

**SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY**

Evidenčné číslo: FEI-5384-8739

**PODPISOVÉ SCHÉMY V POSTKVANTOVEJ
KRYPTOGRAFII
DIPLOMOVÁ PRÁCA**

2016

Pavol Dobročka

**SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY**

Evidenčné číslo: FEI-5384-8739

**PODPISOVÉ SCHÉMY V POSTKVANTOVEJ
KRYPTOGRAFII
DIPLOMOVÁ PRÁCA**

Študijný program: Aplikovaná informatika
Číslo študijného odboru: 2511
Názov študijného odboru: 9.2.9 Aplikovaná informatika
Školiace pracovisko: Ústav informatiky a matematiky
Vedúci záverečnej práce: doc. Ing. Pavol Zajac, PhD.

Bratislava 2016

Pavol Dobročka

SÚHRN

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Študijný program:	Aplikovaná informatika
Autor:	Pavol Dobročka
Diplomová práca:	Podpisové schémy v postkvantovej kryptografii
Vedúci záverečnej práce:	doc. Ing. Pavol Zajac, PhD.
Miesto a rok predloženia práce:	Bratislava 2016

Abstract SK

Kľúčové slová:

ABSTRACT

SLOVAK UNIVERSITY OF TECHNOLOGY IN BRATISLAVA

FACULTY OF ELECTRICAL ENGINEERING AND INFORMATION TECHNOLOGY

Study Programme:	Applied Informatics
Author:	Pavol Dobročka
Diploma Thesis:	Signature schemas in postquantum cryptography
Supervisor:	doc. Ing. Pavol Zajac, PhD.
Place and year of submission:	Bratislava 2016

Abstract EN

Keywords:

Vyhlásenie autora

Podpísaný Pavol Dobročka čestne vyhlasujem, že som diplomovú prácu Podpisové schémy v postkvantovej kryptografii vypracoval na základe poznatkov získaných počas štúdia a informácií z dostupnej literatúry uvedenej v práci.

Vedúcim mojej diplomovej práce bol doc. Ing. Pavol Zajac, PhD.

Bratislava, dňa 1.5.2016

.....
podpis autora

Pod'akovanie

I would like to express a gratitude to my thesis supervisor.

Obsah

Úvod	11
1 Úvod do postkvantovej kryptografie	12
1.1 Motivácia	12
1.2 Súčasný stav	12
2 Code-based kryptografia	13
2.1 Úvod	13
2.2 McEliece a Niederreiter	13
2.2.1 Porovnanie McEliece a Niederreiter	14
3 Code-based podpisové schémy	16
3.1 Úvod	16
3.2 Prehľad code-based podpisových schém	16
3.3 CFS schéma	17
3.4 LDGM podpis	18
3.4.1 Ukážkový príklad	19
3.5 Porovnanie	23
4 Návrh a implementácia LDGM schémy	24
4.1 Parametre	24
4.2 Funkcia ϕ	24
4.3 Funkcia ψ	24
4.4 Generovanie kľúčov	26
4.5 Invertovanie QC matice	27
4.6 Generovanie matice G	30
4.7 Generovanie matice Q	30
4.8 Generovanie matice S	31
4.9 Výpočet matice H_{pub}	31
4.10 Implementácia v BitPunch	32
5 Výsledky meraní	34
5.1 Invertovanie QC matíc	34
5.2 Výkonnosť LDGM	38
Záver	40

Resumé	41
Zoznam použitej literatúry	I
Prílohy	I

Zoznam obrázkov a tabuliek

Obrázok 1	Diagram štruktúr v BitPunch	32
Obrázok 2	Organizačná štruktúra v BitPunch	33
Obrázok 3	Graf závislosti výpočtového času od veľkosti cirkulatných blokov .	34
Obrázok 4	Graf závislosti výpočtového času od počtu blokov v matici	35
Obrázok 5	Graf závislosti výpočtového času od počtu blokov a veľkosti blokov	36
Obrázok 6	Graf závislosti úspešnosti od počtu blokov	37
Obrázok 7	Graf závislosti času generovania od počtu veľkosti blokov	39
Tabuľka 1	Porovnanie parametrov McEliece a Niederreiter	15
Tabuľka 2	Porovnanie CFS a LDGM schémy	23
Tabuľka 3	Parametre a funkcie LDGM	24
Tabuľka 4	Výsledky časovej zložitosti invertovania QC matíc	35
Tabuľka 5	Výsledky pre najúspešnejšie veľkosti blokov	37
Tabuľka 6	Výsledky pre najneúspešnejšie veľkosti blokov	38
Tabuľka 7	Výsledky merní pre LDGM implementáciu	39

Zoznam skratiek a značiek

WWW - World Wide Web

Zoznam algoritmov

1	McEliece - Algoritmus šifrovania	13
2	McEliece - Algoritmus dešifrovania	14
3	Niederreiter - Algoritmus šifrovania	14
4	Niederreiter - Algoritmus dešifrovania	14
5	Schéma digitálneho podpisu	16
6	Algoritmus podpisovania v CFS	17
7	Algoritmus overovania v CFS	18
8	Výpočet matice Q	19
9	Podpis v LDGM	20
10	Overenie v LDGM	21
11	Funkcia ϕ	25
12	Funkcia ψ	26
13	Generovanie kľúčov	27
14	Invertovanie QC matice	28
15	Generovanie matice G	30
16	Generovanie QC matice s pevnou váhou	31

Úvod

Uvod SK

1 Úvod do postkvantovej kryptografie

1.1 Motivácia

Preco skumat postkvantovu kryptografiu, aku vyhodu ponukaju...

1.2 Súčasný stav

Popis ako sa momentalne používajú, či sa používajú (aký majú podiel).

Velmi stručny opis (prípadne len vymenovanie) toho aké rôzne postkvantové kryptosystémy pozname (hash-based, code-based..)

2 Code-based kryptografia

2.1 Úvod

Jednu triedu z postkvantových kryptosystémov tvoria code-based systémy, teda kryptosystémy vychádzajúce z teórie kódovania. Bezpečnosť takýchto systémov je založená na zložitosti takzvaného dekódovacieho problému. V súčasnosti nie je známy algoritmus, ktorý by efektívne dekodoval ľubovoľný kód ako na klasickom, tak aj na kvantovom počítači. Existujú však triedy lineárnych kódov, ktoré efektívne dekódovať vieme. Tento poznatok má kryptografické využitie. Podstata code-based kryptosystémov je skonštruovať kód, ktorý vieme dekódovať a následne tento kód zmodifikovať na kód, ktorý dekódovať nevieme bez toho, aby sme poznali "inverznú" modifikáciu.

2.2 McEliece a Niederreiter

Najstarším a pravdepodobne najznámejším code-based kryptosystémom je McEliecov kryptosystém. Jadro systému tvorí kód C dĺžky n s dimenziou k a minimálnou vzdialenosťou $d \geq 2t + 1$, kde t je počet chýb, ktorý vie kód opraviť. Podľa pôvodného návrhu sa používajú Goppove kódy, ku ktorým existuje efektívny dekódovací algoritmus.

Verejný a súkromný kľúč zostrojíme nasledovne. Určíme generujúcu maticu G s rozmermi $k \times n$ pre kód C . Ďalej zvolíme náhodnú binárnu regulárnu maticu S s rozmermi $k \times k$ a permutačnú maticu P s rozmermi $n \times n$. Verejný kľúč tvorí matica $G' = SG P$ a parameter t . Súkromný kľúč tvoria matice S, G, P .

Algoritmus 1 McEliece - Algoritmus šifrovania

Vstup: Správa m dĺžky k

Výstup: Zašifrovaná správa c

$c' \leftarrow mG'$

Ku zakódovanej správe pripočítame náhodný chybový vektor s váhou t .

$c \leftarrow c' + e, wt(e) = t$

return c

Pre praktickú bezpečnosť sa hodnoty parametrov kódu volia približne $n = 1000, k = 500, t = 50$.

