УНИВЕРЗИТЕТ ”СВ. КИРИЛ И МЕТОДИЈ” - СКОПЈЕ
ФАКУЛТЕТ ЗА ИНФОРМАТИЧКИ НАУКИ
И КОМПЈУТЕРСКО ИНЖЕНЕРСТВО

М-Р ДАНИЕЛА МЕЧКАРОСКА

# ПРИМЕНА НА КВАЗИГРУПИ ВО КРИПТОКОДИРАЊЕ И БЛОК-ВЕРИГИ

ДОКТОРСКА ДИСЕРТАЦИЈА

СКОПЈЕ, 2020

| | |
|---|---|
| Ментор: | д-р Верица Бакева Смиљкова, редовен професор<br>ФИНКИ, УКИМ, Скопје |
| Коментор: | д-р Весна Димитрова, редовен професор<br>ФИНКИ, УКИМ, Скопје |
| Членови на комисијата: | д-р Смиле Марковски, редовен професор<br>ФИНКИ, УКИМ, Скопје (во пензија) |
| | д-р Верица Бакева Смиљкова, редовен професор<br>ФИНКИ, УКИМ, Скопје |
| | д-р Весна Димитрова, редовен професор<br>ФИНКИ, УКИМ, Скопје |
| | д-р Александра Поповска Митровиќ, вонреден професор<br>ФИНКИ, УКИМ, Скопје |
| | д-р Александра Милева, редовен професор<br>Факултетот за информатика, УГД, Штип |
| Датум на одбрана: | _____ |
| Датум на промоција: | _____ |
| Научна област: | Кодирање и криптографија |

# Резиме

Во оваа докторска дисертација истражувањата се движат во две насоки. Првата е примена на квазигрупите за дизајнирање на криптокодови кои поправаат грешки. Разгледани се Случајните кодови базирани на квазигрупи (Random Codes Based on Quasigroups - RCBQ), кои за прв пат се предложени од Д. Глигороски, С. Марковски и Љ. Коцарев. Овие кодови се комбинација на криптографски алгоритми и кодови за поправање на грешки и зависат од неколку параметри. Брзината на процесот на декодирање е еден од најголемите проблеми за овие кодови. Со цел да се забрза процесот на декодирање, А. Поповска-Митровиќ, С. Марковски и В. Бакева, дефинираат нов алгоритам за кодирање/декодирање наречен алгоритам-за-декодирање-со-пресеци (Cut-Decoding или АДП алгоритам), а потоа истите автори прават модификација на овој алгоритам, наречена алгоритам-за-декодирање-со-4-пресеци (4-Sets-Cut-Decoding или АД4П алгоритам). Во оваа докторска дисертација испитувани се перформансите на овие кодови при пренос на пораки низ Гаусов канал. Посебно се проучувани перформансите при пренос на слики и споредени се резултатите добиени со АДП и АД4П. Исто така, дефиниран е филтер со цел визуелно подобрување на оштетувањата настанати од грешки што се јавуваат при пренос на сликите низ Гаусов канал.

За добивање на уште побрз процес на декодирање, во оваа дисертација дефинирани се алгоритми за кодирање/декодирање, наречени Брз-алгоритам-за-декодирање-со-пресеци (Fast-Cut-Decoding или БрзАДП) и Брз-алгоритам-за-декодирање-со-4пресеци (Fast-4-Sets-Cut-Decoding или БрзАд4П). Анализирани се перформансите на на различните алгоритми за декодирање на RCBQ (АДП, БрзАДП, АД4П, БрзАД4П) за код со рата $R = 1/4$ и $R = 1/8$, за различни вредности на $SNR$ во Гаусов канал. Исто така, дефиниран е филтер за подобрување на квалитетот на аудио датотеки при пренос низ Гаусов канал.

Понатаму, проучувани се и перформансите на криптокодовите базирани на квазигрупи за пренос низ канали со рафални (burst) грешки. За симулација

на канал со рафални грешки користен е Гилберт-Елиот моделот. Дефинирани се нови алгоритми (РафаленАДП и РафаленАД4П), кои го подобруваат процесот на декодирање во каналите со рафални грешки. Проучувани се перформансите на различните алгоритми за декодирање на RCBQ (АДП, РафаленАДП, АД4П, РафаленАД4П) за пренос на обични пораки и слики за код со рата $R = 1/4$ и $R = 1/8$, за различни комбинации на веројатностите за премин, од добра во добра состојба и од лоша во лоша состојба во Гилберт-Елиот каналот.

