

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

Evidenčné číslo: FEI-5384-5930

**IMPLEMENTÁCIA QC-MDPC MCELIECOVHO**  
**KRYPTOSYSTÉMU**  
**DIPLOMOVÁ PRÁCA**

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

Evidenčné číslo: FEI-5384-5930

**IMPLEMENTÁCIA QC-MDPC MCELIECOVHO**  
**KRYPTOSYSTÉMU**  
**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.



## ZADANIE DIPLOMOVEJ PRÁCE

Študent: **Bc. Andrej Gulyás**  
ID študenta: 5930  
Študijný program: Aplikovaná informatika  
Študijný odbor: 9.2.9. aplikovaná informatika  
Vedúci práce: doc. Ing. Pavol Zajac, PhD.  
Miesto vypracovania: Ústav informatiky a matematiky

Názov práce: **Implementácia QC-MDPC McElieceovho kryptosystému**

Špecifikácia zadania:

McElieceov kryptosystém je jedným zo základných predstaviteľov postkvantovej kryptografie.

Úlohy:


1. Naštudujte problematiku implementácie McElieceovho kryptosystému vo verzii s QC MDPC kódmi.
2. Navrhňte aplikáciu.
3. Implementujte a otestujte riešenie.

Zoznam odbornej literatúry:


1. Heyse, Stefan, Ingo Von Maurich, and Tim G. "Smaller keys for code-based cryptography: QC-MDPC McEliece implementations on embedded devices." Cryptographic Hardware and Embedded Systems-CHES 2013. Springer Berlin Heidelberg, 2013. 273-292.
2. McEliece, Robert J. "A public-key cryptosystem based on algebraic coding theory." DSN progress report 42.44 (1978): 114-116.
3. Misoczki, Rafael, et al. "MDPC-McEliece: New McEliece variants from moderate density parity-check codes." Information Theory Proceedings (ISIT), 2013 IEEE International Symposium on. IEEE, 2013.


Riešenie zadania práce od: 22. 09. 2014

Dátum odovzdania práce: 22. 05. 2015

  
**Bc. Andrej Gulyás**  
študent



  
**prof. RNDr. Otokar Grošek, PhD.**  
vedúci pracoviska

  
**prof. RNDr. Otokar Grošek, PhD.**  
garant študijného programu

# SÚHRN

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE  
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Študijný program: Aplikovaná informatika  
Diplomová práca: Implementácia QC-MDPC McEliecovho kryptosystému  
Autor: Bc. Andrej Gulyás  
Vedúci záverečnej práce: doc. Ing. Pavol Zajac, PhD.  
Miesto a rok predloženia práce: Bratislava 2015

Práca sa venuje preskúmaniu možnosti využitia post-quantovej kryptografie založenej na teórii kódovania. Zaoberá sa analýzou McEliecovho kryptosystému s Goppa kódmi. Práca poukazuje na nedostatky tohto kryptosystému spočívajúce vo veľkosti kľúčov nevhodných pre zariadenia s obmedzenými zdrojmi. V práci sa analyzuje vývoj McEliecovho kryptosystému s použitím riedkych LDPC kódov miesto Goppa kódov. Vývoj vedie až k nedávno navrhnutému využitiu kvázi-cyklických hustejších MDPC (QC-MDPC) kódov v tomto kryptosystéme. Tento návrh je vybraný pre bližšie skúmanie z dôvodu dostatočnej kryptografickej bezpečnosti pri výraznom zredukovaní veľkosti kľúčov. Ďalšia časť tejto práce sa venuje návrhu McEliecovho kryptosystému s QC-MDPC kódmi. Bližšie rozoberá navrhnuté algoritmy a ich implementáciu. V poslednej časti sú prezentované testy a výsledky dosiahnuté na implementovanom kryptosystéme. Kryptosystém je porovnaný s McEliecovým kryptosystémom s Goppa kódmi a so štandardným kryptosystémom RSA. Testy preukázali možnosť použitia tohto kryptosystému v praxi pri zachovaní kryptografickej bezpečnosti a výraznej redukcii veľkosti kľúčov. Kryptosystém je úspešne implementovaný v jazyku C ako rozšírenie kryptografickej knižnice BitPunch.

Kľúčové slová: post-quantová kryptografia, McEliecov kryptosystém, QC-MDPC kódy, kryptografia založená na teórii kódovania, implementácia kryptosystému

# ABSTRACT

SLOVAK UNIVERSITY OF TECHNOLOGY IN BRATISLAVA  
FACULTY OF ELECTRICAL ENGINEERING AND INFORMATION  
TECHNOLOGY

Study Programme:	Applied Informatics
Diploma Thesis:	Implementation of QC-MDPC McEliece cryptosystem
Author:	Bc. Andrej Gulyás
Supervisor:	doc. Ing. Pavol Zajac, PhD.
Place and year of submission:	Bratislava 2015

The main topic of this thesis is post-quantum code-based cryptography. The thesis starts with an analysis of the McEliece cryptosystem with the Goppa codes. This cryptosystem has several deficiencies, such as large keys for usage in devices with limited resources. The thesis continues with the summary of the McEliece cryptosystem with sparse LDPC codes instead of the Goppa codes. Recently, McEliece cryptosystem with the quasi-cyclic denser MDPC (QC-MDPC) codes was proposed. It has sufficient cryptographical security in combination with significant reduction of key size. The next part of the thesis consists of the design and implementation of algorithms for McEliece cryptosystem with QC-MDPC codes. The tests and achieved results with own implementation of cryptosystem are presented in the last part of the thesis. The results are compared with McEliece cryptosystem with Goppa codes and standardized RSA cryptosystem. The tests show that it is possible to use McEliece cryptosystem with QC-MDPC codes efficiently in practice while preserving the same security level provided by other cryptosystems and small keys. The cryptosystem is successfully implemented in C language as an extension for the cryptographic library BitPunch.

Key words: post-quantum cryptography, McEliece cryptosystem, QC-MDPC codes, code-based cryptography, implementation of cryptosystem

## **Vyhlásenie autora**

Podpísaný Andrej Gulyás čestne vyhlasujem, že som diplomovú prácu Implementácia QC-MDPC McEliecevho kryptosystému vypracoval na základe poznatkov získaných počas štúdia a informácie z dostupnej literatúry uvedenej v práci. Uvedenú prácu som vypracoval pod vedením doc. Ing. Pavol Zajac, PhD.

V Bratislave dňa 15.5.2015

.....

podpis autora

## **Pod'akovanie**

Touto cestou by som sa chcel pod'akovať vedúcemu diplomovej práce doc. Ing. Pavlovi Zajacovi, PhD., za odborné vedenie, neoceniteľné rady a pripomienky, ktoré mi pomohli pri vypracovaní tejto diplomovej práce.

# Obsah

<b>Úvod</b>	<b>1</b>
<b>1 Analýza</b>	<b>3</b>
1.1 Značenie . . . . .	4
1.2 Teória kódovania . . . . .	4
1.3 McEliecov kryptosystém s Goppa kódmi . . . . .	6
1.3.1 Popis McEliecovho kryptosystému . . . . .	6
1.4 Zovšeobecnený McEliecov kryptosystém . . . . .	8
1.4.1 LDPC kódy . . . . .	8
1.4.2 QC-LDPC kódy . . . . .	9
1.5 McEliecov kryptosystém založený na QC-MDPC kódov . . . . .	12
1.5.1 QC-MDPC kódy . . . . .	12
1.5.2 Schéma QC-MDPC McEliecovho kryptosystému . . . . .	13
1.5.3 Dekódovanie QC-MDPC kódov . . . . .	14
1.6 Bit-flipping algoritmus . . . . .	15
1.6.1 Popis algoritmu . . . . .	15
1.6.2 Modifikácie bit-flipping algoritmu . . . . .	18
1.6.3 Parametre kryptosystému . . . . .	21
<b>2 Návrh riešenia</b>	<b>22</b>
2.1 Knižnica BitPunch . . . . .	22
2.2 Návrh QC-MDPC McEliecovho kryptosystému . . . . .	22
2.2.1 Použité štruktúry . . . . .	22
2.2.2 Generovanie kľúčov . . . . .	24
2.2.3 Šifrovanie . . . . .	26
2.2.4 Dešifrovanie . . . . .	26
<b>3 Výsledky testov</b>	<b>30</b>
3.1 Testovacie prostredie . . . . .	30
3.2 Optimalizácia . . . . .	31
3.3 Porovnanie bit-flipping algoritmov . . . . .	33
3.3.1 Algoritmus A3 . . . . .	33
3.3.2 Algoritmus B2 . . . . .	35
3.3.3 Návrh dešifrovania . . . . .	36



3.4	Porovnanie kryptosystémov . . . . .	37
3.5	Porovnanie našej implementácie s inou . . . . .	44
<b>Záver</b>		<b>45</b>
<b>Zoznam použitej literatúry</b>		<b>46</b>
<b>Prílohy</b>		<b>48</b>
	Príloha A - elektronické médium . . . . .	48

## Zoznam grafov

1	Porovnanie voľby parametra $\delta$ pre algoritmus A3 . . . . .	34
2	Veľkosť kľúčov pre rôzne kryptosystémy . . . . .	39
3	Spotreba pamäte pre rôzne kryptosystémy . . . . .	40
4	Dĺžka trvania generovania kľúčov pre rôzne kryptosystémy . . . . .	41
5	Dĺžka trvania šifrovania a dešifrovania pre rôzne kryptosystémy . . . . .	42
6	Priepustnosť šifrovania pre rôzne kryptosystémy . . . . .	42
7	Priepustnosť dešifrovania pre rôzne kryptosystémy . . . . .	43

## Zoznam tabuliek

1	Veľkosť verejného kľúča pre McEliecov kryptosystém s použitím Goppa kódov	8
2	[2] Porovnanie navrhnutých modifikácií bit-flipping algoritmu . . . . .	20
3	[1] Odporúčané parametre pre QC-MDPC McEliecov kryptosystém . . . . .	21
4	[1] Porovnanie veľkostí kľúčov pre McEliecov kryptosystém s použitím rôznych kódov . . . . .	21
5	Časy trvania operácií pre pôvodný návrh . . . . .	31
6	Časy trvania operácií pre zmenu implementácie kontrolnej matice $H$ . . . . .	31
7	Časy trvania operácií pre optimalizáciu funkcií . . . . .	32
8	Časy trvania operácií pre optimalizáciu kontroly $G \times H^T = 0$ . . . . .	32
9	Časy trvania operácií po zmene algoritmu generovanie náhodných polynómov.	33
10	Porovnanie voľby parametra $\delta$ pre algoritmus A3 . . . . .	34
11	Hodnoty DFR v závislosti od parametra $\delta$ . . . . .	35
12	Predpočítané hodnoty $B[i]$ pre algoritmus B2 . . . . .	35
13	Porovnanie algoritmu A3 a algoritmu B2 . . . . .	36
14	Porovnanie algoritmu B2 a nami navrhnutej kombinácie algoritmov B2 a A3	36
15	[18] Parametre McEliecovho kryptosystému s Goppa kódmi . . . . .	38
16	Porovnania kryptosystémov . . . . .	39
17	Porovnanie našej implementácie s implementáciou z Ruhr-Universität Bochum . . . . .	44

# Úvod

Téma post-quantovej kryptografie začína byť v posledných rokoch veľmi aktuálna. Viaceré technologické spoločnosti vynakladajú finančné prostriedky a snahu na vývoj kvantového počítača. Bolo uverejnených niekoľko článkov ohľadne reálneho pokroku v tejto oblasti. Avšak zatiaľ sa predpokladá, že univerzálny kvantový počítač nebol skonštruovaný.

V roku 1994 Shor [10] prezentoval algoritmus pre kvantové počítače, ktorý je schopný v reálnom čase faktorizovať súčin prvočísel a počítať diskretný logaritmus na eliptických krivkách. To v praxi znamená, že kryptosystémy založené na týchto ťažkých matematických problémoch by boli na kvantových počítačoch zlomené. Ide o známe a používané kryptosystémy ako RSA, DSA, ECDSA a Diffie-Hellmanova schéma na výmenu kľúčov.

Toto bolo dôvodom, prečo začal kryptografický svet hľadať možné alternatívy pre nahradenie týchto kryptosystémov v prípade vzniku kvantových počítačov. Jedným z možných smerov sú aj kryptosystémy založené na teórii kódovania. Najznámejšími predstaviteľmi tohto smeru sú McElievov kryptosystém a Niederreiterov kryptosystém. Oblasť kryptosystémov založených na teórii kódovania je dobre preskúmaná v priebehu viac ako 35 rokov.

Táto práca sa zaoberá "lightweight" verziou McElievovho kryptosystému, ktorá využíva QC-MDPC kódy. Tieto kódy boli odporúčané v článku [1] ako vhodná alternatíva k bežne používaným Goppa kódom, ktorých použitie je obmedzené kvôli príliš veľkým kľúčom. Pomocou QC-MDPC kódov je možné zredukovať veľkosť kľúčov pri zachovaní rovnakej bezpečnosti.

V kapitole *Analýza* sú zadefinované základné pojmy z oblasti teórie kódovania. Ďalej sa venuje McElievomu kryptosystému, jeho nevýhodám a skúma navrhnuté použitie kvázi-cyklických kódov. Kapitola bližšie ukazuje vývoj v oblasti použitia kvázi-cyklických kódov pre kryptografiu. V závere kapitoly je uvedený návrh reálneho použitia týchto kódov v McElievom kryptosystéme s odporúčanými parametrami vzhľadom na úroveň bezpečnosti.

Kapitola *Návrh riešenia* sa venuje samotnému návrhu McElievovho kryptosystému s QC-MDPC kódmi. Je v nej prezentovaný návrh reprezentácie potrebných matematických primitív, navrhnuté algoritmy pre generovanie kľúčov, šifrovanie a dešifrovanie.

V kapitole *Výsledky testov* je prezentovaná naša implementácia McElievovho kryptosystému, je tam ukázaný optimalizačný proces algoritmov. Na základe testov je vyhodnotený optimálny prístup k dekódovaniu QC-MDPC kódov pomocou porovnania viacerých bit-flipping algoritmov. Kombináciou viacerých prístupov je navrhnutý dekódovací algo-

ritmus, ktorý dosahuje najlepšie hodnoty. Kapitola sa taktiež orientuje na porovnanie implementácie McEliecovho kryptosystému s Goppa kódmi, nami implementovanej verzie s QC-MDPC kódmi a RSA. Naša implementácia je tiež porovnaná s inou implementáciou QC-MDPC kódov.

V závere sú zhrnuté ciele tejto práce a zhodnotenie dosiahnutých cieľov. Taktiež sú tu spomenuté odporúčania pre ďalšie pokračovanie.

Táto práca bola realizovaná ako súčasť projektu "Secure implementation of post-quantum cryptography", NATO Science for Peace and Security Programme Project Number: 984520.

# 1 Analýza

V úvode tejto kapitoly sa oboznámime so základnými pojmami z oblasti lineárnych kódov. Ďalej sa budeme zaoberať McEliecovým kryptosystémom, ako reprezentantom post-quantovej kryptografie.

