b_2c_2$ polinomov stopnje približno $\frac{n}{2}$, kar zahteva $4 \cdot (\frac{n}{2})^2 = n^2$ operacij, s čimer ne pridobimo ničesar. Srednji koeficient pa lahko zapišemo drugače:

$$b_1c_2 + b_2c_1 = (b_1 + b_2)(c_1 + c_2) - b_1c_1 - b_2c_2.$$

Ker imamo b_1c_1 in b_2c_2 v (5.2) že izračunana, s tem zmanjšamo število množenj s štiri na tri. Na račun nekaj dodatnih seštevanj potrebujemo za izračun le $3 \cdot (\frac{n}{2})^2 = \frac{3}{4}n^2$ operacij.
Če ta proces izvedemo rekurzivno r-krat, se število operacij dodatno zmanjša na $(\frac{3}{4})^r n^2$.

```
1  Function multiplyRecursive(b(x), c(x), int n, int N)
       /* Množimo polinoma b(x) in c(x) stopnje n − 1.
       Proceduro kličemo rekurzivno, dokler stopnji nista
       manjši od določene meje cutOff (naša izbrana meja je
       32).                                                  */
2      if n ≤ cutOff then
3          for (k = 0; k ≤ 2 ∗ n − 2; k++) do
4              a[k] = 0 for (i = max(0, k − n + 1); i ≤ min(k, n − 1);
                i++) do
5                  a[k] += b[i] ∗ c[k − i]
6              end
7          end
8      else
9          n1 = n/2;
10         n2 = n − n1;
11         b_1 = b mod x^{n_1};
12         b_2 = b div x^{n_1};
13         c_1 = c mod x^{n_1};
14         c_2 = c div x^{n_1};
15         B = b1 + b2;
16         C = c1 + c2;
           /* B in C sta stopnje n/2                        */
17         a1 = multiplyRecursive(b1, c1, n1, N);
18         a2 = multiplyRecursive(b2, c2, n2, N);
19         a3 = multiplyRecursive(B, C, n2, N);
20         a = a1 + (a3 − a1 − a2) ∗ x^{n1} + a2 ∗ x^{2∗n1};
21     end
       /* Stopnja a(x) je 2 ∗ n − 1. Če 2 ∗ n − 1 > N, potem
       uporabimo relacijo x^N = 1                            */
22     if 2 ∗ n − 1 > N AND N > 0 then
23         for (k = N; k < 2 ∗ n − 1; k++) do
24             a[k − N] += a[k];
25         end
26     end
27     return a(x)
28 end
```

Slika 3: Algoritem za hitro množenje polinomov

5.2 Inverz polinoma

Privatni del ključa dobimo z računanjem inverza naključno izbranega polinoma f po modulu p in q. V ta namen smo, po priporočilih Security Innovations Inc. [11], uporabili algoritem "Almost Inverses"[8] (Algoritem 4), ki je modificiran razširjeni Evklidov algoritem za iskanje inverzov.
Za vhodni parameter $a(x) \in (\mathbb{Z}/p\mathbb{Z})[x]/(m(x))$, če velja $\gcd(a(x), m(x)) = 1$, $m(0) = 1$ in p praštevilo, algoritem vrne inverz $a(x)^{-1} \in (\mathbb{Z}/p\mathbb{Z})[x]/(m(x))$.
Če poznamo inverz polinoma po modulu p, lahko s preprosto metodo, ki temelji na Newtonovi iterativni metodi, dobimo inverz polinoma po modulu p^r (Algoritem 5).

vhod : $a(x), p$ **(praštevilo)**
izhod : $b(x) \equiv a(x)^{-1} \in (\mathbb{Z}/p\mathbb{Z})[x]/(x^N - 1)$
$k := 0$, $b(x) := 1$, $c(x) := 0$, $f(x) := a(x)$, $g(x) := x^N - 1$
Loop:
while $f_0 = 0$ **do**
    $f(x) := f(x)/x$, $c(x) := c(x) * x$, $k := k + 1$
end
if $\deg(f) = 0$ **then**
    $b(x) := f_0^{-1} \cdot b(x) \pmod p$
    return $x^{N-k} \cdot b(x) \pmod{x^N - 1}$
end
if $deg(f) < deg(g)$ **then**
    zamenjaj f **in** g **ter** b **in** c
end
$u := f_0 \cdot g_0^{-1} \pmod p$
$f(x) := f(x) - u * g(x) \pmod p$
$b(x) := b(x) - u * c(x) \pmod p$
goto Loop

Slika 4: "Almost Inverses" v kolobarju $\mathbb{Z}[x]/(x^N - 1)$

vhod : $a(x)$, p **(praštevilo)**, r, $b(x) \equiv a(x)^{-1} \pmod p$
izhod : $b(x) \equiv a(x)^{-1} \pmod{p^r}$
$q = p$
while $q < p^r$ **do**
    $q = q * p$
    $b(x) := b(x)(2 - a(x)b(x)) \pmod q$
end
return $b(x)$

Slika 5: Inverz polinoma po modulu p^r

V posebnem primeru, ko je $p = 3$, lahko algoritem 4 še pohitrimo tako, kot je to prikazano v algoritmu 6.