Со цел да ги примениме брзите алгоритми на RCBQ за пренесување пораки преку канал со рафални грешки, дефинираме два нови алгоритми за кодирање/декодирање наречени Брз-РафаленАДП (БрзР-АДП) и Брз-РафаленАД4П (БрзР-АД4П). Во овие алгоритми применуваме интерливер на добиениот коден збор и деинтерливер на примените пораки на излезот на каналот, после делењето на две (или четири) делови. Анализирани се перформансите на БрзР-АДП и БрзР-АД4П за код со рата $R = 1/4$ и $R = 1/8$, за пренос преку Гилберт-Елиот канал. Експериментите се направени за различни комбинации на веројатности за премин од добра во добра состојба и од лоша во лоша состојба во каналот.

Втората насока на истражување во оваа докторска дисертација е примена на квазигрупите во забрзување на преносот на трансакциите во блок-веригите (Blockchain) и намалување на просторот за складирање во истите. На почетокот направено е истражување на технологијата на блок-веригите. Разгледани се основните поими користени во оваа технологија, начинот на работа на блок-веригата, нејзините основни карактеристики, како и нејзината примена во масивните податоци (BigData) и Интернет на нештата (Internet of things или IoT).

Еден од главните проблеми во блок-веригите е брзината на преносот. Мрежното кодирање како една техника што ја подобрува мрежната моќност и обезбедува голема безбедност е едно од решенијата на овој проблем. Со цел да се обезбеди побрз и поедноставен процес на декодирање во мрежното кодирање, дефинираме нов алгоритам за мрежно кодирање базиран на квазигрупи, кој ја зголемува пропусната моќ во комуникациската мрежа "пеперутка". Овај алгоритам е поедноставен и побрз отколку алгоритмите што се користат при линеарното мрежно кодирање.

Од друга страна, блок-веригите воспоставуваат криптографски безбедна структура за чување на трансакциите во форма на хаш-верига. Едно од главните ограничувања со кое се среќаваме во оваа технологија е големината на просторот за складирање, затоа што секој јазол мора да чува копија од глав-

ната книга од сите трансакции. Како што се зголемува бројот на трансакции, така се зголемува и просторот потребен за складирање на истите, со што се ограничува скалабилноста (scalability) на овој систем. Намалувањето на големината на трансакцијата која се чува во блоковите на блок-веригата е едно од решенијата за намалување на просторот потребен за складирање и намалување на трошоците за складирање на трансакцијата. Во оваа дисертација, предлагаме алгоритам за намалување на просторот за складирање во блок-веригата со користење на шемата на Шамир за споделување на тајна.

**КЛУЧНИ ЗБОРОВИ:** квазигрупа, квазигрупна трансформација, Гаусов канал, кодови кои поправаат грешки, случајни кодови, криптокодирање, Гилберт-Елиот канал, блок-вериги, мрежно кодирање, простор за складирање.

# Abstract

In this doctoral dissertation, the research is focused in two directions. The first one is the application of quasigroups for designing error-correcting cryptocodes. Random codes based on quasigroups (RCBQ) are considered. These codes are proposed for the first time by D. Gligoroski, S. Markovski and Lj. Kocarev and they are a combination of cryptographic algorithms and error-corecting codes and depend on several parameters. The speed of the decoding process is one of the biggest problems for these codes. In order to increase the speed of the decoding process, A. Popovska-Mitroviќ, S. Markovski and V. Bakeva define a new coding/decoding algorithm called Cut-Decoding algorithm. Then the same authors proposed a modification of this algorithm and the modified coding/decoding algorithm is called 4-Sets-Cut-Decoding algorithm. In this doctoral dissertation the performance of RCBQ for transmission through Gaussian channel are considered. Especially, we investigate their performances for image transmission and compare the results obtained with Cut-Decoding algorithm and 4-Sets-Cut-Decoding algorithm. Also, a filter for visually enhance of the damage caused by errors that occur when images are transmitted through the Gaussian channel is defined.

To get an even faster decoding process, we define encoding/decoding algorithms, called Fast-Cut-Decoding and Fast-4-Sets-Cut-Decoding algorithm. Performances of various RCBQ decoding algorithms (Cut-Decoding, Fast-Cut-Decoding, 4-Sets-Cut-Decoding and Fast-4-Sets-Cut-Decoding algorithms) for code with rate $R = 1/4$ and $R = 1/8$ and different values of $SNR$ in Gaussian channel are analyzed. Also, in order to improve the quality of audio files transmitted through the Gaussian channel we define a filter.