Existuje viacero kryptosystémov, ktoré sú schopné odolať doposiaľ známym kvantovým algoritmom. Najznámejšie sú:

- **Kryptografia založená na hashovacích funkciách.** Klasickým príkladom je Merkleho hashovací strom, čo predstavuje podpisovú schému s verejným kľúčom. Táto podpisová schéma vychádza z Lamportovej a Diffieho jednorázovej podpisovej schémy.
- **Kryptografia založená na teórii kódovania.** Ide o kryptosystémy ako sú McEliecov kryptosystém a Niederreiterov kryptosystém.
- **Mriežkové kryptosystémy.** Najznámejším predstaviteľom tejto triedy je Hoffsteinov Pipherov Silvermanov kryptosystém s verejným kľúčom NTRU.
- **Kryptosystémy založené na systéme kvadratických rovníc.** Predstaviteľom je Patarinova podpisová schéma s verejným kľúčom HFE.
- **Symetrická kryptografia.** Štandardizovaný symetrický kryptosystém Rijndael známy ako AES (Advanced Encryption Standard) navrhnutý Daemenom a Rijmenom.

My si bližšie rozoberieme McEliecov kryptosystém, zdefinujeme si schému šifrovania a dešifrovania. Načrtujeme základné princípy bezpečnosti tohto kryptosystému a použité Goppa kódy. Uvedieme hlavné nedostatky tohto kryptosystému spojené s Goppa kódmi. Predovšetkým hovoríme o kľúčoch, ktoré sú kvôli svojej veľkosti nepoužiteľné na zariadeniach s obmedzenými zdrojmi.

Ďalej sa budeme venovať možným variantom McEliecovho kryptosystému, ktoré sú založené na použití iných kódov. Varianty sa týmto snažia odstrániť nedostatky veľkých kľúčov. Jednou z ciest sa ukazuje použitie riedkych kvázi-cyklických LDPC kódov. Využívajú cyklické matice, ktoré nám redukujú potrebný priestor pre uloženie a spracovanie kľúčov v pamäti. Odprezentujeme kryptoanalýzy, ktoré poukazujú na bezpečnostné nedostatky navrhnutých variantov. Prejdeme si vývojom týchto kódov vzhľadom na použiteľnosť v kryptosystémoch.

Na konci kapitoly sa budeme venovať McElievommu kryptosystému s použitím hustejších MDPC kódov s kvázi-cyklickou štruktúrou matíc. Odstraňujú známe bezpečnostné nedostatky LDPC kódov. Zároveň prinášajú pre McEliev kryptosystém reálnu použiteľnosť na obmedzených zariadeniach, keďže značne redukujú veľkosti kľúčov.

## 1.1 Značenie

V práci budeme ďalej používať nasledovné značenie:

**m** - otvorený text

**c** - zašifrovaný text

**e** - chybový vektor

**C** - lineárny kód

**G** - generujúca matica

**H** - kontrolná matica

$C^\perp$  - duálny kód

**S** - nesingulárna "scrambling" matica

**P** - permutačná matica

**#** - počet

**upc** - nevyhovujúce rovnice z kontrolnej matice (unsatisfied parity-check equations)

**DFR** - miera chyby dekódovania (decoding failure rate)

**gcd** - najväčší spoločný deliteľ (greatest common divisor)

## 1.2 Teória kódovania

Uvedieme si základné definície týkajúce sa teórie kódovania, na ktoré budeme nadväzovať v celej práci. Budeme sa venovať lineárnym kódom, generujúcej a kontrolnej matici lineárneho kódu.

### Lineárny kód

Vo všeobecnosti je možné definovať lineárne kódy nad ľubovoľným poľom  $F_q$ . My sa však obmedzíme pre jednoduchosť na binárny prípad, t.j.  $q = 2$ . Máme dané konečné pole  $F_2$ . Množina  $F_2^n$  binárnych vektorov dĺžky  $n$  nad poľom  $F_2$  s aditívnou operáciou "+" (sčítanie po zložkách) a multiplikatívnou operáciou "." (násobenie zložiek vektora prvkom poľa  $F_2$ ) tvorí vektorový podpriestor. Binárnym lineárnym kódom nad poľom  $F_2$  je ľubovoľný vektorový podpriestor vektorového priestoru  $F_2^n$ . Ak je dimenzia vektorového podpriestoru  $C$  rovná  $k$ , lineárny kód  $C$  sa nazýva lineárny  $(n, k)$ -kód. [11]

**Definícia 1.** [1] Binárny  $(n, k)$ -lineárny kód  $C$  dĺžky  $n$ , s dimenziou  $k$ , je  $k$ -dimenzionálny vektorový podpriestor  $F_2^n$ .

Lineárny kód je taká ľubovoľná neprázdna množina vektorov  $C$ , že pre ľubovoľné slovo  $u \in C$  a prvok  $a \in F_2$  patrí do kódu  $C$  aj slovo  $au$ . Pre dve ľubovoľné kódové slová  $u, v \in C$  je aj  $u + v$  a  $u - v$  kódové slovo kódu  $C$ . Teda aj slovo  $u - u = 0$  patrí do každého kódu a označuje sa nulové slovo.[11]

**Definícia 2.** [1] (Hammingová váha): Hammingová váha vektora  $u \in F_2^n$  sa označuje  $\mathbf{wt}(u)$  a vyjadruje počet nenulových prvkov vektora  $u$ .  $\mathbf{wt}(u) = \sum_{i=0}^{n-1} u_i$ .

Ďalej vychádzame z definícií [12]. Hammingová vzdialenosť  $\mathbf{d}(u, v)$  dvoch kódových slov  $u, v$  je definovaná ako počet miest na ktorých sa tieto dve slová líšia

$$\mathbf{d}(u, v) = \mathbf{wt}(u - v).$$

Najmenšia Hammingová vzdialenosť  $d_{min}$  medzi všetkými kódovými slovami sa nazýva minimálna vzdialenosť a je vyjadrená ako

$$d_{min} = \min_{\forall u, v; u \neq v} \mathbf{d}(u, v).$$

Nakoľko lineárny kód vždy obsahuje aj nulové kódové slovo, minimálna vzdialenosť kódu je

$$d_{min} = \min_{\forall u; u \neq 0} \mathbf{wt}(u).$$

Počet chýb, ktoré dokáže kód opraviť, je daný vzťahom

$$t = \lfloor \frac{d_{min} - 1}{2} \rfloor.$$

## Generujúca a kontrolná matica

Generujúca matica  $G$ ,  $(n, k)$ -lineárneho kódu  $C$ , je matica  $k \times n$  lineárne nezávislých kódových slov. Všetky kódové slová kódu  $C$  môžu byť vytvorené ako lineárne kombinácie riadkov generujúcej matice. Existuje teda  $2^k$  kódových slov pre binárny lineárny kód.

Kontrolná matica kódu  $C$  je  $(n - k) \times n$  matica  $H$ , pre ktorú platí

$$GH^T = 0,$$

kde  $0$  je  $k \times (n - k)$  matica núl.

Kód  $C^\perp$  je podpriestor nazývaný ortogonálny doplnok priestoru  $C$ .  $C^\perp$  je taktiež podpriestor vektorového priestoru  $F_2^n$ , teda je tiež lineárny kód, ktorý budeme nazývať duálny kód ku kódu  $C$ . Teda generujúca matica kódu  $C^\perp$  je kontrolnou maticou  $H$  kódu  $C$ .

**Definícia 3.** [1] Matica  $G \in F_2^{k \times n}$  sa nazýva generujúca matica kódu  $C$ . Matica  $G$  je jadrom matice  $H \in F_2^{(n-k) \times n}$ , nazývanej kontrolná matica kódu  $C$ . Majme kódové slovo  $c \in C$  a vektor  $m \in F_2^k$ , pričom platí  $c = mG$ . Pre každé kódové slovo  $c \in C$  platí,  $Hc^T = 0$ . Syndróm  $s \in F_2^{n-k}$  vektora  $e \in F_2^n$  dostaneme ako  $He^T = s$ . Duálny kód  $C^\perp$  ku kódu  $C$  je lineárny kód popísaný ľubovoľnou kontrolnou maticou kódu  $C$ .

Dva lineárne kódy  $C_1$  a  $C_2$  sú ekvivalentné, ak majú rovnaký priestor kódových slov. Generujúcu maticu  $G$  kódu  $C_1$  môžeme upraviť do systematického redukovaného tvaru

$$G' = [I_k P],$$

kde  $G'$  generuje ekvivalentný kód  $C_2$ . Ak pre lineárne kódy  $C_1$  a  $C_2$  platí, že existuje permutácia  $\sigma$  taká, že  $(u_1, \dots, u_n) \in C_1 \Leftrightarrow (u_{\sigma(1)}, \dots, u_{\sigma(n)}) \in C_2$ , potom sú tieto dva kódy permutačne ekvivalentné.

### 1.3 McEliecov kryptosystém s Goppa kódmi

McEliecov kryptosystém bol navrhnutý v roku 1978 R. McEliecom [7] ako jeden z prvých kryptosystémov s verejným kľúčom. Tento kryptosystém bol navrhnutý s využitím Goppa kódov, ktoré sú schopné opraviť  $t$  chýb. Pri šifrovaní vložíme do zašifrovanej správy presne  $t$  chýb. Iba legitímny príjemca správy pozná štruktúru použitého Goppa kódu a je schopný efektívnym algoritmom tieto chyby opraviť a následne správu dešifrovať. Na tomto predpoklade je založená bezpečnosť šifrovania. Nakoľko je známy rýchly algoritmus na dekódovanie Goppa kódov a šifrovanie je principiálne veľmi rýchle, je tento kryptosystém zaujímavý. Bezpečnosť kryptosystému je založená na dvoch predpokladoch:

- Nerozlíšiteľnosť rodiny použitých kódov v kryptosystéme. Nakoľko tento predpoklad je veľmi silný a ukázal sa byť ťažko dosiahnuteľný, vyžaduje sa slabšie kritérium: nemožnosť nájdania permutačne ekvivalentného kódu, s ktorým sme schopní dešifrovať.
- Problém dekódovania (decoding problem). Problém dekódovania je známym a dobre preskúmaným NP-úplným problémom.

#### 1.3.1 Popis McEliecovho kryptosystému

Predpokladá sa, že je známy kód, ktorý dokáže opraviť  $t$  chýb. Pre McEliecov kryptosystém bolo odporúčané použitie samoopravných Goppa kódov. Poznáme rýchly Pat-



tersonov algoritmus pre dekódovanie Goppa kódov. Potom schéma kryptosystému vyzerá nasledovne.

### Generovanie kľúčov

Používateľ McEliecovho kryptosystému si náhodne zvolí polynóm Goppa kódu, ktorý musí byť ireducibilný a musí mať stupeň  $t$ . Pomocou polynómu vieme zostrojiť kód, ktorý opravuje najviac  $t$  chýb. Polynóm je nad  $F_2^m$ , pričom  $m$  je parameter kryptosystému, ktorý ovplyvňuje úroveň bezpečnosti. Na základe zvoleného polynómu skonštruuje generujúcu maticu  $G$  pre daný Goppa kód. Nakoľko sa generujúca matica upravuje do systematického redukovaného tvaru, je potrebné ju zamaskovať. Používateľ náhodne vygeneruje nesingulárnu maticu  $S$  a permutačnú maticu  $P$ . Vypočíta

$$G' = SGP,$$

čím sa matica "premieša a zamaskuje". Týmto vznikne permutačný ekvivalentný lineárny kód, ktorý má rovnaké vlastnosti ako má kód s generujúcou maticou  $G$ . Verejným kľúčom je matica  $G'$ . Matice  $S, G, P$  ostanú utajené ako súkromný kľúč.

### Šifrovanie

Zoberme otvorený text  $m$  vhodnej dĺžky a šifrujeme ako

$$c = mG' + e = mSGP + e,$$

kde  $e$  je chybový vektor dĺžky  $n$  s váhou  $t$ .

### Dešifrovanie

Legitímny príjemca pozná matice  $S, G, P$ . Vypočíta inverznú maticu  $P^{-1}$  k permutačnej matici  $P$ . Potom vypočíta  $m'$  ako

$$m' = cP^{-1} = mSGPP^{-1} + eP^{-1} = mSG + eP^{-1}.$$

Použitím Pattersonovho algoritmu dekóduje kódové slovo  $m'$  a dostane

$$m'' = mS,$$

kde otvorený text  $m$  je možné dostať ako

$$m = m''S^{-1} = mSS^{-1}.$$

## Parametre

V kryptoanalýze McEliecovho kryptosystému [8] boli odporúčené parametre pre 80-bitovú bezpečnosť s prihliadnutím na všetky doteraz známe najsilnejšie útoky. Ide o voľbu parametrov  $m = 11$  a  $t = 27$ . V článku venovanému reálnej implementácii [6] uvádzajú použité parametre  $m = 11$  a  $t = 50$ . Veľkosť verejného kľúča pre dané úrovne bezpečnosti podľa [1] sú uvedené v tabuľke.

Bitová bezpečnosť	Veľkosť kľúča
80	460 647
128	1 537 536
256	7 667 855

Tabuľka 1 Veľkosť verejného kľúča pre McEliecov kryptosystém s použitím Goppa kódov

McEliecov kryptosystém s použitím Goppa kódov je dostatočne bezpečný pre správnu voľbu parametrov. Taktiež sú známe algoritmy, s ktorých využitím vieme dosiahnuť dostatočujúcu rýchlosť generovania kľúčov, šifrovania a dešifrovania. Nevýhodou pre reálne použitie v praxi predstavujú veľkosti kľúčov. Ide predovšetkým o použitie kryptosystému na obmedzených zariadeniach s limitovanými zdrojmi. Taktiež v prípade McEliecovho kryptosystému s použitím Goppa kódov bol nedávno prezentovaný rozlišovač [13], ktorý dokáže rozlíšiť maticu Goppa kódov od náhodnej matice.

## 1.4 Zovšeobecnený McEliecov kryptosystém

Použitie Goppa kódov nie je úplne ideálne pre McEliecov kryptosystém. My si predstavíme verzie McEliecovho kryptosystému používajúce iné kódy. Hlavným cieľom je zabezpečiť redukciu veľkosti kľúča pri zachovaní rovnakej miery bezpečnosti.

### 1.4.1 LDPC kódy

Low-Density Parity Check (LDPC) kódy [4] sa ukázali byť dobrým kandidátom na použitie v McEliecovom kryptosystéme. Boli viackrát odporúčané pre použitie v kryptosystéme v článkoch [14], [15] a [16].

Predstavme si LDPC kódy.

**Definícia 4.** [1]  $(n, k, w)$ -LDPC (Low Density Parity-Check) kód je lineárny kód dĺžky  $n$ , s dimenziou  $k$  a konštantnou váhou riadkov kontrolnej matice  $w$ .

LDPC kódy majú váhu riadkov kontrolnej matice  $w$  veľmi nízku. Tieto kódy preto nazývame aj riedke. LDPC kódy majú preto veľkú minimálnu vzdialenosť, a teda opravujú veľký počet chýb.

### Generovanie kľúčov

Zjednodušene načrtneme ako funguje kryptosystém s použitím LDPC kódov. Alica vygeneruje náhodnú LDPC kontrolnú maticu  $H$ . Pre riedke LDPC kódy sú známe algoritmy pre efektívne dekódovanie. Nakoľko musíme maticu  $H$  zverejniť ako verejný kľúč, je potrebné ju do istej miery upraviť. Je neprípustné, aby mohol potenciálny útočník efektívne dekódovať so znalosťou matice  $H$ . Preto je z nej potrebné najprv vytvoriť kontrolnú maticu ekvivalentného kódu, ktorá bude hustejšia. Takúto maticu potom môže Alica zverejniť a nebude s ňou možné efektívne dekódovať. Teda vygeneruje náhodnú riedku maticu  $T$ , pričom verejným kľúčom bude matica  $H' = TH$ , ktorá už nie je riedka. Súčasťou verejného kľúča je aj invertibilná matica  $S$ . Súkromný kľúč predstavujú matice  $H$  a  $T$ .