K McEliecovmu kryptosystému existuje varianta, ktorá namiesto generujúcej matice G využíva kontrolnú maticu H . Táto duálna forma sa označuje ako Niederreiterov kryptosystém. V tomto kryptosystéme sa správa m najskôr transformuje na vektor m' dĺžky n s Hammingovou váhou t . Funkciu, ktorá vykonáva túto transformáciu označujeme

Algoritmus 2 McEliece - Algoritmus dešifrovania

Vstup: Zašifrovaná správa c

Výstup: Otvorená správa m

Správu c vynásobíme s P^{-1}

$$c' \leftarrow cP^{-1} = mSG + eP^{-1}$$

$$m' \leftarrow \text{Decode}(c') = mS;$$

$$m \leftarrow m'S^{-1}$$

return m

$\phi_{n,t}(m)$. Verejný kľúč tvorí matica $H' = SHP$ a parameter t . Matica S je nahodná regulárna binárna matica s rozmermi $(n - k) \times (n - k)$ a P je permutačná matica s rozmermi $n \times n$ a súkromný kľúč tvoria matice S, H, P . Šifrovaný text sa vypočíta ako syndróm slova m' , $c = H'm'^T$. Na dešifrovanie slova c vlastník súkromného kľúča najskôr vynásobi slovo c maticou S^{-1} zľava, následne aplikuje dekódovací algoritmus a výsledok vynásobí maticou P^{-1} zľava. $m = P^{-1}\text{decode}(S^{-1}SHPm)$

Algoritmus 3 Niederreiter - Algoritmus šifrovania

Vstup: Správa m

Výstup: Zašifrovaná správa c

$m' \leftarrow \phi(m)$, dostaneme chybové slovo dĺžky n s váhou t

$$c \leftarrow H'm'^T$$

return c

Algoritmus 4 Niederreiter - Algoritmus dešifrovania

Vstup: Vektor c , ktorý predstavuje šifrovanú správu m

Výstup: Dešifrovaná správa, pôvodné m

$$c' \leftarrow S^{-1}c$$

$$e' \leftarrow \text{Decode}(c')$$

$$e \leftarrow P^{-1}e'$$

$$m \leftarrow \phi^{-1}(e)$$

2.2.1 Porovnanie McEliece a Niederreiter

Zhrňme si a porovnajme parametre oboch kryptosystémov a ako sa zvolené parametre kódu prejaví na veľkosti správ a kľúčov.

Tabuľka 1: Porovnanie parametrov McEliece a Niederreiter

	McEliece	Niederreiter
Verejný kľúč	G', t	H', t
Privátny kľúč	S, G, P	S, H, P
Veľkosť VK	$k \times n$	$n - k \times n$
Veľkosť PK	$k \times k, k \times n, n \times n$	$n - k \times n - k, n - k \times n, n \times n$
Veľkosť otvorenej správy	k	n
Veľkosť šifrovanej správy	n	$n - k$
Počet možných správ	2^k	$\sum_{i=0}^t \binom{n}{i}$

3 Code-based podpisové schémy

3.1 Úvod

Prechod na postkvantovú kryptografiu so sebou prináša aj potrebu implementovať podpisové schémy pomocou postkvantového kryptosystému. Vo všeobecnosti sa na realizáciu digitálneho podpisu využívajú asymetrické kryptosystémy, respektíve kryptosystémy s verejným kľúčom. Kryptosystémy, ktoré sme si predstavili v predchádzajúcej časti spĺňajú toto kritérium. Všeobecná schéma na vytvorenie digitálneho podpisu správy má podľa definície tieto časti

- Algoritmus na generovanie páru privátnych a verejných kľúčov
- Podpisový algoritmus závislý od privátneho kľúča, ktorý vytvorí podpis pre danú správu
- Overovací algoritmus závislý od verejného kľúča, ktorý pre správu prijme alebo zamietne zodpovedajúci podpis

Niektoré kryptosystémy túto schému implementujú tak, že ako podpisovú funkciu použijú dešifrovací algoritmus s privátnym kľúčom a ako overovaciu zvolia šifrovací algoritmus s verejným kľúčom. Podpis a overenie v tejto implementácii môže vyzeráť takto

Algoritmus 5 Schéma digitálneho podpisu

Máme správu m , ktorú chceme podpísať a odtlačkovú funkciu H

Vypočítame odtlačok $h = H(m)$

Vypočítame podpis tak, že dešifrujeme odtlačok h pomocou privátneho kľúča

Podpísanú správu tvorí dvojica (m, s)

Možnosť tejto implementácie je silne závislá od toho, ako sa prekrývajú množiny šifrovaných textov a odtlačkov v konkrétnom kryptosystéme. Ako si ukážeme v ďalších častiach práce, nie všetky odtlačky musia byť dešifrovateľné správy.

3.2 Prehľad code-based podpisových schém

V ďalších častiach práce sa už budeme zaoberať iba code-based podpisovými schémami, teda schémami, ktoré využívajú code-based kryptosystémy. Veľkou prekážkou týchto kryptosystémov je v súčasnosti veľkosť kľúča, ktorá je v porovnaní s dnešnými kryptosystémami rádovo tisícnásobne väčšia. Pri implementácii a následne v praxi je dôležité nájsť vhodný kompromis medzi požadovanou bezpečnosťou a výpočtovou a dátovou náročnosťou, ktorá závisí od voľby veľkosti kľúča.

Existuje niekoľko potenciálnych návrhov code-based kryptosystémov, z ktorých si bližšie prejdeme CFS (Courtois-Finiasz-Sendrier) a LDGM (Low-density generator matrix).

3.3 CFS schéma

Jedným z nádejných návrhov code-based podpisových schém je CFS schéma, ktorá používa na podpisovanie Niederreiterov kryptosystém. Základný problém, ktorý treba vyriešiť pri podpisovaní založenom na kódovaní, je ako získať taký odtlačok správy, ktorý je dekódovateľné slovo. Ak máme lineárny kód $C(n, k, 2t + 1)$, syndróm slova je vektor dĺžky $n - k$. Počet všetkých syndrémov je 2^{n-k} a počet dekódovateľných syndrémov je $\sum_{i=0}^t \binom{n}{i}$. To znamená, že $\frac{\sum_{i=0}^t \binom{n}{i}}{2^{n-k}}$ všetkých syndrémov je dekódovateľných. Pre Goppove kódy je to približne $\frac{1}{t!}$. Pravdepodobnosť, že odtlačok správy bude zároveň dekódovateľný, je teda približne $p = \frac{1}{t!}$. Nato, aby sme vedeli podpísať každú správu, budeme musieť ku správe pridať bity navyše, a pokúsiť sa podpísať túto upravenú správu. Priemerný počet pokusov na podpísanie jednej správy je približne $t!$.

Algoritmus 6 Algoritmus podpisovania v CFS

Vstup: Správa m , odtlačková funkcia H ktorá vracia odtlačky dĺžky $n - k$

Výstup: Podpis správy m , ozn. sig

```

 $i \leftarrow 0$ 
repeat
     $h \leftarrow H(m||i)$ 
    if  $h$  nie je dekódovateľné slovo then
         $i \leftarrow i + 1$ 
    end if
until  $h$  je dekódovateľné slovo
 $e \leftarrow Decode(h)$ 
 $sig \leftarrow (e, i)$ 
return  $sig$ 

```

Implementácia uvedeného algoritmu môže byť vylepšená po viacerých stránkach. Prvé vylepšenie sa dá realizovať pri hľadaní dekódovateľného syndrému. Na začiatku podpisovania si vypočítame hash samotnej správy $h' = H(m)$ a v ďalších krokoch počítame $h = H(h'||i)$. Ďalší priestor na vylepšenie, tentokrát dĺžka výsledného podpisu, sa ponúka v spôsobe uloženia časti e z podpisu. Autori tejto podpisovej schémy navrhli ukladať e

ako index I z množiny všetkých n bitových vektorov s váhou t . To predstavuje číslo z rozsahu $< 1, \binom{n}{t} >$

Algoritmus 7 Algoritmus overovania v CFS

Vstup: Podpis (e, i) , správa m , verejný kľúč H' , odtlačková funkcia H ktorá vracia odtlačky dĺžky $n - k$

Výstup: True/False - podpis prijímame/zamietame

$s_1 \leftarrow H'e^T$

$s_2 \leftarrow H(m||i)$

return $s_1 = s_2$

Kedže podpis správy tvorí chybové slovo s pevnou váhou, pri parametroch $n = 2^{16}, t = 9$, ktoré navrhli autori, môže byť podpis výrazne kompromovaný, čo je značná výhoda oproti schéme, ktorá využíva klasický McElieceov kryptosystém.