Furthermore, the performance of cryptcodes based on quasigroups for transmission through the channel with burst errors are studied. For simulation a channel with burst errors, Gilbert-Elliott model is used. We define new burst algorithms (Burst-Cut-Decoding and Burst-4-Sets-Cut-Decoding algorithms), which improve the decoding process in channels with burst errors. We analyze the performances of different RCBQ decoding algorithms (Cut-Decoding, Burst-Cut-Decoding, 4-

Sets-Cut-Decoding and Burst-4-Sets-Cut-Decoding algorithms) for transmission of ordinary messages and images for code with rate $R = 1/4$ and $R = 1/8$ and different combinations of transition probabilities from good to bad and from bad to bad state.

In order to adopt the fast RCBQ algorithms ffor transmission through a burst channel, we define two new encoding/decoding algorithms - called called FastB-Cut-Decoding and FastB-4-Sets-Cut-Decoding algorithms. In these algorithms we apply interleaving on the obtained codewords and deinterleaving on the received messages on the output of the channel, after dividing in two (or four) parts. The performance of FastB-Cut-Decoding and FastB-4-Sets-Cut-Decoding algorithms for code for rate $R = 1/4$ and $R = 1/8$, for transmission through Gilbert-Elliot channel are analyzed. The experiments were made for different combinations of transition probabilities from good to bad and from bad to bad state.

The second direction of research in this thesis is application of quasigroups in Blockchain to speed up transmission and reduce storage space. Initially, research was done on Blockchain technology. The basic concepts used in this technology, the way how the blockchain works, its basic features, as well as its application in BigData and the Internet of Things are discussed.

One of the main problems in blockchains is the speed of transmission. Network coding as a technique that improves network power and provides high security is one of the solutions to this problem. In order to provide a faster and simpler decoding process in network coding, we define a new algorithm for network coding based on quasigroups, that increase the conductivity in the "butterfly" communications network. This algorithm is simpler and faster than the algorithms used in linear network coding.

On the other side, Blockchain establishes a cryptographically secure structure for storing hash-chain transactions. One of the main limitations of this technology is the size of the storage space because each node must keep a copy of the main book of all transactions. As the number of transactions increases, so does the amount of storage required to store them, which limits the scalability of this system. One way to reduce the space required for storage is reducing the size of the transaction stored in the Blockchain blocks and reduce transaction storage costs. In this dissertation, we propose an algorithm for reducing the storage space in the blockchain using Shamir's secret sharing scheme.

**KEY WORDS:** quasigroup, quasigroup transformation, Gaussian channel, error correction, random codes, cryptocoding, Gilbert-Eliot channel, Blockchain, network coding, storage space.

# Содржина

# Вовед

Во голем број истражувања ([64], [66], [67], [69], [70], [71], [72]), се покажува дека квазигрупите и квазигрупните трансформации може да најдат примена во теоријата на кодирање и криптографијата. Примената на квазигрупите за криптографски цели, главно, се должи на големиот број квазигрупни операции над дадено конечно множество, па ако тие операции се користат при дефинирање на функциите за кодирање и декодирање, тогаш кодираната порака тешко може да се открие од трето лице. Исто така, овие операции може да се користат и за дефинирање на многу други криптографски примитиви. Од друга страна, истражувањата на својствата на низите добиени со квазигрупните трансформации покажуваат дека овие трансформации може да се применат во теоријата на кодирање за дизајн на кодови кои откриваат и поправаат грешки. Всушност, квазигрупните трансформации се пресликувања од конечни низи над конечна азбука и покажуваат својства на дискретни динамички системи. Поради тоа, како релативно нова област претставуваат предизвик за нивно понатамошно истражување и нивна примена во овие области.

Истражувањата во оваа докторска дисертација се движат во две насоки. Првата е примена на квазигрупите за дизајнирање на криптокодови кои поправаат грешки, а втората – нивна примена во блок-веригите за забрзување на преносот и за намалување на просторот за складирање.

*Примена на квазигрупите во криптокодирање*

Концептот на криптокодирањето произлегува од потребата за добивање безбеден и точен пренос. Затоа е потребно постојано подобрување на постоечките и развој на нови алгоритми кои ќе обезбедат точен и безбеден пренос на податоците. Ова доведе до интензивен развој на теоријата на кодирање и криптографијата како научни области кои се занимаваат со овие проблеми. За да се обезбеди истовремено ефикасен и безбеден пренос на податоците,

сè повеќе се развива и концептот на криптокодирање во кој процесите на кодирање и шифрирање се спојуваат во еден процес. Постојат многу дизајни ([18], [73], [96], [102], [103], [17], [16]) во кои што се спојуваат овие две научни области: шифрувачи во кои се користат кодови со цел да се зголеми нивната безбедност и обратно кодови во чиј дизајн се имплементирани алгоритми за шифрирање. Всушност, криптокодовите обезбедуваат корекција на одреден број грешки во влезната порака и тајност на податоците, со користење на само еден алгоритам. Главните истражувања во оваа област се во насока на дефинирање на нови алгоритми за кодови кои откриваат и поправаат грешки, случајни кодови, проточни шифрувачи, блок шифрувачи, псевдо генератори на случајни низи, хаш функции, итн.