### Šifrovanie

Bob chce zašifrovať správu  $m$ . Vypočíta generujúcu maticu  $G$  z matice  $H'$ . Následne vypočíta maticu  $G' = S^{-1}G$  a šifrovanie prebieha ako  $c = mG'$ . Takýto kryptosystém má podstatne menšie kľúče, nakoľko všetky matice, okrem verejnej matice  $H'$ , sú riedke. Je teda možné ich veľkosť skomprimovať.

### Bezpečnosť

Bezpečnosť je založená na tom, že útočník nie je schopný so znalosťou kontrolnej matice  $H'$  efektívne dekódovať. Taktiež sa predpokladá, že nie je schopný z matice  $H'$  skonštruovať riedku kontrolnú maticu  $H$ , s ktorou by následne vedel efektívne dekódovať.

Avšak ukázalo sa, že použitie LDPC kódov nie je bezpečné [14]. Riadky kontrolnej matice, ktoré majú malú váhu, sú zároveň kódové slová duálneho kódu. Toto vedie k "útoku na krátke vektory", ktorý spočíva v hľadaní kódových slov duálneho kódu s malou váhou. Pomocou týchto kódových slov je potom útočník schopný skonštruovať riedku kontrolnú maticu a dekódovať efektívne. Efektívnosť útoku spočíva v príliš nízkej váhe LDPC matíc.

#### 1.4.2 QC-LDPC kódy

V práci [15] autori prišli s myšlienkou využitia QC(kvázi-cyklických)-LDPC kódov so zmenou matice  $T$  používanej v predošlom návrhu. Snažili sa odstrániť bezpečnostné nedostatky vyplývajúce z návrhu LDPC McElieceovho kryptosystému, a zároveň sa im podarilo

zredukovať veľkosť kľúčov podstatným spôsobom. Najprv si predstavíme QC-LDPC kódy. Vychádzame z definícií uvedených v knihe [12].

### Teória kvázi-cyklických kódov

**Definícia 5.** Binárny  $(n, k)$ -lineárny kód s dĺžkou  $n = mn_0$  a dimenziou  $k = mk_0$  sa nazýva kvázi-cyklický (QC), ak každá rotácia kódového slova o  $n_0$  je opäť kódové slovo.

**Príklad 1.** Zoberme si generujúcu maticu  $G$   $(8, 4)$  binárneho lineárneho kódu. Matica vznikla rotáciou prvého riadku o  $n_0 = 2$ .

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

Ak preusporiadame stĺpce matice  $G$  ako  $1, 1+n_0, 1+n_0+n_0, \dots, 2, 2+n_0, \dots$ , tak dostaneme generujúcu maticu, ktorá pozostáva z dvoch  $4 \times 4$  cyklických blokov.

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

Cyklická matica  $m \times m$  je definovaná nasledovne:

$$C = \begin{bmatrix} c_0 & c_1 & c_2 & \cdots & c_{m-1} \\ c_{m-1} & c_0 & c_1 & \cdots & c_{m-2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_1 & c_2 & c_3 & \cdots & c_0 \end{bmatrix}$$

kde  $c_i$  je prvok vo všeobecnosti z  $F_q$ , my budeme pre zjednodušenie používať  $F_2$ . Kvázi-cyklický  $(n, k)$  kód nad  $F_2$   $C_{QC}$  je ekvivalentný ku  $(mn_0, mk_0)$  kódu s  $mk_0 \times mn_0$  generujúcou maticou pozostávajúcou z  $m \times m$  cyklických matíc  $C_{i,j}$ .

$$C_{QC} = \begin{bmatrix} C_{1,1} & C_{1,2} & \cdots & C_{1,n_0} \\ C_{2,1} & C_{2,2} & \cdots & C_{2,n_0} \\ \vdots & \vdots & \ddots & \vdots \\ C_{k_0,1} & C_{k_0,2} & \cdots & C_{k_0,n_0} \end{bmatrix}$$

Cyklická matica  $C$  sa dá popísať polynómom podľa prvého riadku matice v tvare  $c(x) = c_0 + c_1x + c_2x^2 + \dots + c_{m-1}x^{m-1}$ . Teda existuje mapovanie medzi cyklickou maticou  $C_i$  a polynómom  $c_i(x)$ .

**Veta 1.** Algebra  $m \times m$  cyklických matíc nad poľom  $F$  je izomorfná k algebre polynómov v okruhu  $F[x]/(x^m - 1)$

Uvedme si nasledujúce vety, ktoré platia pre cyklické matice.

**Veta 2.** Násobok dvoch cyklických matíc je cyklická matica.  $C = A \times B$  kde  $c(x) = a(x)b(x) \bmod x^m - 1$

**Veta 3.** Cyklická matica  $C$  má inverziu  $C^{-1}$  ak  $\gcd(c(x), x^m - 1) = 1$ . Inverziu potom môžeme vyjadriť ako  $c^{-1}(x)$  kde  $c(x)c^{-1}(x) = x^m - 1$ .

**Veta 4.** Transpozíciu cyklickej matice  $C^T$  môžeme pomocou polynómu vyjadriť ako  $c_0 + c_{m-1}x + \dots + c_2x^{m-2} + c_1x^{m-1}$ .

Generujúca matica kvázi-cyklického kódu v systematickom redukovanom tvare vyzerá nasledovne:

$$G = \begin{bmatrix} & C'_{1,1} & C'_{1,2} & \dots & C'_{1,n-k} \\ I_{km} & C'_{2,1} & C'_{2,2} & \dots & C'_{2,n-k} \\ & \vdots & \vdots & \ddots & \vdots \\ & C'_{k,1} & C'_{k,1} & \dots & C'_{k,n-k} \end{bmatrix}$$

kde  $I_{km}$  je  $k_0m \times k_0m$  jednotková matica a  $C'_{i,j}$  sú cyklické matice. Kontrolná matica kódu v systematickom redukovanom tvare potom vyzerá nasledovne:

$$H = \begin{bmatrix} & C_{1,1}^T & C_{2,1}^T & \dots & C_{k,1}^T \\ I_{m(n-k)} & C_{1,2}^T & C_{2,2}^T & \dots & C_{k,2}^T \\ & \vdots & \vdots & \ddots & \vdots \\ & C_{1,n-k}^T & C_{2,n-k}^T & \dots & C_{k,n-k}^T \end{bmatrix}$$

### Použitie kvázi-cyklických kódov

Zhrňme si výhody, ktoré prináša použitie kvázi-cyklických kódov v McEliecovom kryptosystéme. Výhodu predstavuje možnosť popísať celú maticu iba za pomoci prvých riadkov jej cyklických blokov. Táto vlastnosť predstavuje nepredstaviteľnú výhodu z pamäťového

hľadiska. Pričom si pri klasickej matici musíme uložiť všetky riadky matice (môžeme pri najlepšom použiť kompresiu), pre kvázi-cyklické matice nám stačí uložiť si iba prvé riadky cyklických blokov. Celú kvázi-cyklickú maticu vieme zrekonštruovať v priebehu výpočtu na základe  $(k - 1)$  rotácií  $n_0$  prvých riadkov. Ďalšiu výhodu predstavuje izomorfná algebra k algebre polynómov modulo  $x^m - 1$  nad  $F_2$ , nakoľko takéto výpočty vieme realizovať efektívne.

## Bezpečnosť

V práci [15] prezentovali použitie kvázi-cyklických LDPC kódov v McEliecovom kryptosystéme. Nakoľko sa tieto riedke kódy stretli s rovnakým problémom ako v [14], tak bolo potrebné zamaskovanie riedkej generujúcej matice kódu  $G$ , ktorá má byť verejným kľúčom. Z tohto dôvodu bol použitý podobný princíp vygenerovania dvoch náhodných invertibilných matíc  $S$  a  $Q$ . Matica  $Q$  bola zvolená ako diagonálna. Verejná generujúca matica sa vypočíta ako  $G' = S^{-1} \times G \times Q^{-1}$ . Takto zvolená štruktúra verejnej generujúcej matice viedla k štruktúrnemu útoku na kryptosystém. Kryptoanalýza bola zverejnená v článku [9].

Preto sa ďalej hľadalo riešenie ako zmeniť štruktúru matice pre kódy použité v McEliecovom kryptosystéme. Riešenie muselo poskytnúť úsporu veľkosti kľúčov a nesmelo viesť k útoku na kryptosystém. V prezentovaných návrhoch sa ukázala byť hlavným nedostatkom riedkosť použitých LDPC kódov, ktorá viedla k nutnosti použitia ďalších matíc pre zhustenie verejnej matice.

## 1.5 McEliecov kryptosystém založený na QC-MDPC kódov

V článku [1] bola prezentovaná myšlienka na zmenu z QC-LDPC kódov na QC-MDPC kódy. QC-MDPC kódy sú hustejšie, a preto by bolo možné zverejniť generujúcu maticu  $G$  bez nutnosti modifikácie za účelom jej zhustenia  $G' = S^{-1} \times G \times Q^{-1}$ . Práve tieto modifikácie sa ukázali v mnohých návrhoch ako bezpečnostné nedostatky, ktoré viedli k útokom na kryptosystém. Ukážme si kvázi-cyklické MDPC kódy.

### 1.5.1 QC-MDPC kódy

LDPC a MDPC (Moderate Density Parity-Check) kódy sa líšia iba vo váhe riadkov  $w$ . Pričom LDPC kódy majú váhu riadkov veľmi malú, MDPC kódy majú váhu riadkov väčšiu. Inak ide principiálne o rovnaké kódy. Keď hovoríme o kvázi-cyklických MDPC kódach, označujeme ich ako QC-MDPC kódy. LDPC kódy vzhľadom na svoju malú váhu

riadkov majú veľkú minimálnu vzdialenosť kódu. Preto opravujú veľa chýb. MDPC kódy majú váhu riadkov väčšiu, a to vedie k tomu, že opravujú menej chýb. Avšak nám pre kryptografické účely stačí opraviť určitý menší počet chýb pre dosiahnutie dostatočnej bezpečnosti.

### 1.5.2 Schéma QC-MDPC McEliecevho kryptosystému

V článku [2] uvádzajú generovanie QC-MDPC kódov nasledovne. Majme  $(n, k, w)$ -QC-MDPC kód s  $n = mn_0$  a  $k = n - m$ . Generujúca matica  $G \in F_2^{k \times n}$  a kontrolná matica  $H \in F_2^{(n-k) \times n}$ . Pričom  $n - k = m$ .

#### Kontrolná matica

Vygenerujeme náhodný vektor  $h \in F_2^n$  dĺžky  $n = mn_0$  a s váhou  $w$ . Potom QC-MDPC kód je definovaný kvázi-cyklickou kontrolnou maticou  $H \in F_2^{m \times n}$ . Matica  $H$  má prvý riadok  $h$  a ostatných  $m - 1$  riadkov pozostáva z  $m - 1$  rotácií riadku  $h$ . Kontrolná matica má tvar:

$$H = \begin{bmatrix} H_0 | H_1 | \cdots | H_{n_0-1} \end{bmatrix}.$$

Každý blok  $H_i$  je  $m \times m$  cyklický blok. Blok  $H_{n_0-1}$  musí byť regulárna matica, aby bolo možné vypočítať jej inverziu a následne pomocou nej zostrojiť generujúcu maticu. Každý blok  $H_i$  má váhu  $w_i$ , pričom platí  $w = \sum_{i=1}^{n_0-1} w_i$ . Váhy  $w_i$  sú rozdistribuované rovnomerne.

#### Generujúca matica

Generujúca maticu  $G$  je v systematickom redukovanom tvare  $G = (I|Q)$ .  $Q$  je možné zostaviť nasledovne:

$$Q = \begin{pmatrix} (H_{n_0-1}^{-1} \cdot H_0)^T \\ (H_{n_0-1}^{-1} \cdot H_1)^T \\ \vdots \\ (H_{n_0-1}^{-1} \cdot H_{n_0-2})^T \end{pmatrix}$$

Vzhľadom na veľké odlišnosti medzi Goppa kódmi a QC-MDPC kódom sa nám mení generovanie kľúča, šifrovanie a dešifrovanie nasledovne:

#### 1. Generovanie kľúčov

- (a) Vygenerujeme kontrolnú maticu  $H \in F_2^{m \times n}$  pre  $(n, k, w)$ -QC-MDPC kód, ktorý opravuje  $t$  chýb.

- (b) Zostavíme príslušnú generujúcu maticu  $G \in F_2^{k \times n}$  v systematickom redukovanom tvare.
  - (c) Verejným kľúčom je  $G$  a súkromným kľúčom je  $H$ .
2. Šifrovanie. Správa  $m \in F_2^k$  sa zašifruje na  $c \in F_2^n$ .
- (a) Vygenerujeme náhodný chybový vektor  $e \in F_2^n$  s váhou  $t$ .
  - (b) Vypočítame  $c = mG + e$ .
3. Dešifrovanie. Nech  $\Psi_H$  je  $t$  chýb opravujúci MDPC dekódovací algoritmus so znalosťou matice  $H$ . Dešifrujeme zašifrovanú správu  $c \in F_2^n$  na  $m \in F_2^k$ .
- (a) Vypočítame  $mG = \Psi_H(mG + e)$
  - (b) Vytiahneme správu  $m$  z prvých  $k$  pozícií  $mG$ .

Môžeme si všimnúť, že schéma vynecháva maticu  $S$  a maticu  $P$  používanú v pôvodnom návrhu McElieceovho kryptosystému s Goppa kódmi pre premiešanie a zamaskovanie generujúcej matice  $G$ . Taktiež nevyužíva žiadne ďalšie matice na zväčšenie váhy generujúcej matice  $G$ , ktorá je verejným kľúčom.

Avšak ako je vidieť, tak s použitím navrhnutej schémy je generujúca matica  $G$  uverejnená v systematickom redukovanom tvare  $G = (I|Q)$ . Pre šifrovanie používame vzťah  $c = mG + e$ . Teda na prvých  $k$  miestach zašifrovanej správy máme  $c' = mI + e'$ , kde  $c'$  a  $e'$  predstavujú prvých  $k$  pozícií z daných vektorov. Teda zašifrovaná správa  $c'$  je rovnaká ako samotná správa  $m$  s pridaním niekoľkých chýb. Preto je nutné pre zverejnenie generujúcej matice  $G$  v systematickom redukovanom tvare používať CCA-2 bezpečnostný padding, ktorý zo zašifrovanej správy  $c$  vytvorí náhodný vektor. [1]

Dôkazy bezpečnosti tejto schémy sú uvedené v práci [1].

### 1.5.3 Dekódovanie QC-MDPC kódov

Vo všeobecnosti budeme hovoriť o dekódovaní LDPC kódov. Pre dekódovanie LDPC kódov existuje viacero algoritmov. Algoritmy sa snažia nájsť rovnováhu medzi výpočtovou zložitou a chybou dekódovania. Problematika dekódovania bola vo viacerých článkoch dobre preštudovaná, nakoľko ide o výpočtovo najzložitejší proces pri kryptosystémoch založených na teórii kódovania. Preto treba tomuto procesu prikladať adekvátny dôraz. Problémom dekódovania sa budeme detailne zaoberať v nasledujúcej podkapitole.