3.4 LDGM podpis

Ďalší z možných návrhov pre code-based kryptografiu sa pokúša zmenšiť potrebnú veľkosť kľúča pomocou vhodne zvoleného kódu, respektíve pomocou vhodne zvolenej generačnej matice. LDGM (Low-density generation matrix) kódy, čiže kódy s riedkou generovaciou maticou sa v niektorých prípadoch dajú zapísať kompaktne pomocou cirkulantných matíc. Generovacia matica G kódu dĺžky n s dimenziou k sa skladá z $k_0 n_0$ blokov s rozmermi $p \times p$, kde $n_0 = n/p$ a $k_0 = k/p$.

$$G = \begin{bmatrix} C_{0,0} & C_{0,1} & \cdots & C_{0,n_0-1} \\ C_{1,0} & C_{1,1} & \cdots & C_{1,n_0-1} \\ \vdots & \vdots & \ddots & \vdots \\ C_{k_0-1,0} & C_{k_0-1,1} & \cdots & C_{k_0-1,n_0-1} \end{bmatrix}$$

Každé $C_{i,j}$ je $p \times p$ cirkulatná matica. Vďaka tomu nám stačí uložiť z každého bloku iba jeden riadok. Tým zmenšíme veľkosť kľúča p -násobne. Matica v takomto tvare sa nazýva kvázicyklická (QC). K matici G vypočítame kontrolnú maticu H v systematickom tvare, t.j. $H = [X|I]$. Kontrolná matica H je súčasťou súkromného kľúča. Ďalšiu časť kľúča tvoria matice Q a S . Postup ako určiť maticu Q je zhrnutý v nasledovnom algoritme. Maticu S určíme ako náhodnú maticu poskladanú z $n_0 \times n_0$ cirkulantných blokov veľkosti $p \times p$ tak, aby váha každého riadku aj stĺpca bola w_s a aby mala plnú hodnotu.

Verejný kľúč tvorí upravená kontrolná matica $H' = Q^{-1}HS^{-1}$. Spôsob, ktorým počítame matice Q a S zachováva QC vlastnosti pôvodnej matice H , čo nám umožňuje zmenšenie veľkosti verejného kľúča.

Algoritmus 8 Výpočet matice Q

Určíme náhodne matice a, b s rozmermi $z \times r_0$, $z \leq r_0$

Vypočítame maticu $R \leftarrow a^T b \otimes 1_{p,p}$

Určíme maticu T poskladanú z $r_0 \times r_0$ cirkulantných matic tak, aby váha každého riadku aj stĺpca bola w_t a aby $\text{rank}(R + T) = r$

$Q \leftarrow R + T$

Pozn. $r_0 = n_0 - k_0$, $1_{p,p}$ - matica $p \times p$ samé jednotky, \otimes - Kroneckerov súčin

Pri generovaní musíme riešiť podobný problém ako pri CFS schéme. Potrebujeme výstup z hashovacej funkcie transformovať na vektor, ktorý spĺňa podmienky určené štruktúrou kódu. Prepokladajme, že máme funkciu ϕ , ktorá jednoznačne priradí vektoru dĺžky l vektor dĺžky r s váhou w a funkciu ψ , ktorá pre každú správu m vyberie kódové slovo s nízkou váhou w_c z kódu generovaného maticou G . Generovanie podpisu je opísane v nasledovnom algoritme.

Overenie podpisu je rýchle a prebieha takto.

3.4.1 Ukážkový príklad

Pre lepšiu predstavu ako LDGM podpisová schéma funguje si predvedieme generovanie, podpisovanie a overenie na príklade. Hodnoty parametrov použité v príklade sú odlišné od hodnôt vhodných pre praktické využitie.

Generovanie kľúčov Majme kód $C(15, 9)$ s kontrolnou maticou H v systematickom tvare

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Rozmer cirkulantných matíc je $p \times p$, $p = 3$, $n_0 = 5$, $r_0 = 2$.

Ďalej vypočítame maticu Q . Matica Q je tvorená ako súčet matíc R a T . Na výpočet matice R zvolíme matice a, b s rozmermi $z \times r_0$, kde $z \leq r_0$.

$$a = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

$$b = \begin{bmatrix} 0 & 1 \end{bmatrix}$$

Algoritmus 9 Podpis v LDGM

Vstup: Správa m

Výstup: Podpis správy m , ozn. sig

Vypočítame odtlačok správy m

$h \leftarrow H(m)$

$i \leftarrow 0$, $|h| + |i| = l$

repeat

$s \leftarrow \phi(h||i)$

$b_1 \leftarrow b \otimes 1_{1,p}$

if $b_1 s \neq 0$ **then**

$i \leftarrow i + 1$

end if

until $b_1 s = 0$

$s' \leftarrow Qs$

$e \leftarrow [0_{1,k} || s'^T]$

$c \leftarrow \psi(m)$

$e' \leftarrow (e + c)S^T$

$sig \leftarrow (e', i)$

return sig

Pozn. $0_{1,k}$ - nulový vektor dĺžky k

Matica $R = (a' \times b) \otimes 1_{p,p}$

$$R = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Maticu T vyberáme tak, aby sa skladala z $r_0 \times r_0$ cirkulatných blokov, váha každého stĺpca a riadku bola w_t a hodnosť $R + T$ bola $r = r_0 p$. Zvolme $w_t = 1$.

Algoritmus 10 Overenie v LDGM

Vstup: Správa m , podpis (e', i)

Výstup: True/False - podpis prijímame/zamietame

if $wt(e') > (w_t w + w_c) w_s$ **then**

return False

else

$s \leftarrow \phi(H(m)||i)$

if $wt(s) \neq w$ **then**

return False

else

$s_1 \leftarrow H' e'^T$

return $s = s_1$

end if

end if

$$T = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

$$Q = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Matica S je regulárna matica tvorená z $n_0 \times n_0$ cirkulatných blokov, každý stĺpec a riadok má hodnotu w_s . Nech $w_s = 1$, potom matica S môže vyzeráť nasledovne

$$S = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Keď máme matice Q a S , môžeme vypočítať maticu $H' = Q^{-1}HS^{-1}$, ktorá tvorí verejný kľúč

$$H' = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Vytvorenie podpisu Predpokladajme, že máme správu m , pre ktorú je výstup z funkcie ϕ vektor s váhou w

$$s = [0 \ 0 \ 0 \ 1 \ 0 \ 1]^T \quad w = 2$$

Ďalej vypočítame $s' = Q \cdot s$

$$s' = [0 \ 0 \ 0 \ 1 \ 1 \ 0]^T$$

Vektoru s' zodpovedá chybové slovo v tvare $e = [0_{1 \times k}, s'^T]$

$$e = [0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 1 \ 0]$$

Predstavme si, že funkcia $\psi(m)$ vyberie kódové slovo c s váhou $w_c = 4$ pre správu m

$$c = [1 \ 1 \ 0 \ 1 \ 0 \ 0 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0]$$

Posledný krok podpisu je výpočet $e' = (e + c) \cdot S^T$

$$e' = [0 \ 1 \ 0 \ 1 \ 0 \ 1 \ 0 \ 0 \ 0 \ 1 \ 1 \ 0 \ 1 \ 0 \ 0]$$

Podpis tvorí dvojica (e', i) . V tomto príklade však i neuvádzame.

Overenie podpisu Ako prvé overíme, či $wt(e') \leq (w_t \cdot w + w_c) \cdot w_s$. V našom príklade $w_t = 1$, $w = 2$, $w_c = 4$ a $w_s = 1$. Potom $6 \leq (1 \cdot 2 + 4) \cdot 1$, $6 \leq 6$ Prvá podmienka je splnená. Ďalej pomocou verejného kľúča zrekonštruujeme odtlačok správy $s_0 = H' \cdot e'^T$

$$s_0 = [0 \ 0 \ 0 \ 1 \ 0 \ 1]^T$$

Vidíme, že $s = s_0$ a podpis teda akceptujeme.

3.5 Porovnanie

Pozrime sa na porovnanie CFS a LDGM schémy.

Tabuľka 2: Porovnanie CFS a LDGM schémy

	CFS	LDGM
Velkosť VK	$n - k \times n$	$(n - k \times n)/p$
Velkosť PK	$n - k \times n - k, n - k \times n, n \times n$	$(n - k \times n, n - k \times n - k, n \times n)/p$
Velkosť podpisu	n	n

Vidíme, že pri rovnakých parametroch kódu LDGM schéma ponúka p-násobnú úsporu pamäte. Treba však podotknúť, že kódy použité v každej schéme sa líšia a z praktického hľadiska sa volia iné hodnoty parametrov.

4 Návrh a implementácia LDGM schémy

V tejto časti uvidíme návrh a implementáciu LDGM podpisovej schémy spolu s konkrétnymi algoritmami na generovanie kľúčov, podpisov a overovacie funkcie.