Во [18] авторите предлагаат криптографски алгоритам, кој работи со 64-битни зборови и $C(32, 16)$-код за корекција на грешки, со минимално Хамингово растојание 8, каде што секој блок има 2 кодни зборови со капацитет за корекција на 3 грешки. Криптоаналитичарите мора да го знаат алгоритамот и кодот за корекција на грешки употребен за овој систем. Потребно е да ја најдат и големината на испратениот блок. Со овој алгоритам, се обезбедува посакувана доверливост и безбедност, за сметка на постепено зголемување на сложеноста на истиот.

Во [73] комбинирани се код за корекцијата на грешки и блок шифрувач, наречен шифрувач со Голема Дифузија (High Diffusion cypher, HD). Способноста за корекција на грешка на овој шифрувач се должи на новиот код за корекција на грешки наречен код со Голема Дифузија (High Diffusion code). Покажано е дека овие кодови постигнуваат оптимална дифузија и отпорност на грешки. Добиени се теоретски граници за перформансите на HD шифрувачот во однос на безбедноста и корекцијата на грешки. Предложениот HD шифрувач има безбедност еквивалентна на Rijndael шифрувачот во однос на линеарната и диференцијална криптоанализа. Во Rijndael, шифрирањето се прави со клуч 128, 192 или 256 бита, што обезбедува загарантирана зголемена безбедност при напади со груба сила (brute force). Овој шифрувач може да се користи за безбедна размена на клучеви, како и за пренос на податоци со должина од 128 или 256 бита. Безбедноста на овој метод на шифрирање се зголемува кога Rijndael се изведува неколку пати со различни клучеви во рундите. Главната примена на нивниот дизајн е за криптографски цели, иако тој може да биде користен и како код за корекција на грешки.

Тајната шема за кодирање на грешки (SECC) предложена во [102] е таква што обезбедува тајност и веродостојност на податоците во еден процес за справување со проблеми во небезбеден и несигурен канал. Во оваа шема,

само овластениот корисник, кој има тајно чувани информации, може систематски да ги отстрани грешките во каналот. Во овој труд се предложени две шеми на SECC. Првата е блок шифрирање со употреба на Preparata нелинеарни кодови; додека втората шема е базирана на техниката на верижно поврзување на блокови (block chaining) каде што секој шифриран текст е функција од сите претходни оригинални текстови и при декодирањето, грешката на еден шифриран текст ќе се прошири до последниот блок. Овие "проширени грешки" се користат за откривање на која било нелегална промена на шифрираниот текст што обезбедува *интегритет* на податоците.

Трудот [103] го анализира методот "Паралелно Поврзување на Каналот за Кодирање и Криптографија", како еден ефикасен алгоритам за корекција на пораките добиени при декодирање кои се заштитени со безбедносни механизми. Ефикасноста на ова паралелно поврзување зависи од должината на употребените блокови. Направени се симулации со користење на блокови со различна должина и притоа добиено е дека подобрувањето на кодирањето зависи од должината на блоковите.

Првичната идеја за примена на квазигрупите во случајните кодови (Random Codes Based on Quasigroup - RCBQ) е дадена во [17] и [16]. Овие кодови се комбинација на криптографски алгоритми и кодови за корекција на грешки и зависат од неколку параметри. Ако овие кодови се користени за кодирање на пораки, тогаш до оригиналните податоци може да се дојде само доколку се знае точно кои параметри се користени во процесот на кодирање, дури и ако комуникацискиот канал е без пречки. Затоа, ако информациите кои се пренесуваат, се од таен карактер, предложените случајни кодови се во голема предност пред останатите. Случајните кодови базирани на квазигрупи се темелат на концептот на тотално асихроните проточни шифрувачи. Влијанието на параметрите врз перформансите на овие кодови при пренос низ бинарен симетричен канал е проучено во [88]. Од резултатите добиени од тоа проучување, е заклучено дека избраната квазигрупа, должината на почетниот клуч и изборот на патернот за редундантната информација имаат големо влијание на перформансите на овие кодови. Експериментите во трудот [88] се направени со RCBQ при пренос низ бинарен симетричен канал.