## 1.6 Bit-flipping algoritmus

Budeme sa zaoberať Gallagerovým bit-flipping algoritmom [4]. Ide o iteratívny dekódovací algoritmus pre LDPC kódy. Tento algoritmus je efektívne implementovateľný či už softvérovo alebo hardvérovo. Jeho hardvérovú implementáciu môžeme nájsť v článkoch [2] a [3], kde sa pre jeho použitie rozhodli na základe jeho nízkej výpočtovej zložitosti, ľahkej implementácie na hardvérových platformách a nízkych nárokoch na zdroje. Algoritmus je taktiež odporúčaný v článku [1].

### 1.6.1 Popis algoritmu

Algoritmus vypočíta syndróm správy, ktorá sa dekoduje. Následne sa vypočíta počet nevyhovujúcich rovníc z kontrolnej matice (unsatisfied parity-check equations, ďalej budeme označovať ako upc) pre daný bit správy. Každý bit správy, ktorý nevyhovuje viac ako  $b$  upc je preklopený a syndróm správy je znovu vypočítaný. Tento proces pokračuje pokiaľ syndróm správy nie je nulový vektor, alebo kým sa nedosiahne stanovený počet iterácií. Zložitosť tohto algoritmu je  $O(nwI)$ , pričom  $I$  je priemerný počet iterácií. Všeobecný bit-flipping algoritmus môžeme zapísať formálne ako Algoritmus 1.

---

**Algoritmus 1** Gallagerov bit-flipping algoritmus

---

**Vstup:** kontrolná matica  $H$ , zašifrovaná správa  $c = mG + e$ , množina prahových hodnôt pre všetky iterácie  $B = b_0, \dots, b_{max-1}$ , maximálny počet iterácií  $max$

**Výstup:** Dekódovaná správa  $m$  alebo DECODING\_FAILURE

Vypočítaj syndróm  $s = Hc^T$

**for**  $i = 0 \rightarrow max - 1$  **do**

**for**  $j = 0 \rightarrow length(c)$  **do**

        Vypočítaj  $\#upc$  pre  $c[j]$

**if**  $\#upc \geq b_i$  **then**

            Preklop bit  $c[j]$

**end if**

**end for**

    Aktualizuj syndróm  $s = Hc^T$

**if**  $s = 0$  **then**

**return**  $c$

**end if**

**end for**

**return** DECODING\_FAILURE

---

**Príklad 2.** Na príklade môžeme vidieť ukážku výpočtu  $\#upc$ . Vypočítali sme si syndróm správy (0111), ktorý vznikol ako exkluzívny (XOR) súčet vybraných riadkov kontrolnej matice, podľa správy  $c$  (každý bit správy rozhoduje o tom, či príslušný riadok kontrolnej matice je v exkluzívnom súčte pre syndróm alebo nie je).

$$c \times H^T = s$$

$$(01110110) \times \begin{pmatrix} 1011 \\ 1101 \\ 1110 \\ 0111 \\ 1100 \\ 0110 \\ 0011 \\ 1001 \end{pmatrix} = (1011) \oplus (1110) \oplus (0111) \oplus (0110) \oplus (0011) = (0111)$$

Syndróm prijatej správy je 0111. Nenulové bity sú posledné tri v každom riadku kontrolnej matice sa pozrieme na posledné tri bity. Poznačíme si počet jednotiek (šípkou) ku príslušným bitom kódového slova:

1 ◁◁  
0 ◁◁  
1 ◁◁  
1 ◁◁◁  
0 ◁  
1 ◁◁  
1 ◁◁  
0 ◁

Najviac šípiek má štvrtý bit, preto hodnotu tohto bitu v nasledujúcom kroku algoritmu preklopíme a zopakujeme výpočet syndrómu.

$$(101\textcolor{red}{0}0110) \times \begin{pmatrix} 1011 \\ 1101 \\ 1110 \\ \textcolor{red}{0111} \\ 1100 \\ 0110 \\ 0011 \\ 1001 \end{pmatrix} = (1011) \oplus (1110) \oplus (0110) \oplus (0011) = (0000)$$

Dostaneme nový syndróm 0000. To znamená, že sme našli očakávané kódové slovo. Chybový vektor je ich rozdiel, t.j. 00010000.

Nakoľko pri bit-flipping algoritme môže nastať chyba dekódovania (Decoding Failure), je z kryptografického hľadiska nutné nájsť riešenie pre tento problém. Môžeme uvažovať požiadavku na znovu zašifrovanie správy a opätovného preposlania správy. V prípade použitia šifrovania bez dodatočnej CCA2 bezpečnostnej obálky nastane situácia, kde útočník odchyť dve zhodné správy  $m$  šifrované rovnakým kľúčom (generujúcou maticou kódu), pričom budú v tvare  $c_1 = mG + e_1$  a  $c_2 = mG + e_2$ . Útočník si vie vypočítať  $c_1 \oplus c_2 = (mG + e_1) \oplus (mG + e_2)$ , z čoho dostane  $e_1 \oplus e_2$ . Pri použití dobre zvolenej CCA2 bezpečnostnej obálky takáto situácia nenastane, vzhľadom na to, že zašifrované správy  $c_1$

a  $c_2$  nie sú rozoznatelné od náhodných postupností.

Samozrejme je snaha pri algoritme pravdepodobnosť chyby dekódovania minimalizovať. Chyba dekódovania je predovšetkým závislá od nastavenia prahovej hodnoty  $b$  pre bit-flipping algoritmus. Taktiež vzhľadom na to, že ide o výpočtovo zložitý proces, treba algoritmus optimalizovať vzhľadom na počet potrebných iterácií pre dekódovanie správy. Preto si detailnejšie rozoberieme úpravy bit-flipping algoritmu odporúčané v spomínaných článkoch.

### 1.6.2 Modifikácie bit-flipping algoritmu

Uvedený bit-flipping algoritmus je iba všeobecným algoritmom, ktorý bol zadaný v pôvodnom článku o LDPC kódach [4]. V novších článkoch a reálnych implementáciách sa stretneme s jeho modifikovanými verziami. Všetky však vychádzajú z pôvodného návrhu.

Hlavné rozdiely sa vyskytujú v určení prahovej hodnoty  $b$ . Táto prahová hodnota určuje počet upc, ktorý stačí na to, aby bol daný bit správy pretočený. V prípade nastavenia hodnoty  $b$  príliš nízkej je algoritmus rýchlejší, ale nemusí viesť k správne riešeniu. Rýchlosť závisí od toho, že sa v priebehu jednej iterácie pretočí viacero bitov. Avšak toto riešenie môže konvergovať iba k lokálnemu najlepšiemu riešeniu. Pre prahovú hodnotu  $b$  príliš vysokú je algoritmus pomalší, avšak k správne riešeniu konverguje lepšie. Rozdiely je možno vidieť aj v iných častiach algoritmu. Zvažuje sa, či syndróm aktualizovať po každej iterácii, alebo po každom pretočení bitu správy.

#### Návrh bit-flipping algoritmu podľa [1]

V článku uvažujú tri možné prístupy:

1. *Algoritmus*  $A_1$  používa predpočítané  $b_i$  podľa [4].
2. *Algoritmus*  $A_2$  zvolí prahovú hodnotu  $b_i$  v každej iterácii ako  $Max_{upc}$ , teda maximálny počet upc, podľa článku [17]. Teda v každej iterácii pretočíme jeden alebo malý počet bitov.
3. *Algoritmus*  $A_3$  predstavuje ich vlastné navrhnuté riešenie voľby  $b_i$  ako  $Max_{upc} - \delta$  pre malú hodnotu  $\delta$ . Teda v každej iterácii sa pretočí väčší počet bitov.

*Algoritmus*  $A_1$  je relatívne rýchly, avšak algoritmus vedie k väčšej chybe dekódovania (označme DFR - decoding failure rate). Narozdiel od *algoritmu*  $A_1$ , je *algoritmus*  $A_2$  pomalší, ale vedie k menšiemu DFR. Ich navrhnutý prístup uvedený ako *algoritmus*  $A_3$  by mal využiť výhody prístupov z *algoritmu*  $A_1$  a  $A_2$ . Mal by byť rýchlejší a zároveň by mal mať

rovnakú schopnosť správne dekodovať ako *algoritmus*  $A_2$ . Schopnosť dekodovať je zaručená návrhom, že ak dôjde k chybe dekodovania, hodnota  $\delta$  je znížená o 1 a dekodovanie je spustené znova. Je zrejmé, že ak sa dostaneme až po hodnotu  $\delta = 0$ , máme rovnaký algoritmus ako  $A_2$ . Na základe experimentov dospeli v článku k záveru, že ideálna voľba  $\delta$  je okolo hodnoty 5. Vtedy dosiahli zníženie počtu iterácií zo zhruba 65 na menej ako 10. DFR takýmto prístupom odhadli na najviac  $10^{-7}$ .

### Návrh bit-flipping algoritmu podľa [2]

V článku autori uvažovali 7 modifikácií celého bit-flipping algoritmu. Tieto rôzne verzie naimplementovali a následne otestovali vzhľadom na rýchlosť, počet iterácií potrebných pre dekodovanie správy, a zisťovali pre ne DFR. Uvedme si dva algoritmy, ktoré dosiahli najlepšie výsledky. Taktiež testovali *algoritmus*  $A_3$  z predošlého článku, ktorý použijeme ako referenčnú hodnotu.

1. *Algoritmus*  $A_3$  je popísaný vyššie.
2. *Algoritmus*  $B_1$  vypočíta syndróm správy a vypočíta počty upc (raz za iteráciu  $i$ ). Priamo pri výpočte upc pre  $j$ -ty bit správy zistí, či je počet upc väčší, ako predpočítaná hodnota  $b_i$ . Ak áno, bit správy sa pretočí. Následne spraví aktualizovanie syndrómu správy exkluzívnym súčtom príslušného riadku kontrolnej matice  $h_j$  a aktuálneho syndrómu. Po skončení iterácie sa zistí, či nie je syndróm 0. V prípade, že je, algoritmus sa zastaví, nakoľko bola správa úspešne dekodovaná, inak algoritmus pokračuje. Hodnoty  $b_i$  sú predpočítané podľa návrhu z [4].
3. *Algoritmus*  $B_2$  je podobný ako predošlý *algoritmus*  $B_1$ , avšak po aktualizovaní syndrómu sa zistí, či syndróm nie je 0. V prípade, že je, algoritmus sa zastaví, nakoľko bola správa úspešne dekodovaná. V predošlom algoritme sa táto kontrola vykonávala až po celej iterácii.

Uvedme si dosiahnuté výsledky v nasledujúcej tabuľke.

Algoritmus	počet chýb	čas v $\mu s$	DFR	priemerný počet iterácií
$A_3$	84	26.8	0.00041	5.2964
$A_3$	85	27.3	0.00089	5.3857
$A_3$	86	27.9	0.00221	5.4975
$A_3$	87	28.7	0.00434	5.6261
$A_3$	88	29.3	0.00891	5.7679
$A_3$	89	30.1	0.01802	5.9134
$A_3$	90	31.0	0.03264	6.0677
$B_1$	84	7.02	0.00001	2.4002
$B_1$	85	7.04	0.00003	2.4980
$B_1$	86	7.24	0.00004	2.5979
$B_1$	87	7.53	0.00031	2.6958
$B_1$	88	7.78	0.00093	2.7875
$B_1$	89	8.13	0.00234	2.8749
$B_1$	90	8.31	0.00552	2.9670
$B_2$	84	6.68	0.00000	2.4047
$B_2$	85	6.92	0.00002	2.5000
$B_2$	86	7.11	0.00008	2.5983
$B_2$	87	7.59	0.00039	2.6939
$B_2$	88	7.68	0.00094	2.7912
$B_2$	89	7.99	0.00209	2.8793
$B_2$	90	8.54	0.00506	2.9630

Tabuľka 2 [2] Porovnanie navrhnutých modifikácií bit-flipping algoritmu

Získané výsledky boli pre 1000 náhodných kódov, a pre každý kód bolo uskutočnených 100000 náhodných dekódovaní. Experimenty boli prevedené na Intel Xeon E5345 CPU taktovanom na 2.33GHz.

Z dosiahnutých výsledkov je vidieť, že najlepším algoritmom je práve *Algoritmus  $B_2$* . Pre počet chýb 84 DFR konverguje k 0. Avšak ako uvádzajú autori, aj tak nehovoríme o nulovej pravdepodobnosti. Teda pre použitie v kryptografii treba nájsť riešenie možného neúspešného dekódovania.

### 1.6.3 Parametre kryptosystému

Odporúčané parametre pre McEliecov kryptosystém s QC-MDPC kódmi sú podľa [1] uvedené v tabuľke.

Bitová bezpečnosť	$n_0$	$n$	$r$	$w$	$t$	veľkosť kľúča
80	2	9602	4801	90	84	4801
80	3	10779	3593	153	53	7186
80	4	12316	3079	220	42	9237
128	2	19714	9857	142	134	9857
128	3	22299	7433	243	85	14866
128	4	27212	6803	340	68	20409
256	2	65542	32771	274	264	32771
256	3	67593	22531	465	167	45062
256	4	81932	20483	644	137	61449

Tabuľka 3 [1] Odporúčané parametre pre QC-MDPC McEliecov kryptosystém

Porovnanie veľkostí kľúčov podľa [1] je uvedené v nasledujúcej tabuľke.

Bitová bezpečnosť	QC-MDPC	QC-LDPC	QD-Goppa	Goppa
80	4 801	12 096	20 480	460 647
128	9 857	-	32 768	1 537 536
256	32 771	-	65 536	7 667 855

Tabuľka 4 [1] Porovnanie veľkostí kľúčov pre McEliecov kryptosystém s použitím rôznych kódov

## 2 Návrh riešenia

V tejto kapitole sa budeme venovať samotnému návrhu kryptosystému. Popíšeme si jeho implementáciu z pohľadu využitých algoritmov. Kryptosystém je navrhnutý ako rozšírenie existujúcej knižnice BitPunch [18].

### 2.1 Knižnica BitPunch

Samotný kryptosystém je navrhnutý ako rozšírenie existujúcej kryptografickej knižnice BitPunch, ktorá je naprogramovaná v jazyku C. Táto knižnica je nezávislá na externých knižniciach. Je to open-source riešenie, ktoré je publikované pod licenciou GNU (General Public License). Knižnica momentálne obsahuje implementáciu McElievovho kryptosystému s Goppa kódmi. Obsahuje CCA2 bezpečnostnú konverziu podľa schémy Kobara-Imai CCA2 gamma. Knižnica je dobre zdokumentovaná a je kompatibilná s bežne používanými štandardami ASN1. [19] Zraniteľnosť implementácie McElievovho kryptosystému s Goppa kódmi je preskúmaná z pohľadu postranných kanálov. [20]

### 2.2 Návrh QC-MDPC McElievovho kryptosystému

V tejto podkapitole sa budeme venovať návrhu QC-MDPC McElievovho kryptosystému. Návrh vychádza zo schémy uvedenej v 1.5.2. Preberieme si jednotlivé časti a k nim prislúchajúce návrhy riešenia a použité algoritmy:

- Použité štruktúry
- Generovanie kľúčov
- Šifrovanie
- Dešifrovanie

#### 2.2.1 Použité štruktúry

Pre pochopenie samotného návrhu a implementácie kryptosystému je potrebné uviesť na začiatok spôsob reprezentácie polynómov a kvázi-cyklických matíc. V návrhu používame iba binárne polynómy a binárne matice. Pre efektívnosť výpočtov odlišujeme riedke polynómy (s malou hammingovou váhou) a všeobecné polynómy.