4.1 Parametre

Nasledujúca tabuľka obsahuje všetky voliteľné parametre a funkcie pre celú podpisovú schému LDGM spolu s odporúčanými hodnotami na dosiahnutie 80 bitovej bezpečnosti

Tabuľka 3: Parametre a funkcie LDGM

Parameter	Popis	80 bit SL
n	dĺžka kódu (n_0p)	9800
k	dimenzia kódu (k_0p)	4900
p	veľkosť cirkulantných matíc	50
z	počet riadkov matice a, b pozn. ($z \leq (n - k)$)	2
w_g	váha riadku generujúcej matice G	20
w_t	váha riadku/stĺpca matice T	1
w_s	váha riadku/stĺpca matice S	9
w_c	váha slova c , ktoré určuje funkcia ψ	160
H	hashovacia funkcia	-
ψ	funkcia, ktorá jednoznačne priradí správe m kódové slovo c	-
ϕ	funkcia, ktorá zobrazuje slová dĺžky x na vektory dĺžky y s váhou z	-

4.2 Funkcia ϕ

Dôležitou súčasťou pre celú implementáciu, je zvoliť funkciu ϕ , ktorá jednoznačne mapuje vektor dĺžky n na vektor dĺžky l s váhou t . Pre tento účel zvolíme algoritmus, ktorý navrhol Sendrier v rámci CFS schémy v [xx]. Tento algoritmus má lineárnu zložitosť a oproti algoritmu, ktorý použili autori LDGM schémy ponúka výrazné zrýchlenie celého výpočtu.

Implementácia ako ju navrhol Sendrier je uvedená v algoritme 11.

4.3 Funkcia ψ

Na zamaskovanie chybového slova, ktoré tvorí podstatu podpisu a má špecifický tvar $[0k|s]$ (viď algoritmus 9), k nemu pripočítame kódové slovo, ktoré sa odvodí v závislosti od správy. Toto slovo musí mať váhu nižšiu alebo rovnú ako predpísaný parameter schémy w_c . Existuje veľa spôsobov ako takéto slovo určiť a v algoritme 12 je uvedený postup,

Algoritmus 11 Funkcia ϕ

Vstup: Dĺžka slova - n , váha slova - t , prúd bitov - B

Výstup: Vektor dĺžky n s váhou t

$t_{tuple} \leftarrow \text{BTOCW}(n, t, 0, B)$

return $\text{convertTupleToVector}(t_{tuple})$

function $\text{BTOCW}(n, t, \delta, B)$

if $t = 0$ **then**

return

else if $n \leq t$ **then**

return $\delta, \text{BTOCW}(n - 1, t - 1, 0, B)$

else

$d \leftarrow (n - \frac{t-1}{2})(1 - \frac{1}{2^{1/t}})$

if $\text{read}(B, 1) = 1$ **then**

return $\text{BTOCW}(n - d, t, \delta + d, B)$

else

$i \leftarrow \text{DECODEFD}(d, B)$

return $\delta + i, \text{BTOCW}(n - i - 1, t - 1, 0, B)$

end if

end if

end function

function $\text{DECODEFD}(d, B)$

$u \leftarrow \lceil \log_2(d) \rceil$

$\delta \leftarrow \text{read}(B, u - 1)$

if $\delta \geq 2^u - d$ **then**

$\delta \leftarrow 2\delta + \text{read}(B, 1) - 2^u + d$

end if

end function

Pozn. $\text{read}(n, B)$ je funkcia, ktorá prečíta n bitov z prúdu v desiatkovej reprezentácii a posunie prúd o n bitov.

ktorý používame v našej implementácii. Generujúca matica G je špeciálne vytvorená aby každý riadok mal vopred určenú váhu w_g a keďže je to náhodná matica s nízkou váhou, je malá pravdepodobnosť, že jednotky v riadkoch sú na rovnakých pozíciách. Preto na

určenie slova s váhou menšou alebo rovnou w_c stačí spočítať w_c/w_g riadkov z matice G .

Algoritmus 12 Funkcia ψ

Vstup: Správa m , váha výsledného slova w_c

Výstup: Kódové slovo s váhou $\leq w_c$

```

 $b \leftarrow \log_2(k)$ 
 $count \leftarrow w_c/w_g$ 
 $h \leftarrow H(m)$ 
 $c \leftarrow 0_n$ 
for  $i \leftarrow 0$  to  $count$  do
     $index_{row} \leftarrow binToDec(h[(i * b) : (i + 1) * b])$ 
     $c \leftarrow c \oplus G[index_{row}]$ 
end for
return  $c$ 

```

Pozn. G - generujúca matica, k - počet riadkov G , n - dĺžka G , H - hash. funkcia

4.4 Generovanie kľúčov

Generovanie páru kľúčov sa skladá z viacerých krokov

- Vygenerovať generujúcu maticu G
- Vygenerovať maticu Q
- Vygenerovať maticu S
- Vypočítať maticu H_{pub}

Pseudokód generovania kľúčov je zhrnutý v algoritme 13

Každý krok generovania rozoberieme v osobitnej časti. Podľa návrhu schémy potrebujeme inverzné matice Q^{-1} a S^{-1} , takže matice Q a S musia byť regulárne. Pseudokód v algoritme 13 ukazuje, že sme na generovanie matíc Q a S zvolili stratégiu generuj a testuj a teda ak sa nám nepodarí vygenerovať maticu invertovať, generovanie opakujeme.

Invertovanie kvazicyklických matíc zohráva pri generovaní kľúčov podstatnú úlohu. Preto predtým ako si priblížime algoritmy na generovanie matíc Q a S , sa budeme venovať invertovaniu kvazicyklických matíc.

Algoritmus 13 Generovanie klúčov

Vstup: Parametre kódu a schémy - $params$

Výstup: Pár klúčov

```
 $G \leftarrow generateGenMatrix(params)$ 
repeat
   $Q \leftarrow generateMatrixQ(params)$ 
   $Q_{inv} \leftarrow tryToInvert(Q)$ 
until  $exists(Q_{inv})$ 
repeat
   $S \leftarrow generateMatrixS(params)$ 
   $S_{inv} \leftarrow tryToInvert(S)$ 
until  $exists(S_{inv})$ 

 $H_{pub} \leftarrow buildMatrixH(G, Q_{inv}, S_{inv})$ 
 $PublicKey \leftarrow H_{pub}$ 
 $PrivateKey \leftarrow G, Q_{inv}, S_{inv}$ 
return  $PrivateKey, PublicKey$ 
```

4.5 Invertovanie QC matice

Binárne cirkulantné matice veľkosti n môžu byť reprezentované polynómom $p(x) \in GF(2)[x]/(x^n - 1)$. $p(x)$ je stupňa maximálne $n - 1$ a reprezentuje prvý riadok cirkulantnej matice, i -ty riadok matice je reprezentovaný polynómom $x^{i-1}p(x)$ pre $i \in 1, \dots, n$.

Kvázicyklická matica je matica, ktorej všetky prvky sú polynómy z $GF(2)[x]/(x^n - 1)$. Každý polynóm predstavuje jeden cirkulantný blok. Pre kvázicyklickú maticu môžeme riadkovo ekvivalentné operácie rozšíriť o operácie, ktoré pracujú nad blokmi. To znamená, že na počítanie inverznej matice môžeme použiť Gaussovu eliminačnú metódu nad okruhom polynómov. Kvázicyklickú maticu M veľkosti n rozšírime zprava o jednotkovú maticu (tiež v QC tvare). Rozšírenú maticu upravíme do redukovaného stupňovitého tvaru. Maticu M^{-1} potom tvoria stĺpce $[n + 1, 2n]$.

$$\left[\begin{array}{ccc|cccc} p_{1,1}(x) & \cdots & p_{1,n}(x) & 1 & 0 & \cdots & 0 \\ p_{2,1}(x) & \cdots & p_{2,n}(x) & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ p_{n,1}(x) & \cdots & p_{n,n}(x) & 0 & 0 & \cdots & 1 \end{array} \right] \longrightarrow \left[\begin{array}{cccc|cccc} 1 & 0 & \cdots & 0 & p'_{1,1}(x) & \cdots & p'_{1,n}(x) \\ 0 & 0 & \cdots & 0 & p'_{2,1}(x) & \cdots & p'_{2,n}(x) \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & p'_{n,1}(x) & \cdots & p'_{n,n}(x) \end{array} \right]$$