Од направените експерименти со RCBQ може да се заклучи дека брзината на декодирање е еден од најголемите проблеми кај овие кодови. За да се подобри брзината на декодирање, во трудовите [89] и [92], авторите дефинираат нови алгоритми за декодирање: алгоритам-за-декодирање-со-пресеци (Cut-Decoding Algorithm) и алгоритам-за-декодирање-со-4пресеци

(4-Sets-Cut-Decoding Algorithm). Во сите овие трудови, проучувани се перформансите на Случајните кодови базирани на квазигрупи при пренос низ бинарен симетричен канал. Во мојата магистерска теза, чии главни резултати се објавени во [76], започнав со проучување на перформансите на овие кодови при пренос низ Гаусов канал. Поточно, проучено е како изборот на патернот за додавање на редундансата, должината на клучот и изборот на квазигрупата влијаат на перформансите на овие кодови. Во направените експерименти користен е $(72, 288)$ код со рата $R = 1/4$. Добиените заклучоци се слични како при пренос низ бинарен симетричен канал.

Една од целите на истражувањето во оваа докторска дисертација е забрзување на алгоритмите за декодорање при пренос низ Гаусов канал. Имено, декодирањето со овие алгоритми е декодирање со листа, па оттука брзината на декодирање зависи од големината на листата (помала листа значи побрзо декодирање). Во овие алгоритми, големината на листата зависи од $B_{max}$ (претпоставен број на бит грешки при пренос на еден блок). За помали вредности на $B_{max}$ се добива помала листа, но проблемот е тоа што никогаш не може однапред да се знае точниот број на настанати грешки при пренос на еден блок. Ако тој број е поголем од предвидениот $B_{max}$, грешката нема да биде поправена, но ако предвиденото $B_{max}$ е преголемо, тогаш имаме преголеми листи и премногу бавно декодирање. За да се избегне ова, барем за поголеми вредности на $SNR$ (помал шум), се предлагаат нови алгоритми кои обезбедуваат декодирање со помали листи. Проучувано е и какви се перформансите на овие алгоритми при пренос на слики и аудио датотеки. За подобрување на квалитетот на сликите и аудио датотеките декодирани со криптокодовите базирани на квазигрупи се дефинираат филтри со кои се овозможува почисто визуелно гледање и слушање на оригиналните датотеки (шумот се прочистува) дури и за помали вредности на $SNR$. Понатаму, проучувани се и перформансите на криптокодовите базирани на квазигрупи за пренос низ канали со рафални (burst) грешки. За вакви канали се користи моделот на Гилберт-Елиот за рафални грешки (Gilbert-Elliott burst model). Направени се експерименти за два вида на вакви канали. Првиот вид на Гилберт-Елиот канал е канал кој е бинарно симетричен во секоја состојба, а кај вториот вид на Гилберт-Елиот канал, каналите се Гаусови.

Да нагласиме дека сите случајни кодови и оние што се претходно предложени и овие што се предлагаат во оваа дисертација се криптокодови. Всушност, криптокодовите се дефинирани со користење на криптографски алгоритам во самиот процес на кодирање/декодирање. Овие кодови со само еден алгоритам овозможуваат не само поправање на одреден број грешки кои

настануваат при пренос низ канал со пречки, туку истовремено обезбедуваат и тајност на податоците. Примателот на информацијата до оригиналните податоци може да дојде само доколку знае точно кои параметри се користени во процесот на кодирање, дури и ако комуникацискиот канал е без пречки. Во оваа дисертација ги разгледуваме RCBQ како кодови кои поправаат грешки и затоа тука нема да бидат анализирани нивните криптографски својства. Тие криптографски својства следуваат од резултатите во [64], [67], [71] и др.