## Reprezentácia všeobecných polynómov

Všeobecné polynómy sú reprezentované poľom bezznamienkových celých čísiel (unsigned integer) o veľkosti 32bitov. Jeden unsigned integer si označme ako element. Takýto jeden element je schopný reprezentovať polynóm stupňa maximálne 31 (teda 32 koeficientov,  $a_0$  až  $a_{31}$ ). Pomocou poľa elementov sme schopní reprezentovať efektívne polynómy ľubovoľného stupňa ( $n$  koeficientov,  $a_0$  až  $a_{n-1}$ ). Polynómy sú do elementov ukladané ako little-endian. To znamená, že koeficient  $a_0$  je uložený ako LSB (najnižší bit) v elemente s indexom 0. Takáto reprezentácia je efektívna a výhodná pre aritmetické operácie s polynómami s vyššou hammingovou váhou. Vynásobenie polynómu s polynómom  $x$  znamená pre našu reprezentáciu polynómov shift doľava o 1. Nakoľko máme polynóm uložený v  $n$  elementoch, ide o  $n$  shiftov doľava a nastavenie najnižšieho bitu na hodnotu najvyššieho bitu predošlého elementu. Spolu cca  $n * 3$  operácií.

## Reprezentácia riedkych polynómov (s malou hammingovou váhou)

Riedke polynómy nie je efektívne reprezentovať všeobecným polynómom. Riedke polynómy reprezentujeme poľom indexov koeficientov v polynóme, ktoré sú 1. Pre polynóm s váhou  $w$  to znamená pole o veľkosti  $w$ . Ukážme si zlepšenie predošlého príkladu násobenia polynómu s polynómom  $x$ . Pre reprezentáciu riedkych polynómov znamená toto násobenie zvýšenie hodnoty indexov koeficientov o 1 a zmodulovanie stupňom polynómu. Teda zhruba  $w*2$  operácií. Pre reálne parametre kryptosystému s 80-bitovou bezpečnosťou ( $n = 151$ ,  $w = 45$ ) je to 90 operácií oproti 453 operáciám pre reprezentáciu všeobecných polynómov.

## Kvázi-cyklická matica

Vychádzame z teórie kvázi-cyklických matíc z 1.4.2. Analýza ukázala, že pre účely McElieceovho kryptosystému potrebujeme vytvoriť reprezentáciu kvázi cyklickej matice, ktorá bude mať  $k_0 = 1$  (počet cyklických matíc v stĺpci je 1) alebo  $n_0 = 1$  (počet cyklických matíc v riadku je 1). Každá cyklická matica bude reprezentovaná polynómom. Kvázi-cyklická matica obsahuje viac cyklických matíc, bude teda reprezentovaná poľom polynómov. Pre generujúcu maticu  $G$  budeme používať všeobecné polynómy, nakoľko matica  $G$  je hustá. Naopak pre kontrolnú maticu  $H$  budeme používať riedke polynómy. Pre získanie príslušného riadku matice cyklicky shiftujeme potrebné polynómy.

Pre efektívnosť výpočtov a zároveň zníženie pamäťových nárokov sa používa v celom návrhu reprezentácia kvázi-cyklických (a cyklických) matíc ako polynómov.

Možnosť izomorfnej algebry medzi kvázi-cyklickými maticami a polynómami je popísaná v 1.4.2.

### 2.2.2 Generovanie kľúčov

Návrh generovania kľúčov je popísaný algoritmom 2.

---

#### Algoritmus 2 Generovanie kľúčov

---

**Vstup:** Parametre:  $n_0$  - počet cyklických matíc v kontrolnej matici,  $m$  - dĺžka cyklickej matice,  $w$  - váha riadku kontrolnej matice,  $t$  - počet chýb

**Výstup:** Kontrolná matica  $H$ , generujúca matica  $G$ , alebo GENERATION\_ERROR

```

function GENERATE_KEYS( $n_0, m, w, t$ )
    for  $i \leftarrow 0 \rightarrow n_0 - 2$  do
        Vygeneruj náhodný polynóm  $H[i]$  s váhou  $w[i]$ 
    end for

    Generovanie posledného polynómu:
    Vygeneruj náhodný polynóm  $H[n_0 - 1]$  s váhou  $w[n_0 - 1]$ 
    if  $\gcd(H[n_0 - 1], x^m - 1) = 1$  then
        Vypočítaj  $H^{-1}[n_0 - 1]$ 
    else
        go to Generovanie posledného polynómu
    end if

    Vytvor kontrolnú maticu  $H$  z polynómov  $H[0]$  až  $H[n_0 - 1]$ 
    for  $i \leftarrow 0 \rightarrow n_0 - 2$  do
         $G[i] \leftarrow H[i] \times H^{-1}[n_0 - 1]$ 
    end for

    Vytvor generujúcu maticu  $G$  z polynómov  $G[0]$  až  $G[n_0 - 2]$ 
    if  $G \times H^T \neq 0$  then
        return GENERATION_ERROR
    end if
    return  $G^T, H^T$ 
end function

```

---

Pre výpočet  $\gcd(H[n_0 - 1], x^m - 1)$  a inverzného polynómu  $H^{-1}[n_0 - 1]$  sa používa rozšírený Euklidov algoritmus (XGCD). Euklidov algoritmus slúži na výpočet najväčšieho spoločného deliteľa (gcd) dvoch polynómov a, b. Jeho rozšírená verzia je schopná okrem

gcd vypočítat aj koeficienty  $s$  a  $t$  pre Bézoutovú rovnosť  $at + bs = \gcd(a, b)$ , kde ak  $\gcd(a, b) = 1$ , tak polynóm  $t = a^{-1}$ . Algoritmus je popísaný 3. [21]

---

**Algoritmus 3** rozšírený Euklidov algoritmus

---

**Vstup:** polynómy  $a, b$

**Výstup:**  $\gcd, s, t$

```

function XGCD(a,b)
     $s \leftarrow 0$ ;  $old\_s \leftarrow 1$ 
     $t \leftarrow 1$ ;  $old\_t \leftarrow 0$ 
     $r \leftarrow b$ ;  $old\_r \leftarrow a$ 
    while  $r \neq 0$  do
         $quotient \leftarrow old\_r \text{ div } r$ 
         $(old\_r, r) \leftarrow (r, old\_r - quotient * r)$ 
         $(old\_s, s) \leftarrow (s, old\_s - quotient * s)$ 
         $(old\_t, t) \leftarrow (t, old\_t - quotient * t)$ 
    end while
     $\gcd \leftarrow old\_r$ 
     $s \leftarrow old\_s$ 
     $t \leftarrow old\_t$ 
    return  $\gcd, s, t$ 
end function

```

---

Potom algoritmus na výpočet inverzného polynómu zapíšeme nasledovne 4.

---

**Algoritmus 4** Výpočet inverzného polynómu

---

**Vstup:** polynómy  $a, mod$

**Výstup:**  $a^{-1}$  alebo NOT\_EXIST

```

function INVERSION_POLY(a,mod)
     $(\gcd, s, t) \leftarrow xgcd(a, mod)$ 
    if  $\gcd = 1$  then
         $a^{-1} \leftarrow t$ 
        return  $a^{-1}$ 
    else
        return NOT_EXIST
    end if
end function

```

---

### 2.2.3 Šifrovanie

Šifrovanie je realizované algoritmom 5.

---

**Algoritmus 5** Šifrovanie

---

**Vstup:** Parameter:  $t$  - počet chýb. Generujúca matica  $G$ , správa  $m$

**Výstup:** Zašifrovaný text  $c$

```
function ENCRYPT( $G, m$ )  
     $c \leftarrow m \times G$   
    Vygeneruj náhodný chybový vektor  $e$  s váhou  $t$   
     $c \leftarrow c \oplus e$   
    return  $c$   
end function
```

---

Algoritmus šifrovania je veľmi jednoduchý, nakoľko ide o vynásobenie správy  $m$  maticou  $G$  a pridanie náhodnej chyby.

### 2.2.4 Dešifrovanie

Dešifrovanie môžeme popísať algoritmom 6.

---

**Algoritmus 6** Dešifrovanie

---

**Vstup:** Kontrolná matica  $H$ , zašifrovaná správa  $c$

**Výstup:** Správa  $m$

```
function DECRYPT( $H, c$ )  
     $decode(H, c, decoded)$   
     $m \leftarrow$  prvých  $k$  bitov z  $decoded$   
    return  $m$   
end function
```

---

Je vidieť, že podstata celého algoritmu dešifrovania je vo funkcii `decode`, ktorá sa stará o dekódovanie zašifrovanej správy. Princípom dekódovania je odstránenie chybového vektora, ktorý je vnesený počas šifrovania. Pri znalosti kontrolnej matice  $H$  a pridania počtu chýb, ktorý zodpovedá charakteristike kódu, ide o riešiteľný problém. Ako bolo uvedené v 1.6, pre dekódovanie využijeme rýchly a efektívny algoritmus s názvom bit-flipping.

Vzhľadom na analýzu sme sa rozhodli o implementáciu dvoch verzií bit-flipping algoritmu, ktoré sa ukazujú ako najrýchlejšie a vykazujú najmenšiu chybu dekódovania.

Každá z vybraných verzií algoritmu má svoju výhodu. Ďalej v kapitole 3 tieto algoritmy podrobíme testom a porovnáme vzhľadom na zadefinované kvalitatívne parametre. Podľa výsledkov porovnania vyberieme lepšiu z týchto dvoch algoritmov a zanalyzujeme jeho možné nedostatky. Taktiež na základe testov navrhujeme zlepšenia nedostatkov a otestujeme voči predpokladu, aby sme zistili, či sme dosiahli zlepšenie.

### Algoritmus A3

Prvým algoritmom je algoritmus A3 3. Algoritmus je parametrizovateľný pomocou parametra  $\delta$ . Algoritmus je rýchly a dosahuje malú chybu DFR. Avšak rýchlosť a chyba DFR závisia od voľby parametra. Algoritmus si môžeme popísať nasledovne 7.

---

#### Algoritmus 7 Bitflipping A3

---

**Vstup:** Kontrolná matica  $H$ , zašifrovaná správa  $c$

**Výstup:** Dekódovaná správa *decoded*, alebo DECODING\_FAILURE

```

function DECODE_A3( $H, c$ )
    syndrom  $\leftarrow$  calculate_syndrom( $H, c$ )
    if syndrom = 0 then
        return  $c$ 
    end if
    for iter  $\leftarrow$  0  $\rightarrow$  max_iter do
        for bit  $\leftarrow$  0  $\rightarrow$  length( $c$ ) do
            upc[bit]  $\leftarrow$  calculate_upc( $H$ [bit], syndrom)
        end for
        for bit  $\leftarrow$  0  $\rightarrow$  length( $c$ ) do
            if upc[bit]  $\geq$  max(upc)  $- \delta$  then
                Preklop príslušný bit v  $c$ 
                syndrom  $\leftarrow$  calculate_syndrom( $H, c$ )
                if syndrom = 0 then
                    return  $c$ 
                end if
            end if
        end for
    end for
    return DECODING_FAILURE
end function

```

---

Algoritmus najprv vypočíta syndróm dekodovanej správy. Následne pre nastavený počet iterácií opakuje dva kroky:

1. Vypočíta  $\#upc$  pre každý bit dekodovanej správy (pre nájdenie maximálneho počtu  $upc$  v danej iterácii)
2. Ak je  $\#upc$  pre daný bit správy väčší alebo rovný ako maximum v danej iterácii (mínus  $\delta$ ), tak bit správy preklopí

Algoritmus je možné pri vrátení chyby `DECODING_FAILURE` iterovať so znižovaním parametra  $\delta$ . Takýto upravený algoritmus by mal dosahovať veľmi nízke hodnoty DFR.

### **Algoritmus B2**

Druhým algoritmom je algoritmus B2 3. Algoritmus vyžaduje predpočítanie a nastavenie hodnôt  $b_i$ , podľa ktorých sa rozhoduje, či vykoná pretočenie daného bitu. Ukážme si algoritmus 8.

---

**Algoritmus 8** Bitflipping B2

---

**Vstup:** Kontrolná matica  $H$ , zašifrovaná správa  $c$

**Výstup:** Dekódovaná správa *decoded*, alebo DECODING\_FAILURE

```
function DECODE_B2( $H, c$ )  
     $syndrom \leftarrow calculate\_syndrom(H, c)$   
    if  $syndrom = 0$  then  
        return  $c$   
    end if  
    for  $iter \leftarrow 0 \rightarrow max\_iter$  do  
        for  $bit \leftarrow 0 \rightarrow length(c)$  do  
             $upc \leftarrow calculate\_upc(H[bit], syndrom)$   
            if  $upc \geq b[iter]$  then  
                Preklop príslušný bit v  $c$   
                 $syndrom \leftarrow calculate\_syndrom(H, c)$   
                if  $syndrom = 0$  then  
                    return  $c$   
                end if  
            end if  
        end for  
    end for  
    return DECODING_FAILURE  
end function
```

---

Keďže algoritmus B2 má predpočítané hodnoty  $b_i$ , tak je možné vynechať krok 1 z algoritmu A3. Preto je prístup tohto algoritmus rýchlejší. Avšak predpokladáme, že upravený algoritmus A3 (s iterovaním pri znižovaní parametra  $\delta$ ) bude dosahovať nižšie hodnoty DFR.

### 3 Výsledky testov

Kapitola je zameraná na otestovanie implementácie McEliecovho kryptosystému s QC-MDPC kódmi. V kapitole analýza sme načrtli schému tohto kryptosystému. V návrhu riešenia sme navrhli konkrétne algoritmy, ktoré boli implementované v jazyku C. V tejto kapitole sa budeme venovať:

- optimalizácií samotného kódu
- porovnaniu dvoch navrhnutých bit-flipping algoritmov
- návrhu zlepšenia bit-flipping algoritmu a otestovaniu navrhnutého riešenia
- porovnaniu McEliecovho kryptosystému s Goppa kódmi, McEliecovho kryptosystému s QC-MDPC kódmi a štandardným kryptosystémom RSA
- porovnaniu našej implementácie McEliecovho kryptosystému s QC-MDPC kódmi a inej implementácie

#### 3.1 Testovacie prostredie

Na úvod kapitoly si zadefinujeme testovacie prostredie, na ktorom sa vykonávali nasledujúce testy.

- Operačný systém: Ubuntu 13.10, Saucy Salamander, 64bit (bežiaci pod Oracle VM VirtualBox 4.3.6)
- Systémová konfigurácia:
  - Procesor: Intel Core i7-3610QM @ 2.30GHz (jedno jadro pridelené Ubuntu)
  - Pamäť: 8.00GB (2.00GB pridelené Ubuntu)
  - Kompilátor: gcc verzia 4.8.1

Pre testy bola použitá odporúčaná voľba parametrov kryptosystému z 1.6.3 pre 80-bitovú bezpečnosť,  $n_0 = 2$ ,  $n = 9602$ ,  $r = 4801$ ,  $w = 90$ ,  $t = 84$ . Program je kompilovaný s optimalizačným prepínačom -O3. Na meranie rýchlosti je použitá funkcia gettimeofday() z knižnice sys/time.h.



## 3.2 Optimalizácia

Budeme testovať pôvodnú implementáciu a pomocou kompilovania s prepínačom `-pg` a následnej analýzy výstupov pomocou `gprof` sa budeme snažiť odhaliť funkcie, ktoré je potrebné optimalizovať. Pre nasledujúce testy sa použilo generovanie 1000 kľúčov, šifrovanie 1000 náhodných správ a dešifrovanie 1000 správ. Zameriame sa na priemernú rýchlosť vykonania 1 operácie. Pre sériu testov v tejto podkapitole sa používal algoritmus A3 s parametrom  $\delta$  nastaveným na hodnotu 5.