vhod : $a(x)$
izhod : $b(x) \equiv a(x)^{-1} \in (\mathbb{Z}/3\mathbb{Z})[x]/(x^N - 1)$
$k := 0, b(x) := 1, c(x) := 0, f(x) := a(x), g(x) := x^N - 1$
Loop:
while $f_0 = 0$ **do**
    $f(x) := f(x)/x$, $c(x) := c(x) * x$, $k := k + 1$
end
if $f(x) = \pm 1$ **then**
    return $\pm x^{N-k} b(x) \pmod{x^N - 1}$
end
if $deg(f) < deg(g)$ **then**
    zamenjaj f **in** g **ter** b **in** c
end
if $f_0 = g_0$ **then**
    $f(x) := f(x) - g(x) \pmod 3$
    $b(x) := b(x) - c(x) \pmod 3$
end
else
    $f(x) := f(x) + g(x) \pmod 3$
    $b(x) := b(x) + c(x) \pmod 3$
end
goto Loop

Slika 6: "Almost Inverses" v kolobarju $(\mathbb{Z}/3\mathbb{Z})[x]/(x^N - 1)$

Poglavje 6

Zaključek

Parametri, priporočeni v prvi predstavitvi kriptosistema NTRUEncrypt [2], so omogočali možnost, da dešifriranje ne uspe. Avtorji so najprej predlagali, da to možnost zanemarimo, ker v praksi le redko nastopi. Kasneje se je izkazalo, da lahko neuspešna dešifriranja izkoristimo za napad z izbranimi tajnopisi. Ta pomanjkljivost je bila odpravljena v zadnji verziji, ki je tudi standardizirana kot IEEE P1363.1. Kriptosistem je s skrbno izbranimi parametri odporen tudi proti napadu z grobo silo in proti napadom s srečanjem v sredini. Tako v teoriji kot v praksi je kriptosistem NTRUEncrypt za isto stopnjo varnosti mnogo hitrejši kot trenutno uveljavljeni asimetrični sistemi (RSA, Elgamal, ...).

Literatura

[1] N. Gama, P. Q. Nguyen, *New Chosen-Ciphertext Attacks on NTRU*, Paris: Ecole normale superieure, 2007.

[2] J. Hoffstein, J. Pipher, J. H. Silverman, NTRU: A Ring-Based Public Key Cryptosystem, *Algorithmic Number Theory: Third International Symposium*, Portland, Oregon, 21. - 25. junij, 1998, str. 267–288.

[3] J. Hoffstein, J. Silverman, Optimization for NTRU, *Public-Key Cryptography and Computational Number Theory*, Warsaw: Stefan Banach International Mathematical Center, 2000, str. 77–88.

[4] J. Hoffstein, J. Pipher, J. H. Silverman, *An Introduction to Mathematical Cryptography, Second Edition*, New York: Springer Science+Business Media, LLC, 2014.

[5] N. Howgrave-Graham, J. H. Silverman, W. Whyte, A Meet-in-the-Middle Attack on an NTRU Private Key, *NTRU Cryptosystems Technical Report 4, Version 2*.
`https://assets.securityinnovation.com/static/downloads/NTRU/resources/NTRUTech004v2.pdf` (dostop julij 2016)

[6] J. Proos, C. Zalka, *Shor's discrete logarithm quantum algorithm for elliptic curves*.
Quantum Inf. Comput. 3(4), 2003, str. 317–344.

[7] IEEE, *P1363.1 Public-Key Cryptographic Techniques Based on Hard Problems over Lattices*.
`http://grouper.ieee.org/groups/1363/lattPK/index.html` (dostop julij 2016)

[8] R. Schroeppel, H. Orman, S. O'Malley, and O. Spatscheck, Fast key exchange with elliptic curve systems. *Advances in Cryptology – CRYPTO '95, Lecture Notes in Computer Science*, Berlin, Springer-Verlag, 1995, str. 43–56.

[9] Security Innovation Inc., *NTRU Enhancements 1.*
`https://assets.securityinnovation.com/static/downloads/NTRU/resources/NTRU-Enhancements-1.pdf` (dostop julij 2016)

[10] P. W. Shor. *Algorithms for quantum computation: discrete logarithms and factoring.*
Santa Fe: IEEE Comput. Soc. Press, Los Alamitos, 1994, str.124–134.

[11] J. H. Silverman, Almost Inverses and Fast NTRU Key Generation, *NTRU Cryptosystems Technical Report 14.*
`https://assets.securityinnovation.com/static/downloads/NTRU/resources/NTRUTech014.pdf` (dostop julij 2016)

[12] J. H. Silverman, Invertibility in Truncated Polynomial Rings, *NTRU Cryptosystems Technical Report 9.*
`https://assets.securityinnovation.com/static/downloads/NTRU/resources/NTRUTech009.pdf` (dostop julij 2016)

[13] J. H. Silverman, High-Speed2 Multiplication of (Truncated) Polynomials, *NTRU Cryptosystems Technical Report 10.*
`https://assets.securityinnovation.com/static/downloads/NTRU/resources/NTRUTech010.pdf` (dostop julij 2016)

Stvarno kazalo

algoritem
  Babai-jev, 6
  Evklidov, 14
  Gaussov, 7
  LLL, 9

ključ
  javen, 15
  privaten, 14

LLL-reducirana baza, 10
Lovászov pogoj, 10

napad
  na rešetke, 20
  s srečanjem na sredini, 19
  z grobo silo, 18
  z izbranimi tajnopisi, 20
NTRUEncrypt
  parametri, 13

pogoj velikosti, 10
polinom
  privaten, 14
  slepilen, 15
  začasen, 14

rešetka
  celoštevilska, 3
  CVP, 6
  dimenzija, 3
  NTRU, 17
  rang, 3
  SVP, 6

sporočilo, 15
  translacija, 15

temeljna domena, 4

zasuk polinoma, 18