Na to aby sme pomocou jedného riadku mohli eliminovať ostatné, potrebujeme najst

Algoritmus 14 Invertovanie QC matice

Vstup: QC matica M

Výstup: QC matica M^{-1} ak existuje, inak 0

```
 $M_{eye} \leftarrow [M|I]$ 
for  $i \leftarrow 1$  to  $n$  do
  for  $j \leftarrow i$  to  $n$  do
    if existsPolynomialInverse( $M_{eye}[j][i]$ ) then
       $inverse \leftarrow polynomialInverse(M_{eye}[j][i])$ 
       $swap(M_{eye}[i], M_{eye}[j])$ 
       $M_{eye}[i] \leftarrow M_{eye}[i] * inverse$ 
      break
    end if
  end for
  if exists( $inverse$ ) then
    for  $j \leftarrow 1$  to  $n$  do
      if  $j = i$  then
        continue
      end if
       $M_{eye}[j] \leftarrow M_{eye}[j] + M_{eye}[i] * M_{eye}[j][i]$ 
    end for
  else
    return 0
  end if
end for
return  $M_{eye}[1 : n][n + 1 : 2n]$ 
```

vedúci prvok (pivot). V polynomickej reprezentácii to znamená nájsť v stĺpci prvok, ktorý sa dá invertovať modulo $x^n - 1$. Inverziu hľadáme pomocou Euklidovho rozšíreného algoritmu. Ak nájdeme pivot, pripočítame k ostatným riadkom taký násobok riadku, ktorý obsahuje pivot, aby sme v stĺpci nad aj pod ním dostali nuly. Ak v stĺpci pivot nenájdeme, buď matica nie je regulárna a teda inverzia neexistuje, alebo je štruktúra matice taká, že jednoduchou Gaussovou elimináciou nie sme schopní inverziu nájsť. Je dôležité zdôrazniť, že nie sme schopní z výsledku algoritmu určiť či inverzia existuje alebo nie. Ak inverzia neexistuje, algoritmus je neúspešný, ale môže nastať prípad keď inverzia existuje ale napriek tomu algoritmus inverziu nevypočíta. Úspešnosť algoritmu

do veľkej miery súvisí od veľkosti cirkulatných blokov. Čím viac faktorov má polynóm $x^n - 1$, tým je väčšia šanca, že náhodný polynóm s ním bude súdeliteľný a teda k nemu neexistuje inverzný prvok. Ak je malá šanca nájsť inverzný prvok, znižuje to celkovú šancu na úspech Gaussovej eliminácie, pretože nevieme nájsť pivot. Predchádzajúce tvrdenie sa potvrdzuje aj vo výsledkoch meraní, ktoré sú uvedené v samostatnej kapitole. Napríklad pri veľkosti bloku 21 počítame s polynómom $x^{21} - 1$. Tento polynóm má rozklad nad $GF(2)$ $(x + 1)(x^2 + x + 1)(x^3 + x + 1)(x^3 + x^2 + 1)(x^6 + x^4 + x^2 + x + 1)(x^6 + x^5 + x^4 + x^2 + 1)$. Polynóm $x^{19} - 1$ má rozklad $(x + 1)(x^{18} + x^{17} + x^{16} + x^{15} + x^{14} + x^{13} + x^{12} + x^{11} + x^{10} + x^9 + x^8 + x^7 + x^6 + x^5 + x^4 + x^3 + x^2 + x + 1)$. Invertovanie by teda malo byť výrazne úspešnejšie pri veľkosti bloku 19 ako pri veľkosti 21, čo aj potvrdzujú merania.

4.6 Generovanie matice G

Generovanie náhodnej matice G je pomerne priamočiare. G je kvázicyklcká matica tvaru $[I|X]$ kde matica X je náhodná matica, v ktorej každý riadok má rovnakú váhu. Parametre ktoré vstupujú do generovania sú počet blokov v riadku n , počet blokov v stĺpci k , veľkosť bloku p a váha riadku w_g . Pseudokód generovania je uvedený v algoritme 15

Algoritmus 15 Generovanie matice G

Vstup: Počet blokov v riadku - n , počet blokov v stĺpci - k , veľkosť bloku - p , váha riadku

w_g

Výstup: Generujúca matica G

$X \leftarrow 0_{k,n-k}$

for $i \leftarrow 1$ **to** k **do**

$w \leftarrow w_g - 1$

while $w > 0$ **do**

$p(x) \leftarrow X[i][rand(n)]$

if $wt(p(x)) < p$ **then**

$addRandBit(p(x))$

$w \leftarrow w - 1$

end if

end while

end for

$G \leftarrow [I|X]$

return G

Pozn. $rand(n)$ vráti celé číslo z rozsahu $[1, n]$, $addRandBit(p(x))$ pridá jednotku na náhodnú pozíciu v polynóme $p(x)$

4.7 Generovanie matice Q

Matica Q sa skladá z matíc R a T , ktoré su blokovy cirkulantné. Matica R vznikne ako Kroneckerov súčin $a^T b \otimes 1_{p,p}$. Matice a, b náhodne vygenerujeme v závislosti od parametra z . Matica T sa skladá z cirkulantných matíc s rozmerom $p \times p$. A váha každého riadku a stĺpca je w_t , nepárne. Generovanie kvázicyklických matíc s pevnou váhou stĺpcov a riadkov je popísane v algoritme 16.

Algoritmus 16 Generovanie QC matice s pevnou váhou

Vstup: Počet cirkulantných blokov - n , veľkosť bloku - p , predpísaná váha w_t

Výstup: Kvázicyklická matica s predpísanou váhou riadkov a stĺpcov

```
blocks  $\leftarrow$   $0_n$ 
 $i \leftarrow w_t$ 
if  $\text{odd}(w_t)$  then
    blocks[1]  $\leftarrow$  1
     $i \leftarrow i - 1$ 
end if
while  $i > 0$  do
     $r \leftarrow \text{rand}(n)$ 
    if blocks[r]  $\leq n - 2$  then
        blocks[r]  $\leftarrow$  blocks[r] + 2
         $i \leftarrow i - 2$ 
    end if
end while
 $T \leftarrow 0_{n,n}$ 
for  $i \leftarrow 1$  to  $n$  do
    for  $j \leftarrow 1$  to  $n$  do
         $T[i][j] \leftarrow \text{randPolyOfWeight}(\text{blocks}[(i + j) \bmod n])$ 
    end for
end for
 $X \leftarrow \text{permuteRowBlocks}(X)$ 
 $X \leftarrow \text{permuteColumnBlocks}(X)$ 
return  $X$ 
```

4.8 Generovanie matice S

S je náhodná kvázicyklická matica s pevnou váhou riadkov a stĺpcov. Na jej generovanie môžeme použiť algoritmus 16, ktorý sme použili pri generovaní matice Q