### Pôvodný návrh

Pôvodný návrh bol implementovaný s použitím všeobecných polynómov a všeobecných matíc.

operácia	pôvodný návrh
generovanie kľúčov	150 ms
šifrovanie	40 ms
dešifrovanie	768 ms

Tabuľka 5 Časy trvania operácií pre pôvodný návrh

Ako je vidieť z dosiahnutých výsledkov, dešifrovanie je pomalé. Práve táto skutočnosť nás viedla k prerobeniu kontrolnej matice na samostatnú štruktúru s riedkymi polynómami ako je uvedené v 2.2.1.

### Riedka kontrolná matica

Porovnáme dosiahnuté výsledky po zmene implementácie kontrolnej matice  $H$ .

operácia	pôvodný návrh	$H$ matica
generovanie kľúčov	150 ms	132 ms
šifrovanie	40 ms	40 ms
dešifrovanie	768 ms	59 ms

Tabuľka 6 Časy trvania operácií pre zmenu implementácie kontrolnej matice  $H$

Z výsledkov je vidieť enormné zrýchlenie dešifrovania. Taktiež sa zlepšil čas generovania kľúčov. Následne sme zanalyzovali pomocou profilovania najvyťažujúcejšie funkcie. Išlo

o funkcie BPU\_gf2PolyAdd a BPU\_setPolyDeg. Ide o funkciu sčítavanie všeobecných polynómov a nastavenie stupňa polynómu s upravením alokovanej pamäte.

### Optimalizácia sčítavania polynómov a nastavenia stupňa polynómu

Po optimalizácii týchto dvoch funkcií sme dosiahli tieto výsledky.

operácia	pôvodný návrh	$H$ matica	opt. 1
generovanie kľúčov	150 ms	132 ms	67 ms
šifrovanie	40 ms	40 ms	29 ms
dešifrovanie	768 ms	59 ms	54 ms

Tabuľka 7 Časy trvania operácií pre optimalizáciu funkcií

Výsledky testov ukázali zlepšenie predovšetkým pri generovaní kľúčov a šifrovaní. Ďalšou možnosťou zlepšenia je kontrola po vygenerovaní generujúcej matice  $G$  a kontrolnej matice  $H$ . Kontroluje sa či vychádza vzťah  $G \times H^T = 0$ .

### Optimalizácia kontroly $G \times H^T = 0$

Uvedme si výsledky po optimalizovaní testovania vzťahu  $G \times H^T = 0$ .

operácia	pôvodný návrh	$H$ matica	opt. 1	opt. 2
generovanie kľúčov	150 ms	132 ms	67 ms	59 ms
šifrovanie	40 ms	40 ms	29 ms	29 ms
dešifrovanie	768 ms	59 ms	54 ms	54 ms

Tabuľka 8 Časy trvania operácií pre optimalizáciu kontroly  $G \times H^T = 0$

Podarilo sa nám zlepšiť generovanie kľúčov. Vzhľadom na pomerne pomalé šifrovanie sme sa rozhodli podrobne preskúmať dôvod spomalenia. Po zanalyzovaní, viacerých pokusoch a testovaní sa nám podarilo nájsť príčinu spomalenia. Išlo o neefektívne implementovaný náhodný generátor polynómov a vektorov. Po zmene implementácie sme dosiahli tieto výsledky.

operácia	pôvodný návrh	$H$ matica	opt. 1	opt. 2	opt. 3
generovanie kľúčov	150 ms	132 ms	67 ms	59 ms	19 ms
šifrovanie	40 ms	40 ms	29 ms	29 ms	1.1 ms
dešifrovanie	768 ms	59 ms	54 ms	54 ms	54 ms

Tabuľka 9 Časy trvania operácií po zmene algoritmu generovanie náhodných polynómov.

Z profilovania nevidieť žiadne ďalšie možnosti väčšieho zlepšenia v algoritmoch.

Pre šifrovanie je údaj rýchlosti relatívna hodnota. Dôležitá je priepustnosť šifrovania, ktorá je závislá na rýchlosti a veľkosti dát, ktoré sa šifrujú. Pre nastavené parametre kryptosystému ide o správu o veľkosti 0.586 KB. Najrýchlejšie dosiahnuté šifrovanie zašifruje správu tejto veľkosti za 1.1 ms. Z toho je priepustnosť šifrovania 533 KB/s.

### 3.3 Porovnanie bit-flipping algoritmov

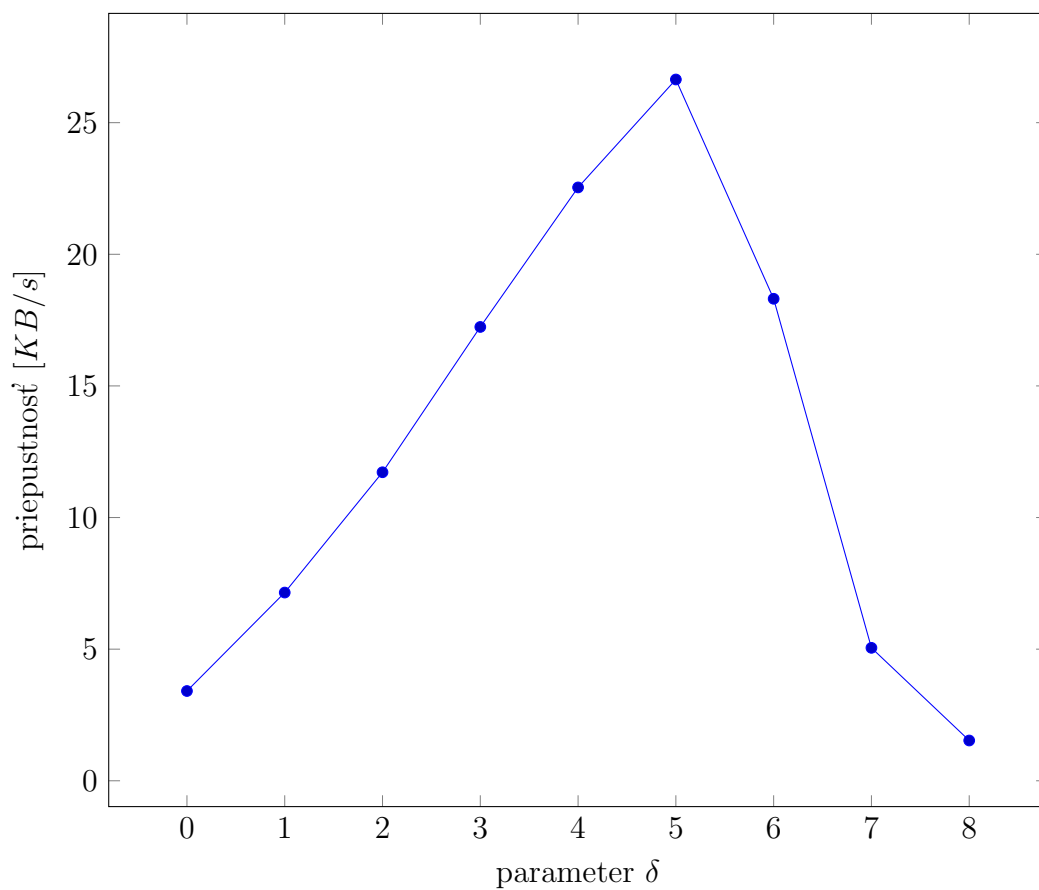
Táto podkapitola je venovaná otestovaniu dvoch vybraných kandidátov na dekodovací bit-flipping algoritmus pre McEliecov kryptosystém s QC-MDPC kódmi. Ako sme uviedli v *Návrhu riešenia*, budeme sa zaoberať testovaním algoritmu A3 7 a algoritmu B2 8.

#### 3.3.1 Algoritmus A3

Ako prvé nás zaujíma ideálne nastavenie parametra  $\delta$ , ktorý parametrizuje tento algoritmus. Pre nasledujúce testy sme menili parameter  $\delta$  z hodnoty 0 po hodnotu 8. Pre každú voľbu parametra sme dali vygenerovať kľúče, zašifrovali 100 náhodných správ a dešifrovali. Sledovali sme priemerný počet iterácií v algoritme na dešifrovanie správ, rýchlosť dešifrovania a hodnotu DFR.

parameter $\delta$	priemerný počet iterácií	čas dešifrovania	priepustnosť	DFR
0	41	172 ms	3.41 KB/s	0.00 %
1	20	82 ms	7.15 KB/s	0.00 %
2	12	50 ms	11.72 KB/s	0.00 %
3	8	34 ms	17.24 KB/s	0.00 %
4	7	26 ms	22.54 KB/s	0.00 %
5	5	22 ms	26.64 KB/s	0.00 %
6	8	32 ms	18.31 KB/s	0.00 %
7	27	116 ms	5.05 KB/s	0.00 %
8	90	383 ms	1.53 KB/s	0.00 %

Tabuľka 10 Porovnanie voľby parametra  $\delta$  pre algoritmus A3



Graf 1 Porovnanie voľby parametra  $\delta$  pre algoritmus A3

Z dosiahnutých výsledkov vidieť, že ideálna voľba parametra  $\delta$  je 5. Algoritmus dosahuje pri tomto nastavení najvyššiu rýchlosť. Tieto testy však nie sú vhodné pre samotné sledovanie DFR.

Pre relevantnejšie výsledky z pohľadu DFR sme navrhli test, kde sa generuje 100 kľúčov, a pre každý kľúč sa vykoná zašifrovanie 1000 náhodných správ a následne sa dešifrujú. Teda spolu 100 000 dešifrovaní. Pri tomto teste sledujeme hodnotu DFR. Našou snahou je minimalizácia hodnoty DFR. Testujeme nasledovné dva prípady:

1. algoritmus A3 s nastaveným parametrom  $\delta$  na 5
2. algoritmus A3 s nastaveným parametrom  $\delta$  na 5, a v prípade neúspešného dešifrovania hodnotu parametru  $\delta$  znižujeme až po 0

nastavenie parametra $\delta$	DFR
5	0.03 %
$5 \rightarrow 0$	0.00 %

Tabuľka 11 Hodnoty DFR v závislosti od parametra  $\delta$

Z dosiahnutých výsledkov je zrejmé, že prístup znižovania parametra  $\delta$  v prípade neúspešného dekódovania vedie k minimalizácii DFR. Teda kombinujeme rýchlosť, ktorú dosahuje voľba parametra  $\delta = 5$  a nízku chybovosť pri nižšej hodnote  $\delta$ .

### 3.3.2 Algoritmus B2

Ďalším bit-flipping algoritmom, ktorý by mal dešifrovať rýchlejšie, avšak pri potenciálne vyššej chybovosti, je algoritmus B2. Vykonali sme 100 000 dešifrovaní za rovnakých podmienok ako pri predošlom teste. Algoritmus ma predpočítané hodnoty  $B[i]$  nasledovné:

i	0	1	2	3	4
$B[i]$	28	26	24	22	20

Tabuľka 12 Predpočítané hodnoty  $B[i]$  pre algoritmus B2

Pre porovnanie uvádzame výsledky oboch algoritmov v jednej tabuľke.

algoritmus	počet iterácií	čas dešifrovania	priepustnosť	DFR
algoritmus A3 s $\delta = 5 \rightarrow 0$	5.35	23 ms	25.48 KB/s	0.0000 %
algoritmus B2	2.46	7 ms	83.71 KB/s	0.0001 %

Tabuľka 13 Porovnanie algoritmu A3 a algoritmu B2

Je zrejmé, že algoritmus B2 je rýchlejší. Avšak ako sme predpokladali, nebude dosahovať rovnako nízke hodnoty DFR.

### 3.3.3 Návrh dešifrovania

Po zhodnotení predošlých testov sme si potvrdili predpoklad, že algoritmus A3 má nižšie DFR. Hodnota DFR konverguje k 0 pre nami testovaný počet 100 000 dešifrovaní. Algoritmus B2 napriek tomu poskytuje viac ako 3-násobne rýchlejšie dešifrovanie pri malej hodnote DFR (okolo 0,0001%). Spojením týchto dvoch algoritmov vieme získať rýchle dešifrovanie a zanedbateľnú úroveň DFR. Preto navrhujeme dešifrovanie s použitím bit-flipping algoritmu B2. V prípade neúspešného dekódovania pomocou tohto algoritmu opakovanne spustíme dekódovanie pomocou algoritmu A3 s otestovaným parametrom  $\delta$  s hodnotou od 5 do 0.

Pre overenie nízkeho DFR otestujeme nasledujúce prípady:

1. algoritmus B2
2. nami navrhnuté dešifrovanie

Testy prevedieme na množine 1000 kľúčov, kde pre každý kľúč zašifrujeme 1000 náhodných správ a pokúsime sa ich dešifrovať. To nám dáva spolu 1 000 000 dešifrovaní. Uved'me si výsledky v tabuľke.

algoritmus	počet iterácií	čas dešifrovania	priepustnosť	DFR
algoritmus B2	2.46	7 ms	83.71 KB/s	0.00009 %
nami navrhnutá kombinácia	2.46	7 ms	83.71 KB/s	0.00000 %

Tabuľka 14 Porovnanie algoritmu B2 a nami navrhnutej kombinácie algoritmov B2 a A3

Testom sa náš predpoklad zlepšenia DFR potvrdil. Nami navrhnutá kombinácia dvoch algoritmov je rýchla a zároveň má hodnotu DFR konvergujúcu k nule. Avšak aj v tomto

prípade treba uvažovať situáciu, že sa správu nepodarí dešifrovať. V takom prípade je nutné požiadať o znovu zašifrovanie správy. Ak je správne použitá CCA-2 bezpečnostná konverzia (ktorá je pri našom návrhu nevyhnutná), tak útočník nezíska žiadne informácie, nakoľko ide o dve náhodné správy.

### 3.4 Porovnanie kryptosystémov

Motivácia tejto práce je zanalyzovať možnosti minimalizácie post-quantového McElieceovho kryptosystému s Goppa kódmi. McElieceov kryptosystém s QC-MDPC kódmi je vhodný kryptosystém, ktorý vyhovuje kryptografickej bezpečnosti, a máme tento kryptosystém úspešne naimplementovaný. Je potrebné porovnať tieto dva kryptosystémy. Pre porovnanie sme použili kryptosystémy parametrizované pre rovnakú úroveň bezpečnosti, a to 80 bitov. Implementácia McElieceovho kryptosystému s Goppa kódmi je použitá z knižnice BitPunch. Hlavné sledované kritéria pre porovnanie vychádzajú z potreby zmenšiť veľkosť kľúčov a zmenšiť pamäťové nároky kryptosystému. Preto sa predovšetkým zameriavame na spotrebu pamäte pri generovaní kľúčov, šifrovaní a dešifrovaní. Taktiež sledujeme rýchlosť generovania, šifrovania a dešifrovania.

Navyše pre lepšie porovnanie si ukážeme aj dosiahnuté hodnoty s bežne používaným asymetrickým kryptosystémom RSA. Implementáciu kryptosystému RSA sme použili z open-source knižnice OpenSSL. RSA je parametrizovaný na 1024 bitov, čo predstavuje ekvivalentnú symetrickú mieru bezpečnosti na úrovni 80 bitov, rovnako ako zvyšné dva porovnávané McElieceove kryptosystémy.

Na úvod si uveďme parametre testovaného McElieceovho kryptosystému s Goppa kódmi.