*Примена на квазигрупите во блок-вериги*

Главната идеја за воведување на биткоинот (Bitcoin) и блок-веригите (Blockchain) е креирање на децентрилизиран систем (без централен авторитет на одлучување) за размена на вредности директно помеѓу физичките лица, на пример, трансфер на пари без централен авторитет (банка, финансиска институција, ...). Таа децентрализирана мрежа за пренос на вредности (криптовалути) е заснована на мрежата на рамноправни корисници (peer-to-peer или кратко $P2P$ мрежа) и се нарекува технологија на блок-вериги (Blockchain technology) ([50], [84], [19], [7]). Главната карактеристика на технологијата на блок-веригите е децентрализираноста, односно непостоење на централна банка или било која друга централна институција која би правела пренос на трансакциите, како и 100% користење на информатичката технологија во процесот на емитување и реализирање на истите. Други нејзини основни карактеристики се и релативно безбедно плаќање на трансакциите, ниски трансакциски трошоци, анонимност на корисниците, практично невозможно фалсификување, нереверзибилност на трансакциите. Информациите во блок-веригата се споделени, сите учесници во мрежата имаат пристап до нив, секој нов внес се регистрира и запишува во оваа база на податоци, која се чува на сите компјутери во мрежата. Оваа база на податоци, во секој момент е достапна до сите корисници на мрежата со што се овозможува целосна транспарентност. Кога некој бара трансакција или кога две страни разменуваат податоци, како што се пари, договор или кое било средство кое може да е дигитално опишано, бараната трансакција се шири во $P2P$ мрежата која се состои од компјутери наречени јазли. Оваа мрежа од компјутери ја валидира трансакцијата и статусот на корисникот користејќи познати алгоритми. Во зависност од мрежните параметри, трансакцијата или ќе се верификува веднаш или се запишува во безбеден запис и се става во списокот на трансакции што чекаат за верификација. Низата од поврзани блокови од трансакции креира безбедна, независна верига. Блоковите мора прво да се

валидираат, па потоа да бидат додадени во блок-веригата. Кога блокот се верификува, се дистрибуира низ мрежата и секој јазол го додава блокот во мнозинската блок-верига. Тогаш трансакцијата е комплетна.

Во 2008 година, поединец или група луѓе, потпишани под името Сатоши Накомото, го издале трудот [83]. Биткоинот, првата главна примена на блок-веригите претставува $P2P$ верзија на електронски кеш, што овозможува директно електронско плаќање од една страна на друга без потреба од вклучување на трета страна. Оваа криптовалута е прва децентрализирана валута која користи криптографија за безбедни трансакции. Основните карактеристики на биткоинот се:

– *Децентрализираност* што значи не постоење на централна банка. Емитерите на оваа валута се корисниците или носителите на компјутерите кои "рударат" и кои ја верификуваат секоја трансакција.

– Тајноста обезбедена со криптографијата со јавен клуч ја дава *довербата* на оваа валута.

– *Автентикација* на трансакциите од еден до друг јазол во блок-веригата направена е со дигиталниот потпис.

– Дигиталниот потпис на пораката исто така обезбедува *интегритет* низ преносот.

*Рударењето* е процес со кој се создаваат биткоините. Биткоин *рударите* ги верификуваат трансакциите, ги стават во блокови од трансакции и одлучуваат кој блок е следниот, т.е. *биткоин системот* ги групира трансакциите во блокови и ги поврзува во блок-веригата. Бидејќи, многу луѓе можат да создадат блокови во исто време, кој блок ќе влезе прв во блок-веригата се решава со употреба на хаш функцијата. Излезот од хаш функцијата на блокот мора да биде под специфична вредност наречена *цел*. Првиот рудар компјутер што ќе ја реши загатката (да најде хаш вредност под дадената цел), ќе направи емитување на неговиот блок до сите учесници во мрежата, кои треба да го потврдат решението. По добивањето на неколку такви потврди, блокот се смета за валиден и со тоа неговата група на трансакции е прифатена како следна во блок-веригата. Наградата што ја добива рударот, се преполовува на секој ископани 210000 блокови (приближно на секој 4 години). Кога првпат беше ископан биткоин (2009 год.) наградата изнесуваше 50 биткоини за ископување на еден блок. Во моментов (2020 год.) наградата изнесува 6.25

биткоини за блок. На крајот, наградата ќе се намали на нула, а границата од 21 милиони биткоини (според биткоин протоколот) ќе биде достигната во 2040 година. Потоа награди ќе бидат само провизии за процесирање на трансакции.

Другата голема примена на блок-веригите е наречена *паметен договор*. Разменувањето на пари, имот, акции или која било вредност на транспарентен, без конфликтен начин, меѓу двајца сопственици врз основа на множество на услови вклучени во договор, го дефинираат паметениот договор. Овој договор е контролиран со децентрализираната согласност на блок-веригата. Како и кај традиционалниот договор, со паметниот договор се дефинирани и правилата и казните. Покрај тоа, паметниот договор автоматски ги спроведува тие обврски [26]. Секој паметен договор се состои од програмски код, датотека за складирање и биланс на сметки. Датотеката за складирање на договорот се чува во блок-веригата, додека програмскиот код на договорот го извршува мрежата на рудари во блок-веригата. Кодот на договорот се извршува секогаш кога еден корисник или некој друг договор испраќа порака. Договорот може исто така да прима и испраќа пари на други договори или корисници во билансот на сметката [10].