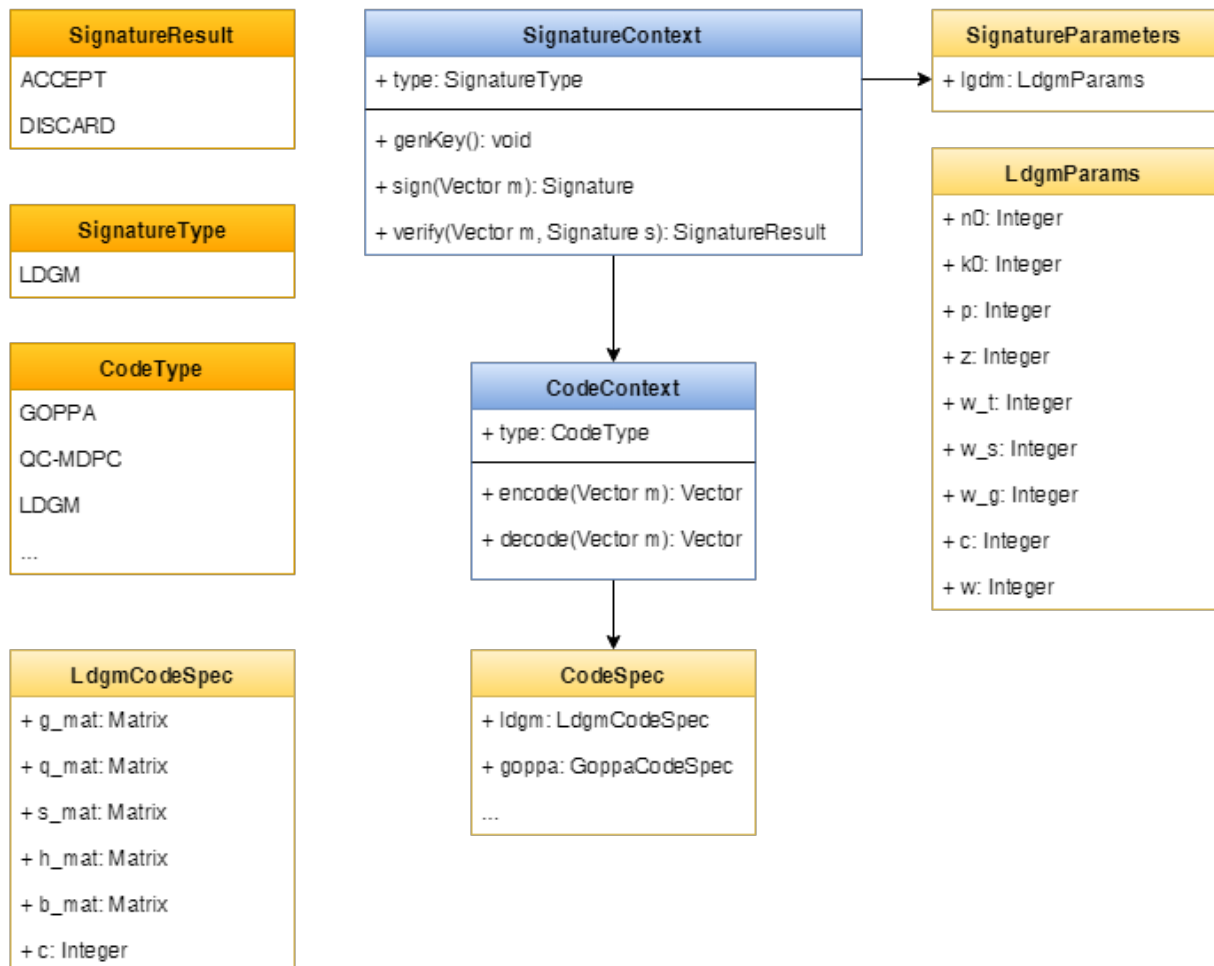
4.9 Výpočet matice H_{pub}

Predtým, ako sa pustíme do výpočtu matice H_{pub} , potrebujeme inverzné matice ku Q a S . Ak nevieme k niektorej z matíc nájsť inverziu, opakujeme generovanie a skúsime nájsť inverziu znova. Ak sa nám úspešne podarí nájsť inverziu k oboom maticiam, maticu H_{pub} vypočítame ako súčin matíc $Q^{-1}HS^{-1}$.

4.10 Implementácia v BitPunch

BitPunch je opensource kryptografická knižnica napísaná v jazyku C, ktorú postupne vyvíjajú študenti FEI STU. Primárne zameranie knižnice je na postkvantovú kryptografiu založenej na dekodovacom probléme. Táto práca rozširuje funkčnosť o digitálne podpisy, konkrétne o LDGM schému. BitPunch obsahoval jednoduchú implementáciu kvázicyklických matic, ktoré sa používali pri QC-MDPC implementácii McEliecovho kryptosystému. Táto implementácia však bola prispôbená špeciálne pre potreby QC-MDPC schémy a musela byť rozšírená pre všeobecné použitie. Do knižnice bolo pridaných veľa procedúr, ktoré pracujú s kvázicyklickými maticami, vektormi a polynómami nad $GF(2)$. Spôsob akým sú digitálne podpisy integrované do BitPunch knižnice je načrtnutý na diagrame.

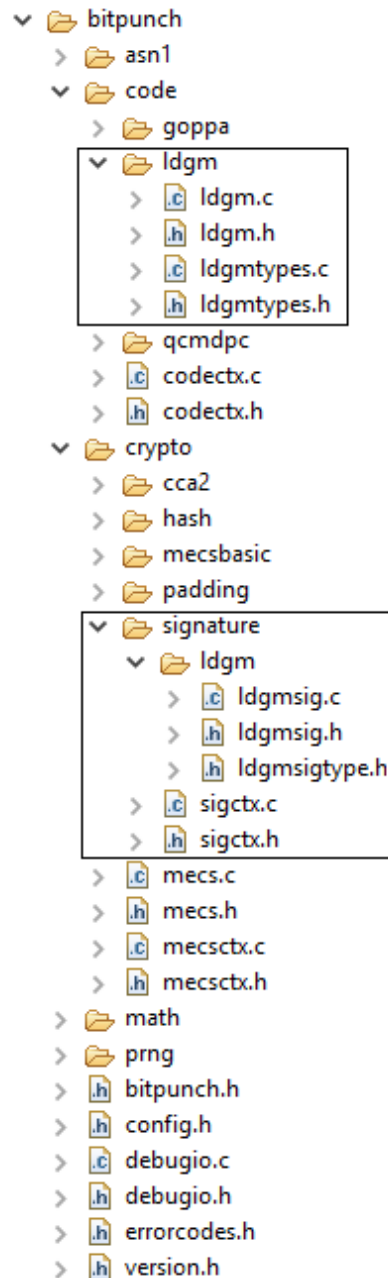
Obrázok 1: Diagram štruktúr v BitPunch



SignatureContext slúži ako premenná prostredia, ktorá obsahuje špecifikáciu kódu a parametre pre podpisovú schému. Štruktúra *CodeSpec* je implementovaná ako union. Tento spôsob používania union dátového typu má simulovať návrhový vzor *Strategy*,

ktorý však nie je možné kvalitne implementovať v jazyku C, pretože C nie je objektovo orientovaný jazyk.

Obrázok 2: Organizačná štruktúra v BitPunch



Na obrázku s organizačnou štruktúrou projektu BitPunch sú vyznačené časti, ktoré pribudli ako súčasť tejto práce.

Kompletné zdrojové kódy k implementácii sa nachádzajú v prílohe.

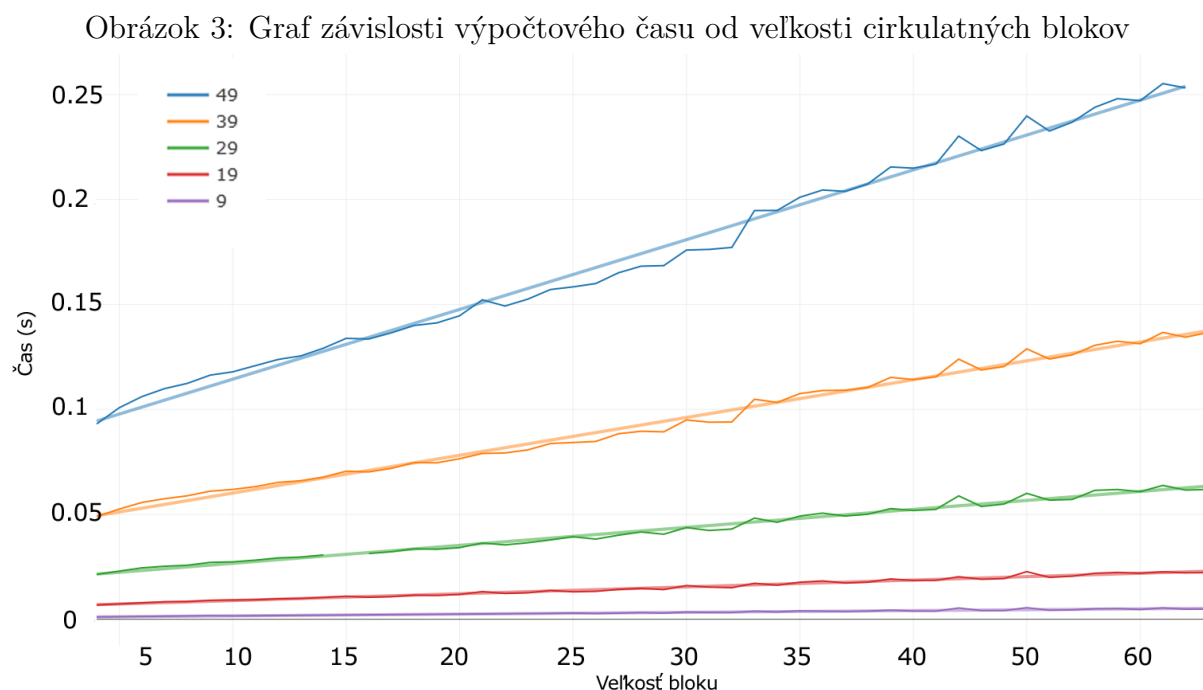
5 Výsledky meraní

BitPunch s LDGM podpismi sme testovali na zostave Intel Core2 Quad Q8200 @ 2.33GHz, 4GB RAM s operačným systémom Windows 10. Knižnica bola pri testovaní kompilovaná s GCC kompilátorom s O3 optimalizáciou. V rámci testovania sme testovali úspešnosť a časovú zložitosť invertovania kvázicyklických matíc, výkonnosť životného cyklu LDGM podpisovej schémy, čo znamená generovanie kľúčov, podpisovanie náhodnej správy a overenie podpisu. Výsledky sme porovnali s výkonnosťou digitálneho podpisu implementovaného v OpenSSL.