Parameter	Význam	Veľkosť (Bit)
$m$	Stupeň Galoisovho poľa	11
$t$	Počet opraviateľných chýb	50
$n$	Dĺžka kódu $n = 2^m$	2048
$k$	Stupeň kódu $k = n - mt$	1496
$l$	Hashovacia hodnota SHA-512	512
$m$	Otvorený text	512
$z$	Šifrovaný text	3072
$\mathbf{R}^T$	Verejný kľúč	$1498 \times 550$
$\mathbf{P}$	Permutačná matica	$2048 \times 11$
$g(x)$	Goppov polynóm	$51 \times 11$
$\mathbf{H}$	Kontrolná matica	$(50 \times 11) \times 2048$

Tabuľka 15 [18] Parametre McEliecovho kryptosystému s Goppa kódmi

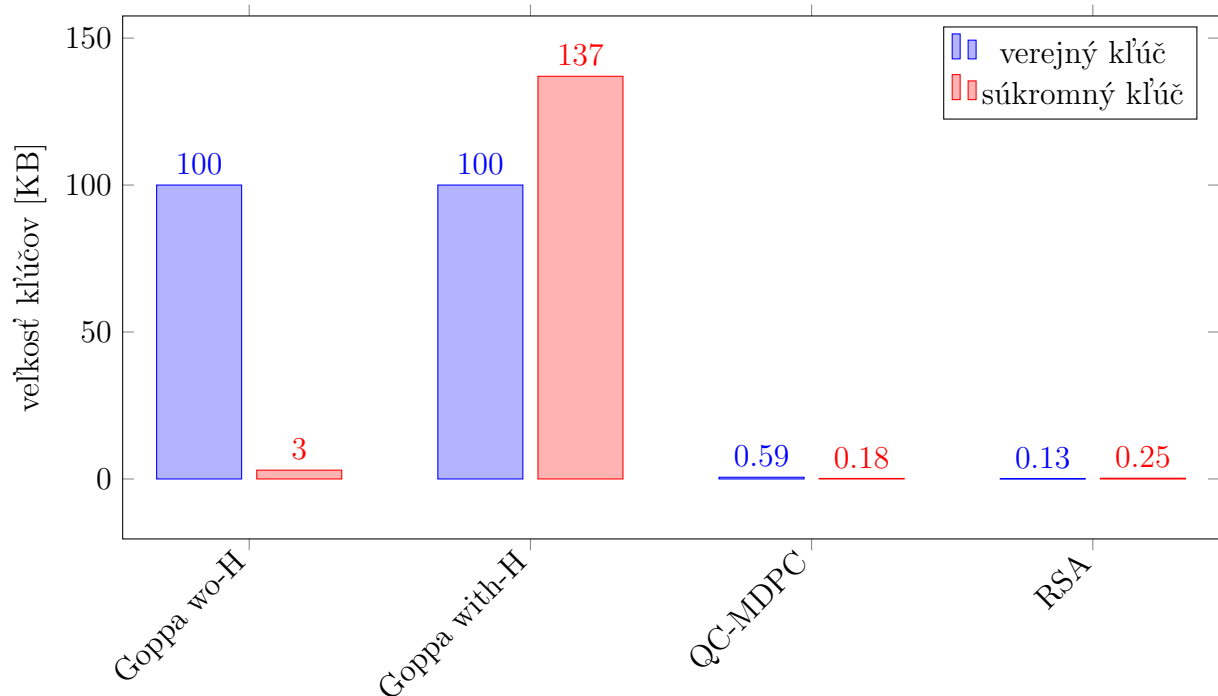
McEliecov kryptosystém s danými nastavenými parametrami je schopný zašifrovať najviac 0.183 KB správu. Knížnica BitPunch poskytuje dve verzie McEliecovho kryptosystému s Goppa kódmi. Jedna je s predpočítanou maticou  $H$  pre dešifrovanie (označme Goppa with-H), a v druhej sa matica  $H$  počíta počas dešifrovania (označme Goppa wo-H). Pre otestovanie sme vygenerovali 1000 kľúčov, náhodne vygenerovali správu, zašifrovali a následne dešifrovali. V tabuľke uvádzame priemerné hodnoty z týchto 1000 opakovaní. RSA dokáže zašifrovať pri zvolených parametroch najviac 0.125 KB správu. Spotrebu pamäte sme testovali pomocou nástroja valgrind s prepínačom `-tool=massif`.



kryptosystém	Goppa wo-H	Goppa with-H	QC-MDPC	RSA
maximálna veľkosť správy [KB]	0.183	0.183	0.586	0.125
veľkosť verejného kľúča [KB]	100	100	0.586	0.129
veľkosť súkromného kľúča [KB]	3	137	0.176	0.25
spotreba pamäte [KB]	180	378	7	8
rýchlosť generovania [ms]	702	703	19	33
rýchlosť šifrovania [ms]	0.062	0.065	1.1	0.051
priepustnosť šifrovania [KB/s]	2951	2815	533	2451
rýchlosť dešifrovania [ms]	60	3.5	7	1.1
priepustnosť dešifrovania [KB/s]	3	52	84	116

Tabuľka 16 Porovnania kryptosystémov

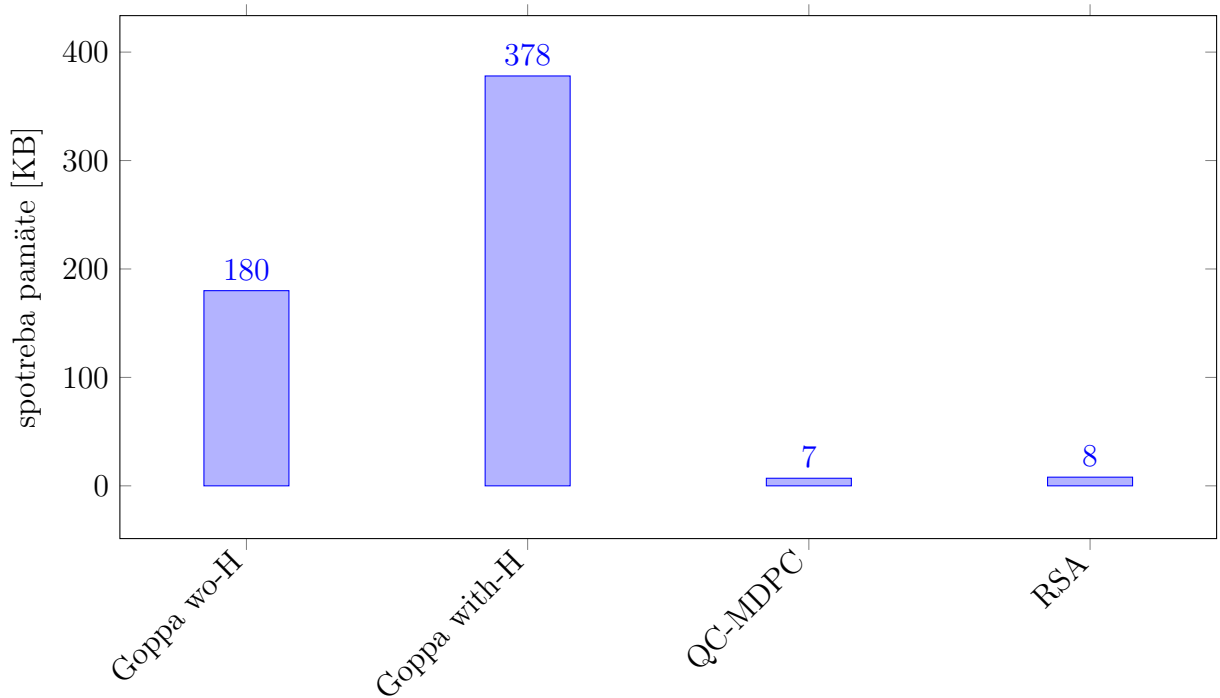
Dosiahnuté výsledky zanalyzované v tabuľke potvrdzujú dosiahnutie našich cieľov. Porovnajme si McEliecov kryptosystém s Goppa kódmi a nami implementovaný kryptosystém s QC-MDPC kódmi, a pozrime sa aj na výsledky dosiahnuté s kryptosystémom RSA.



Graf 2 Veľkosť kľúčov pre rôzne kryptosystémy

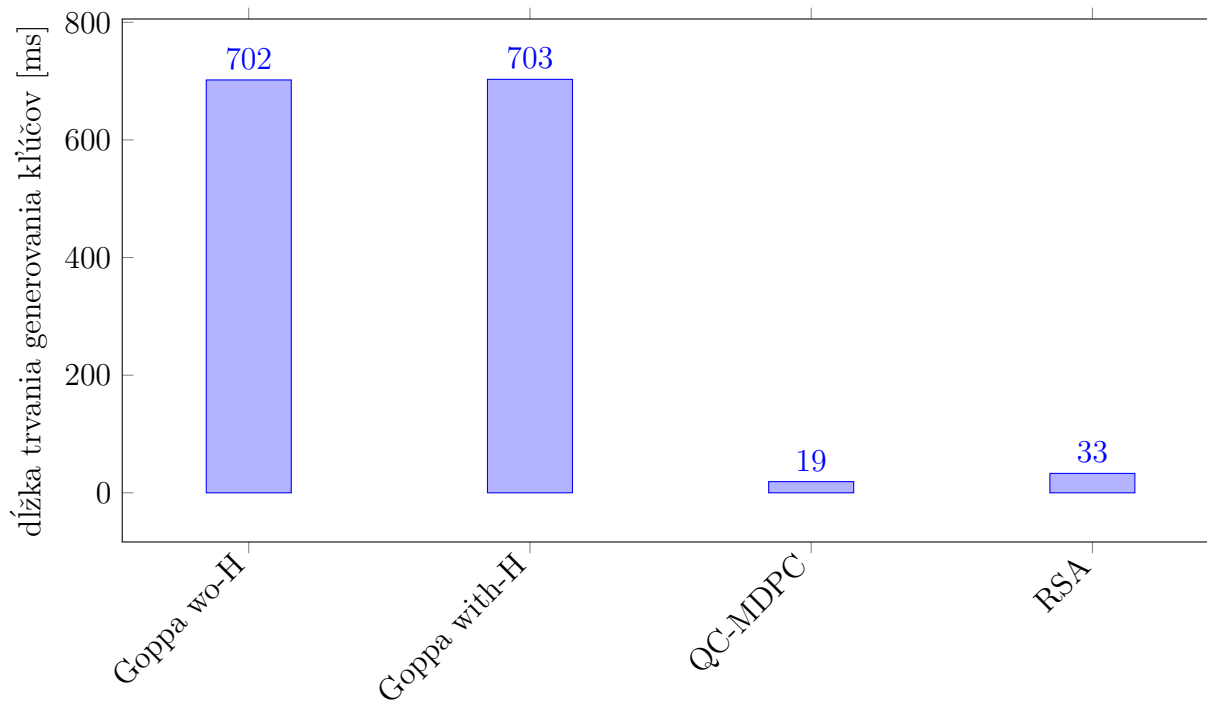
Ak hovoríme o veľkosti verejného kľúča, dosiahli sme minimalizáciu zo 100 KB na veľkosť 0.6 KB. Stále je to niekoľko násobne viac ako v prípade RSA, ktoré má verejný kľúč veľkosti zhruba 0.1 KB. Avšak je to výrazné zlepšenie, čo sa týka asymetrických kryptosystémov založených na teórii kódovania. Treba si uvedomiť, že táto minimalizácia je dosiahnutá vďaka použitým kvázi-cyklickým maticiam. Nepotrebuje sa ukladať celú maticu, ale stačí nám uložiť si jej prvé riadky cyklických matíc a počas šifrovania / dešifrovania si vieme zvyšné riadky matice dopočítať.

Ak si porovnáme veľkosti súkromného kľúča, tak s použitím QC-MDPC kódov v McElieceovom kryptosystéme sme znížili potrebnú veľkosť oproti Goppa kódom z 3 KB (resp. 137 KB v prípade predpočítanej matice  $H$ ) na iba 0.18 KB. Je to dokonca menej, ako pri RSA, ktorý potrebuje 0.25 KB na uloženie súkromného kľúča.



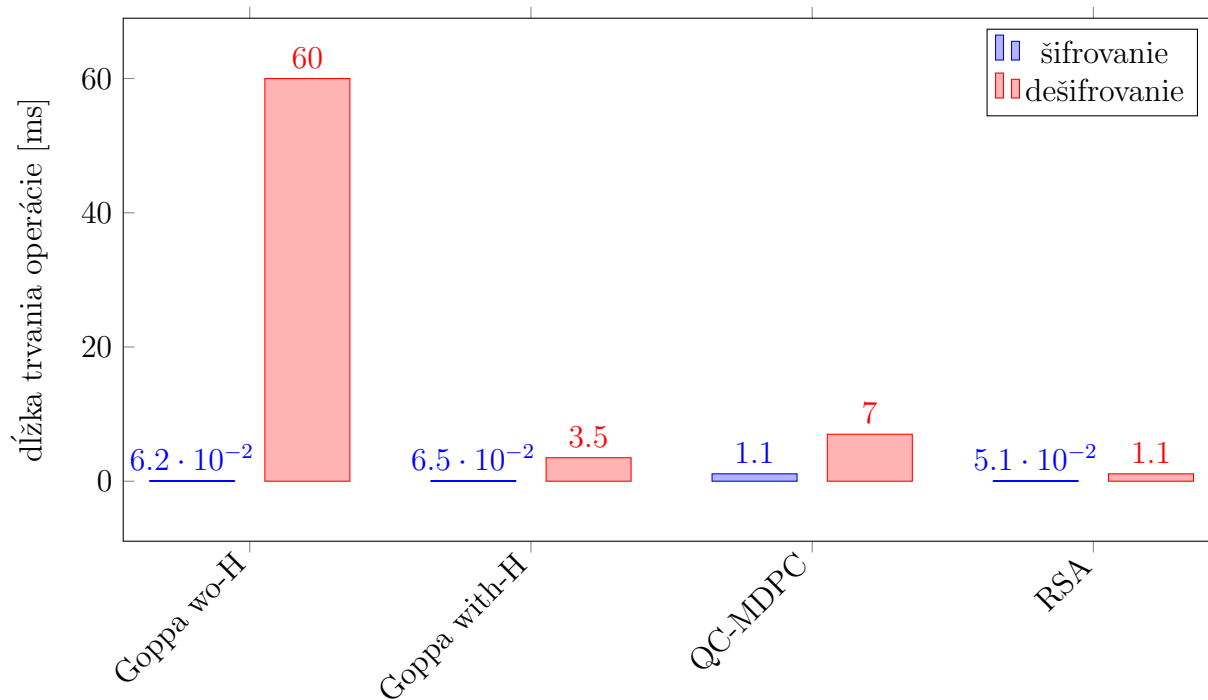
Graf 3 Spotreba pamäte pre rôzne kryptosystémy

Vďaka minimalizácii reprezentácie kľúčov sa nám úspešne podarilo zminimalizovať taktiež aj spotrebu pamäte. Pre Goppa kódy predstavuje spotreba pamäte 180 KB, pre verziu s QC-MDPC kódmi klesla spotreba iba na 7 KB. Pre verziu s Goppa kódmi, kde máme predpočítanú celú súkromnú maticu  $H$ , ide dokonca o spotrebu 378 KB. Spotreba pamäte pre McElieceov kryptosystém s QC-MDPC kódmi je nižšia ako v prípade RSA, pri ktorom sme namerali spotrebu 8 KB.