Технологијата на блок-веригите наоѓа примена во масивните податоци и IoT за потребите на индустријата, како и финансиските услуги, владата и здравствената заштита. Во овој случај, имплементацијата на технологијата на блок-веригите обезбедува тајност на податоците и го гарантира нивниот интегритет кога се споделуваат ([27], [43]), како и потврда дека податоците се точни и безбедни, а размената на податоци е поедноставна. Интегрирањето на блок-веригите со масивните податоци има многу предности бидејќи овозможува подобро управување со огромните количини и разновидност на информациите. Примената на блок-веригите во IoT им овозможува на IoT уредите да учествуваат во трансакциите на блок-веригите и да пронаоѓаат нови стилови на дигитални интеракции. Оваа технологија ќе обезбеди едноставна инфраструктура за уредите директно и безбедно да пренесуваат податоци или пари користејќи паметни договори.

Еден од главните проблеми во блок-веригите е брзината на преносот. Едно од решенијата на овој проблем е мрежното кодирање како една техника што ја подобрува мрежната моќност и гарантира голема безбедност. Во трудот [20], акцент се става на првите најприродни прашања што би се поставиле за оваа техника: како функционира мрежното кодирање и кои се нејзините придобивки, како се дизајнирани мрежните кодови и колку чини да се распоредат мрежите за спроведување на вакви кодови како и, дали постојат методи за

справување со циклусите и одложувањето што е присутно во сите реални мрежи.

Во [5] авторите воведуваат шема за практично мрежно кодирање. Оваа мрежа користи: баферинг за синхронизирање на произволните пакети кои што пристигнуваат и заминуваат од секој јазол; случајно кодирање за справување со различниот број на пакети во баферот и пакет формат што вклучува глобални вектори за кодирање, со цел да им обезбеди на примателите само точни информации за декодирањето на пакетите. Пренесените податоци се кодираат со цел да се зголеми пропусната моќ, да се намали доцнењето и да се направи мрежата постабилна.

Начинот на функционирање на технологијата на мрежното кодирање е следниот: во еден јазол на мрежата, се користат алгебарските алгоритми кои ги кодираат примените пораки генерирајќи на тој начин излезни пораки. Потоа, добиените пораки се декодираат во нивните дестинации. На овој начин, потребни се помалку пораки за пренесување на сите податоци, но за тоа е потребна поголема обработка во посредничките и крајните јазли. Во [63], авторите истражуваат колку брзо секој јазол може да ги добие целосните информации, или еквивалентно, која е стапката на информации што пристигнува до секој јазол. Дозволувајќи еден јазол да ги кодира добиените податоци пред да ги пренесе, истражувањето вклучува оптимизација на механизмите за испраќање податоци преку компјутерска мрежа на повеќе корисници во исто време (multicast mechanisms) на јазлите. Најпопуларното мрежно кодирање е линеарното мрежно кодирање, каде што секој јазол генерира нови пакети кои се линеарни комбинации на претходно примени пакети, множејќи ги со коефициенти избрани од конечно поле. Во крајните јазли, оригиналните пораки можат да се добијат со решавање на систем линеарни равенки со Гаусов метод на елиминација.

Мрежата “пеперутка” е форма на мрежна топологија која може да се користи за поврзување на различни јазли во мултипроцесорски систем [20]. Оваа мрежа често се користи за да се илустрира како мрежното кодирање може да го заобиколи рутирањето (насочувањето).

Сложеноста на декодирањето поради користење на Гаусовата метода на елиминација, ни ја дава идејата за конструирање нов алгоритам за мрежно кодирање базиран на квазигрупи, за зголемување на спроводливоста во ”пеперутка” комуникациската мрежа. Овај алгоритам е поедноставен и побрз отколку алгоритмите што се користат при линеарното мрежно кодирање.

Од друга страна, блок-веригите воспоставуваат криптографски безбедна структура за чување на трансакциите во форма на хаш-верига. Едно од глав-

ните ограничувања со кое се среќаваме во оваа технологија е големината на просторот за складирање затоа што секој јазол мора да чува копија од главната книга од сите трансакции. Како што се зголемува бројот на трансакции, така се зголемува и просторот потребен за складирање на истите, со што се ограничува скалабилноста на овој систем. Намалувањето на големината на трансакцијата која се чува во блоковите на блок-веригата е едно од решенијата за намалување на просторот потребен за складирање и намалување на трошоците за складирање на трансакцијата [105]. Во [8] авторите предлагаат решавање на овој проблем со помош на мрежно кодирање. Тие го делат блокот со големина $G$ во $k$ пакети со еднаква должина. Овие $k$ пакети се кодираат во $n$ кодирани пакети користејќи линеарни комбинации на оригинални пакети.