5.1 Invertovanie QC matíc

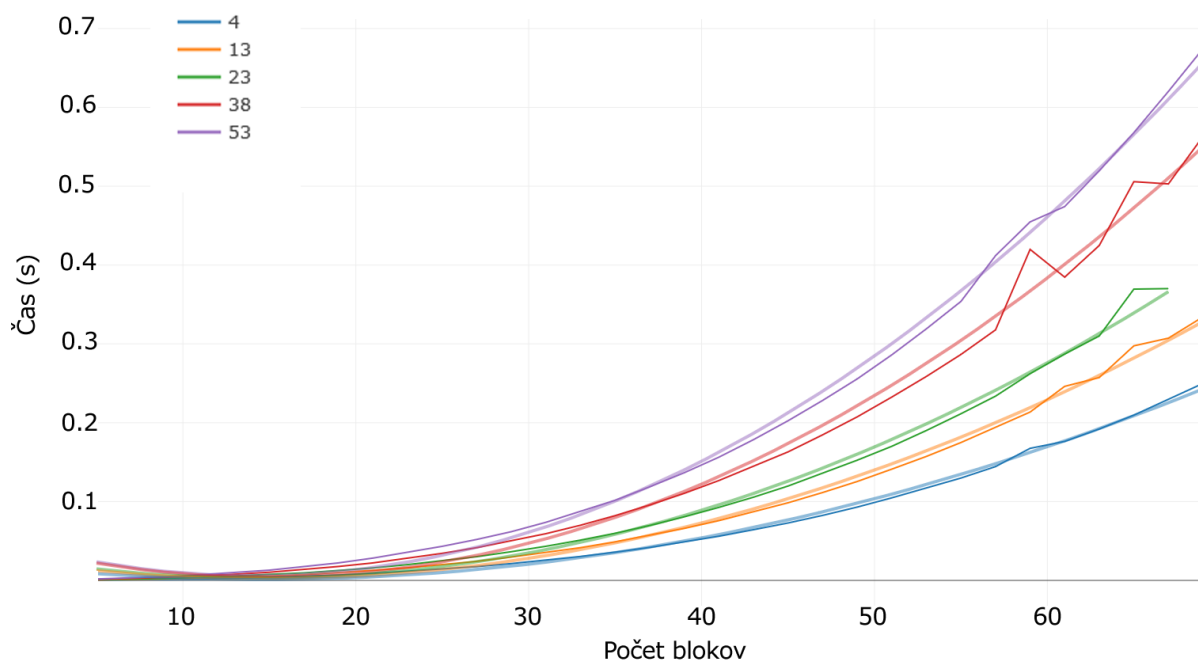
Vstupné parametre testovania invertovania QC matíc boli počet cirkulatných blokov a veľkosť bloku. Výstupom testu bola úspešnosť invertovania zo 100 pokusov a priemerný čas výpočtu.

Parameter počtu blokov sme testovali s hodnotami 5, 7, 9, ..., 71 a veľkosť bloku od 4 po 53. Čas výpočtu inverznej matice je kvadraticky závislý od počtu blokov a lineárne od veľkosti bloku. Toto tvrdenie vyplýva z návrhu algoritmu a potvrdzujú ho aj výsledky testovania.



Na grafe sú zobrazené časové závislosti od veľkosti blokov pre rôzne veľké matice. Pod krivkami sú naznačené regresné priamky.

Obrázok 4: Graf závislosti výpočtového času od počtu blokov v matici



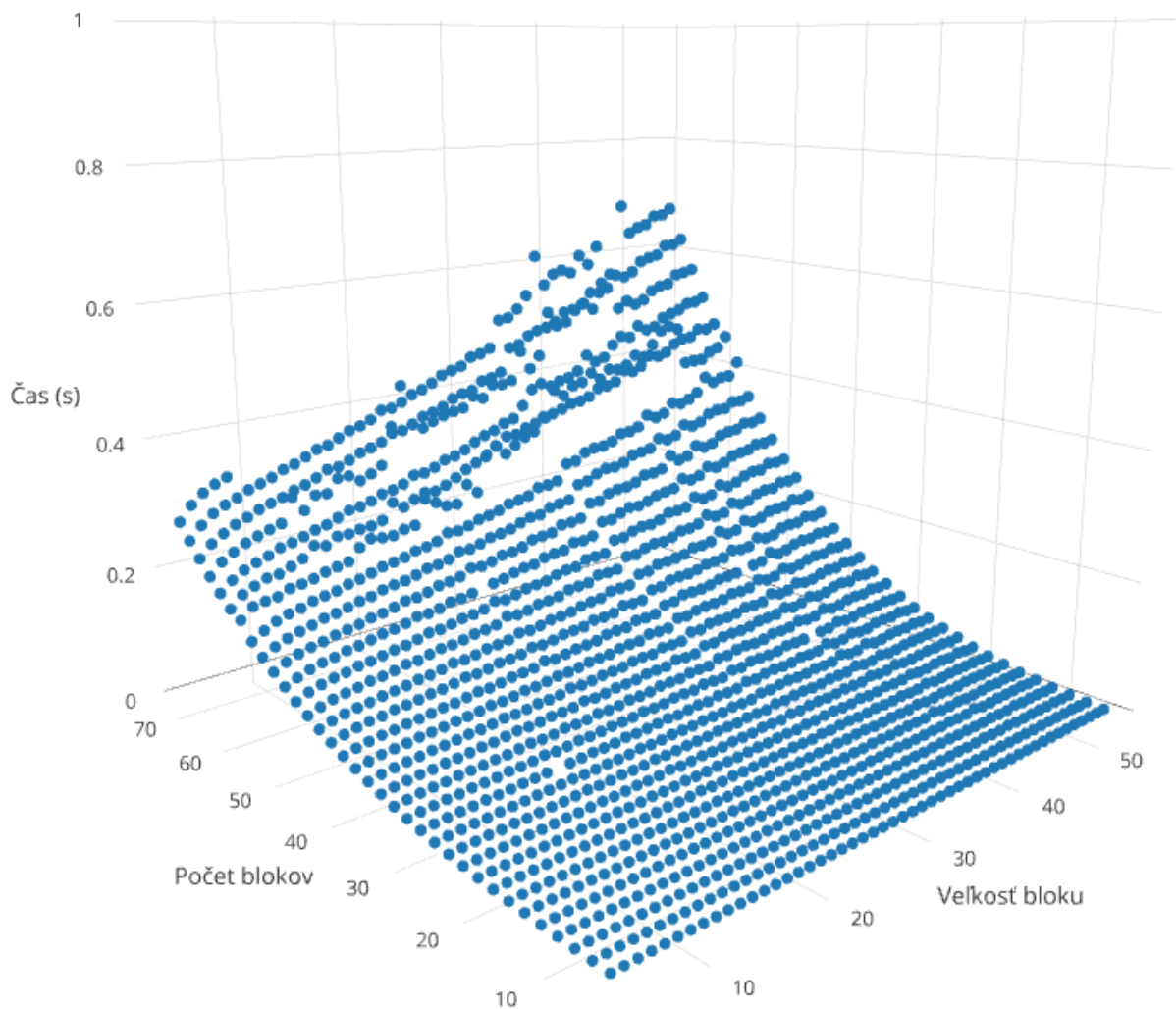
Predchádzajúci graf zobrazuje časové závislosti od počtu blokov pre rôzne veľkosti cirkulantných blokov. Pod krivkami sú naznačené kvadratické regresné krivky.

V tabuľke 4 sú uvedené hodnoty pre vybrané parametre testovania, kompletne výsledky sú uvedené v prílohe.