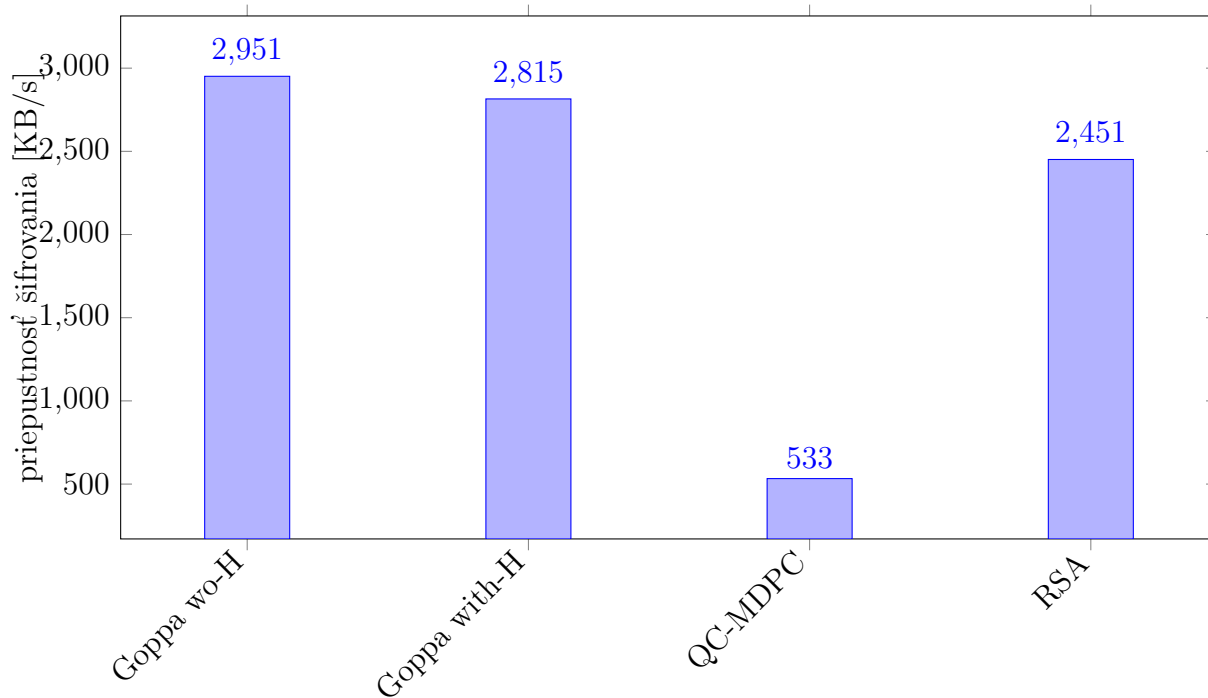


Graf 4 Dĺžka trvania generovania kľúčov pre rôzne kryptosystémy

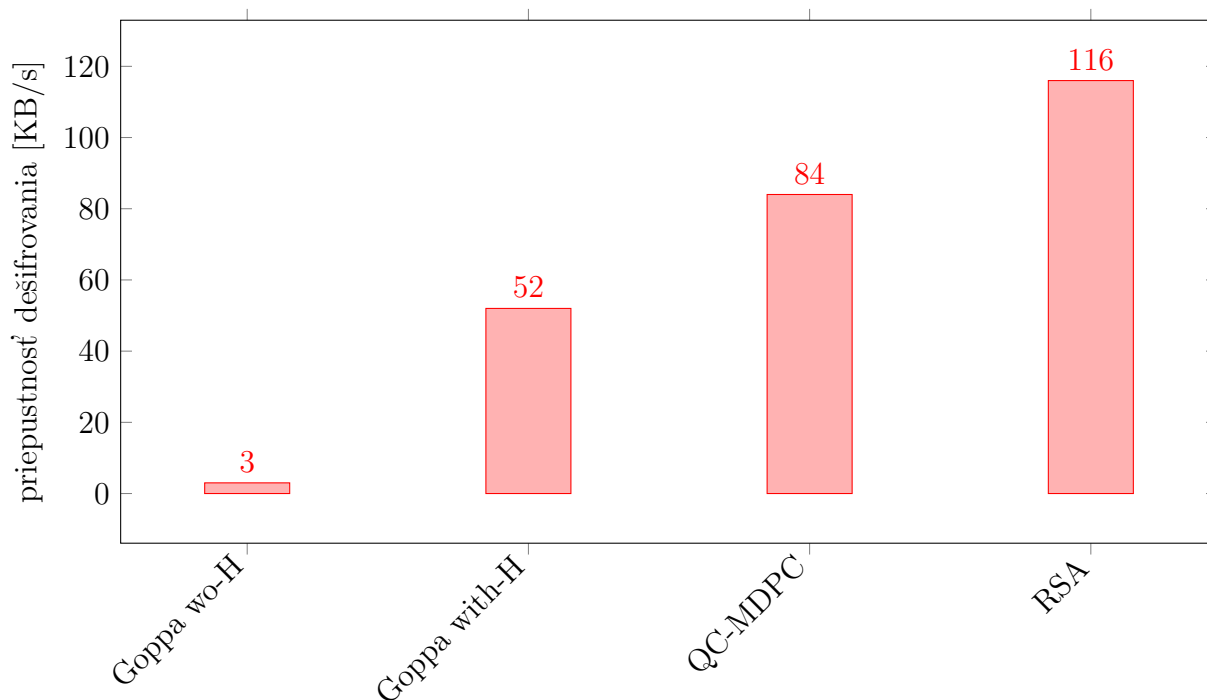
Keď sa pozrieme na dosiahnuté rýchlosti generovania kľúčov, tak sme dosiahli viac ako 36-násobné urýchlenie oproti Goppa kódom. Zo 700 ms na generovanie sme sa dostali na čas iba 19 ms. Aj v porovnaní s RSA sme dosiahli lepšie výsledky. Kľúče sa vygenerujú za 19 ms, čo je zlepšenie oproti času 33 ms pre RSA.



Graf 5 Dĺžka trvania šifrovania a dešifrovania pre rôzne kryptosystémy



Graf 6 Priepustnosť šifrovania pre rôzne kryptosystémy



Graf 7 Priepustnosť dešifrovania pre rôzne kryptosystémy

Pre porovnanie rýchlostí šifrovania a dešifrovania je dôležité zohľadniť aj veľkosť správy, ktorá sa šifruje / dešifruje. Preto je lepším ukazovateľom na porovnanie rôznych kryptosystémov práve priepustnosť, ktorá uvádza rýchlosti v jednotkách KB/s. Priepustnosť dešifrovania McEliecovho kryptosystému s Goppa kódmi pri verzii bez predpočítanej súkromnej matice  $H$  je 3 KB/s. S predpočítanou maticou  $H$  je dosiahnutá priepustnosť omnoho vyššia, 52 KB/s. Avšak predpočítanie matice  $H$  vedie k omnoho väčšiemu súkromnému kľúču a výrazne vyššej spotrebe pamäte. Naša implementácia s QC-MDPC kódmi vykazovala rýchlosť dešifrovania na úrovni 84 KB/s. Pre porovnanie, RSA kryptosystém dosahoval priepustnosť 116 KB/s pre dešifrovanie.

Keď si zanalyzujeme priepustnosť šifrovania, tak najlepšie hodnoty dosahoval McEliecov kryptosystém s Goppa kódmi, keďže ide o veľmi jednoduchú operáciu vynásobenia správy s verejnou maticou a prídanie náhodnej chyby. Dosiahnutá priepustnosť je na úrovni 2951 KB/s. Podobné výsledky dosahovalo aj RSA s rýchlosťou 2451 KB/s, keďže využíva jednoduchý verejný exponent pre umocňovanie správy, väčšinou celé číslo do maximálne 3-4 B. Nami implementovaná verzia McEliecovho kryptosystému s QC-MDPC kódmi dosiahla nižšiu priepustnosť, 533 KB/s. Toto je spôsobené tým, že pre šifrovanie dopočítavame riadky verejnej matice za behu, čo vedie k veľkému množstvu operácií.

Možnosťou zlepšenia by bolo predpočítanie celej verejnej matice, avšak vzhľadom na rozmery matice  $4801 \times 4801$  (bez jednotkovej matice  $I$ ) by sme sa odklonili od cieľa zminimalizovať kľúče a spotrebu pamäte.

### 3.5 Porovnanie našej implementácie s inou

Pre porovnanie sme si vybrali implementáciu z Nemeckej univerzity Ruhr-Universität Bochum od autorov Stefan Heyse, Ingo von Maurich a Tim Güneysu.[2] Ich práca sa primárne zameriavala na platformy mikroprocesorov a FPGA programovateľných hradlových polí. Pre porovnanie používame testovaciu verziu kompatibilnú s x86 architektúrou. Implementácia je v jazyku C. Táto testovacia verzia slúžila iba na otestovanie parametrov pre kryptosystém. V nasledujúcej tabuľke uvádzame výsledky porovnania pri nastavených rovnakých parametroch kryptosystému. Počet prevedených testov je 1000 pre každú operáciu, pričom uvádzame priemerné výsledky.

kryptosystém	naša implementácia	Ruhr-Universität Bochum
rýchlosť generovania [ms]	19	10
rýchlosť šifrovania [ms]	1.1	1.25
priepustnosť šifrovania [KB/s]	533	469
rýchlosť dešifrovania [ms]	7	5.4
priepustnosť dešifrovania [KB/s]	84	109

Tabuľka 17 Porovnanie našej implementácie s implementáciou z Ruhr-Universität Bochum

Dosiahnuté výsledky generovania kľúčov a dešifrovania sú lepšie v prospech implementácie z Ruhr-Universität Bochum. Šifrovanie je rýchlejšie v našej implementácii. Celkovo sú výsledky porovnateľné a nie je vidieť veľké odlišnosti v rýchlostiach a priepustnostiach jednotlivých operácií.

## Záver

Podarilo sa nám úspešne vytvoriť reálnu implementáciu post-kvantového McEliecovho kryptosystému s použitím kvázi-cyklických štruktúr. Tento variant vedie k minimalizácii veľkosti kľúčov, k zníženiu pamäťových nárokov a je vhodný aj pre použitie v obmedzených zariadeniach. Zároveň zachováva rovnakú mieru bezpečnosti ako pôvodný McEliecov kryptosystém s Goppa kódmi. Kryptosystém sme úspešne implementovali ako rozšírenie vznikajúcej kryptografickej knižnice BitPunch. Nie sú zatiaľ známe žiadne útoky, ktoré by znížili úroveň bezpečnosti tohto kryptosystému. Implementáciu by bolo potrebné zanalyzovať z pohľadu možnosti útokov pomocou postranných kanálov a spraviť protiopatrenia k týmto nedostatkom.

## Zoznam použitej literatúry

- [1] Misoczki, Rafael, et al. "MDPC-McEliece: New McEliece variants from moderate density parity-check codes." Information Theory Proceedings (ISIT), 2013 IEEE International Symposium on. IEEE, 2013.
- [2] Heyse, Stefan, Ingo Von Maurich, and Tim Güneysu. "Smaller keys for code-based cryptography: QC-MDPC McEliece implementations on embedded devices." Cryptographic Hardware and Embedded Systems-CHES 2013. Springer Berlin Heidelberg, 2013. 273-292.
- [3] von Maurich, Ingo, and Tim Güneysu. "Lightweight code-based cryptography: QC-MDPC McEliece encryption on reconfigurable devices." Proceedings of the conference on Design, Automation & Test in Europe. European Design and Automation Association, 2014.
- [4] Gallager, Robert G. "Low-density parity-check codes." Information Theory, IRE Transactions on 8.1 (1962): 21-28.
- [5] Bernstein, Daniel J., Johannes Buchmann, and Erik Dahmen. Post-quantum cryptography. Springer, 2009.
- [6] Shoufan, Abdulhadi, et al. "A novel processor architecture for McEliece cryptosystem and FPGA platforms." Application-specific Systems, Architectures and Processors, 2009. ASAP 2009. 20th IEEE International Conference on. IEEE, 2009.
- [7] McEliece, Robert J. "A public-key cryptosystem based on algebraic coding theory." DSN progress report 42.44 (1978): 114-116.
- [8] D. J. Bernstein, T. Lange, and C. Peters, "Attacking and defending the McEliece cryptosystem," Post-Quantum Cryptography, LNCS, vol. 5299, pp. 31–46, 2008.
- [9] Otmani, Ayoub, Jean-Pierre Tillich, and Léonard Dallot. "Cryptanalysis of two McEliece cryptosystems based on quasi-cyclic codes." Mathematics in Computer Science 3.2 (2010): 129-140.
- [10] Shor, Peter W. "Algorithms for quantum computation: discrete logarithms and factoring." Foundations of Computer Science, 1994 Proceedings., 35th Annual Symposium on. IEEE, 1994.



- [11] Olejár D., and Martin Stanek. "Úvod do teórie kódovania." 2003.
- [12] Gulliver, Thomas Aaron. Construction of quasi-cyclic codes. Diss. University of Victoria, 1989.
- [13] Faugere, J-C., et al. A distinguisher for high rate McEliece cryptosystems. Information Theory Workshop (ITW), 2011 IEEE. IEEE, 2011.
- [14] Rosenthal, Chris Monico Joachim, and Amin Shokrollahi. "Using low density parity check codes in the McEliece cryptosystem." (2000).
- [15] Baldi, Marco, and Franco Chiaraluce. "Cryptanalysis of a new instance of McEliece cryptosystem based on QC-LDPC codes." Information Theory, 2007. ISIT 2007. IEEE International Symposium on. IEEE, 2007.
- [16] Baldi, Marco, Marco Bodrato, and Franco Chiaraluce. "A new analysis of the McEliece cryptosystem based on QC-LDPC codes." Security and Cryptography for Networks. Springer Berlin Heidelberg, 2008. 246-262.
- [17] Huffman, William Cary, and Vera Pless. Fundamentals of error-correcting codes. Vol. 22. Cambridge: Cambridge university press, 2003.
- [18] Gulyás A., Klein M., Kudláč J., Machovec F., Uhrecký F. "Reálna implementácia Code-based cryptography." Tímový projekt. UIM FEI STU, Ilkovičova 3, 81219 Bratislava, Slovakia, 2014.
- [19] Uhrecký František. "Implementácia kryptografickej knižnice s McEliece kryptosystémom." Diplomová práca. UIM FEI STU, Ilkovičova 3, 81219 Bratislava, Slovakia, 2015.
- [20] Klein Marek. "Postranné kanály v SW implementácii McElieceovho kryptosystému." Diplomová práca. UIM FEI STU, Ilkovičova 3, 81219 Bratislava, Slovakia, 2015.
- [21] Donald, E. Knuth. "The art of computer programming." Sorting and searching 3 (1999): 426-458.

# Prílohy

## Príloha A - elektronické médium

Cesta	Popis
diplomova_praca.pdf	Súbor obsahuje textovú časť diplomovej práce.
mceliece_qcmdpc/	Priečinok obsahuje praktickú časť diplomovej práce.
mceliece_qcmdpc/doc/	Priečinok obsahuje dokumentáciu k implementáciám QC-MDPC McEliecovho kryptosystému. Dokumentácia je vygenerovaná pomocou nástroja doxygen.
mceliece_qcmdpc/doc/html/index.html	Súbor pre zobrazenie dokumentácie vo formáte HTML cez internetový prehliadač.
mceliece_qcmdpc/implementation/	Priečinok obsahuje samotnú implementáciu kryptosystému.
mceliece_qcmdpc/implementation/Makefile	Make file pre skompilovanie zdrojových kódov.
mceliece_qcmdpc/implementation/src/	Priečinok obsahuje zdrojové kódy.
mceliece_qcmdpc/implementation/build/	Priečinok obsahuje skompilovaný spustiteľný súbor.