Tabuľka 4: Výsledky časovej zložitosti invertovania QC matíc

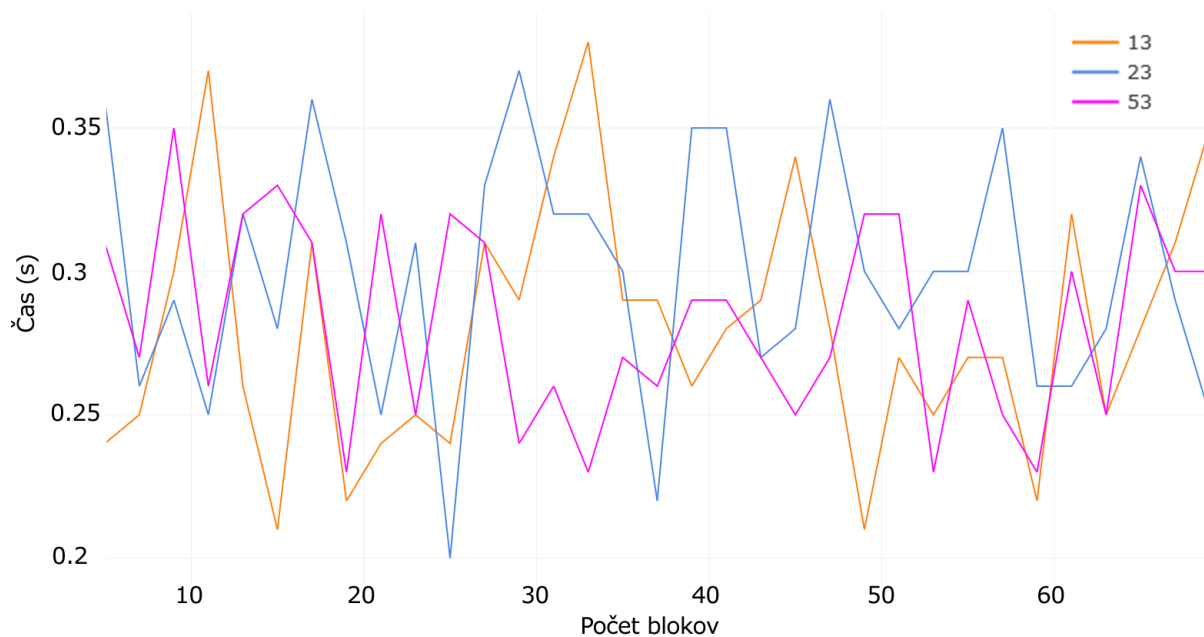
Počet blokov	Veľkosť bloku	Priemerný čas (s)
5	4	0.0003
9	19	0.0022
13	31	0.0065
21	41	0.0234
31	49	0.0721
41	52	0.1543
51	53	0.2860
63	53	0.5195

Obrázok 5: Graf závislosti výpočtového času od počtu blokov a veľkosti blokov



Z výsledkov testovania nevyplýva, že úspešnosť invertovania závisí od počtu blokov v matici.

Obrázok 6: Graf závislosti úspešnosti od počtu blokov



Ako bolo uvedené v časti, ktorá sa venovala implementácii, úspešnosť nezávisí priamo od veľkosti a počtu blokov ale od počtu faktorov polynómu $x^n - 1$.

Tabuľka 5: Výsledky pre najúspešnejšie veľkosti blokov

Veľkosť bloku	Úspešnosť (s)
16	0.337
32	0.336
43	0.336
37	0.335
52	0.334
8	0.334
46	0.33
34	0.33
38	0.33
41	0.33
23	0.327
47	0.326
4	0.325
17	0.324

V tabuľke 5 sú zobrazené veľkosti blokov, pre ktoré je úspešnosť najvyššia. Je zaujímavé pozorovať, že sa v nej nachádzajú všetky mocniny dvojky, ktoré boli súčasťou testovania. Pre veľkosti blokov, ktoré sú v tvare 2^n , má polynóm $x^{2^n} - 1$ nad $GF(2)$ rozklad $(x - 1)^{2^n}$, teda malý počet rôznych faktorov a preto vyššia šanca na úspech.

Tabuľka 6: Výsledky pre najneúspešnejšie veľkosti blokov

Veľkosť bloku	Úspešnosť (s)
21	0.028
42	0.03
45	0.04
15	0.046
30	0.047
36	0.06
27	0.071
35	0.072
7	0.0733
28	0.07375
18	0.076
24	0.077
49	0.077
9	0.077

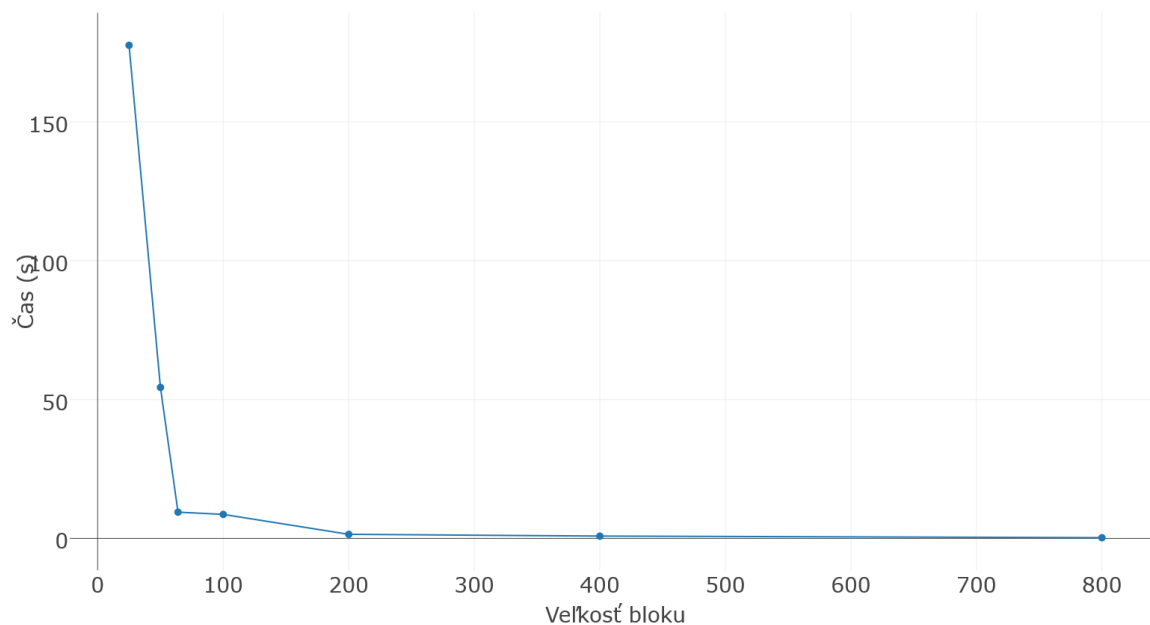
V tabuľke 6 sú zobrazené výsledky pre bloky, ktoré mali najnižšiu úspešnosť. Vidieť, že vo výsledkoch sú hodnoty, pre ktoré majú polynómy veľa faktorov a ich spoločné násobky.

5.2 Výkonnosť LDGM

Pri testovaní LDGM implementácie sme sa zamerali hlavne na testovanie s odporúčanými parametrami, ale testovali sme aj to, ako sa prejaví zmeny v parametroch na časovej zložitosti. Merali sme časovú zložitnosť generovania kľúčov, podpisovania a overovania. Podľa odporúčaných parametrov je veľkosť generujúcej matice 9800×4900 a veľkosť blokov je 50. Generujúcu maticu teda tvorí 196×98 blokov. Testovali sme však aj iné násobky počtu blokov s veľkosťou blokov. Keďže pri generovaní kľúčov sa počítajú inverzné matice a časová zložitnosť počítania inverzie závisí kvadraticky od počtu blokov, čím menej blokov tým rýchlejšie je generovanie. Toto je vidieť aj na výsledkoch testov. Ďalšie zrýchlenie generovania vieme dosiahnuť ak zvolíme veľkosť blokov takú, pre ktorú má polynóm $x^n - 1$ málo faktorov. Na druhú stranu, keďže chceme maximálne využiť zmenšenie veľkosti

klúča, je vhodné zvoliť veľkosť bloku v okolí \sqrt{k} , kde k je veľkosť generujúcej matice.

Obrázok 7: Graf závislosti času generovania od počtu veľkosti blokov



Tabuľka 7: Výsledky merní pre LDGM implementáciu

Počet blokov	Veľkosť blokov	Generovanie (s)	Podpis (s)	Overovanie(s)
12	800	0.253	0.0218	0.291
25	400	0.827	0.01	0.275
49	200	1.461	0.005	0.224
98	100	8.641	0.004	0.239
153	64	9.449	0.009	0.225
196	50	54.393	0.004	0.221
392	25	177.634	0.007	0.157

Z výsledkov, ktoré sú v tabuľke 7 vychádza, že najlepšie výsledky a najefektívnejšie zmenšenie veľkosti kľúča je pre veľkosť bloku 64.

Pre porovnanie výkonnosti s RSA2048 uvádzame tabuľku s výsledkom merania OpenSSL.

	Podpis (s)	Overovanie (s)
RSA	0.012	0.0003
LDGM	0.004	0.239

Záver

Záver SK

Resumé

Resume SK

Prílohy