Концептот на *Дистрибуираниот простор за складирање во блок-веригите* (DSB) е воведен во [94]. Во ваквиот дистрибуиран простор за складирање, трансакцискиот блок се шифрира со употреба на различни приватни клучеви и потоа се дистрибуира на подмножествата од јазли. Исто така, хаш вредностите и клучеви за шифрирање се чуваат меѓу јазлите (peers) од подмножеството во $P2P$ мрежата со помош на шемата на Шамир за споделување на тајна. Дистрибуираниот простор за складирање на блок-веригите значително го намалува просторот за складирање во системот на блок-веригите. За да се подобри дистрибуираниот простор за складирање на блок-веригите, се предлага локална шема за споделување на тајна во [56].

Во [95] авторите ги подобруваат перформансите на блок-веригите и ги намалуваат трошоците за складирање со дизајнирање на шема за кодирање така што секој јазол содржи само дел од секоја трансакција.

Во оваа докторска дисертација, предложено е на кој начин може да се намали просторот за складирање во блок-веригата со користење на шемата на Шамир за споделување на тајна.

Резултатите од истражувањата дадени во оваа докторска дисертација се објавени во меѓународни списанија (од кои еден е со фактор на влијание (IF)) и зборници на меѓународни конференции и се презентирани на неколку домашни и меѓународни конференции и семинари. Дел од трудовите се веќе објавени или пратени за објавување во меѓународни и домашни списанија и зборници од конференции. Во продолжение е даден списокот на овие трудови:

1. D. Mechkaroska, A. Popovska-Mitrovikj, V. Bakeva: Cryptcodes Based on Quasigroups in Gaussian channel, Quasigroups and Related Systems, vol.

24 (2), 2016, pp. 249-268.

2. D. Mechkaroska, A. Popovska-Mitrovikj, V. Bakeva: A Filter for Images Decoded using Cryptcodes Based on Quasigroups, Proceedings of 14th International Conference on Informatics and Information Technologies, April 2017, Mavrovo, Macedoinia, pp. 52–56.

3. A. Popovska-Mitrovikj, V. Bakeva, D. Mechkaroska: New Decoding Algorithm for Cryptcodes Based on Quasigroups for Transmission Through a Low Noise Channel, In: Trajanov, D., Bakeva, V. (eds.): Communications in Computer and Information Science Series (CCIS), Vol.778, ICT-Innovations 2017, Springer, pp. 196–204.

4. D. Mechkaroska, V. Dimitrova, A. Popovska-Mitrovikj: A survey on applications of Blockchain technology, Proceedings of 15th International Conference on Informatics and Information Technologies, April 2018, Mavrovo, Macedoinia, pp. 66–69.

5. D. Mechkaroska, V. Dimitrova, A. Popovska-Mitrovikj: Analysis of the possibilities for improvement of Blockchain technology, 2018 26th Telecommunications Forum (TELFOR), Belgrade, Serbia, Nov.2018,
doi: 10.1109/TELFOR.2018.8612034
*https://ieeexplore.ieee.org/document/8612034*

6. D. Mechkaroska, A. Popovska-Mitrovikj, V. Dimitrova: Secure Big Data and IoT with implementation of Blockchain, International Scientific Journal Security & Future, vol. 2(4), 2018, pp. 183 – 185

7. D. Mechkaroska, A. Popovska-Mitrovikj, V. Bakeva Smiljkova: Performances of Fast Algorithms for Random Codes Based on Quasigroups for Transmission of Audio Files in Gaussian Channel, In: Kalajdziski, S., Ackovska, N. (eds): ICT Innovations 2018. Engineering and Life Sciences, Springer International Publishing, pp. 286–296.

8. D. Mechkaroska, A. Popovska-Mitrovikj, V. Bakeva: New Cryptcodes for Burst Channels, In: Ćirić M., Droste M., Pin JÉ. (eds) Algebraic Informatics. CAI 2019. Lecture Notes in Computer Science, vol 11545. Springer, Cham, pp. 202–212.

9. V. Bakeva, A. Popovska-Mitrovikj, D. Mechkaroska, V. Dimitrova, B. Jakimovski, V. Ilievski: Gaussian channel transmission of images and audio

files using cryptcoding, IET Communications, Vol. 13, Issue 11, (2019), p. 1625 – 1632. (**IF. 1.779**)

10. D. Mechkaroska, A. Popovska-Mitrovikj, V. Bakeva: Cryptcoding of Images for Transmission Trough a Burst Channels, Journal of Engineering Science and Technology Review (JESTR), pp 65—69.

11. D. Mechkaroska, A. Popovska-Mitrovikj, V. Bakeva, V.Dimitrova: Network Coding based on quasigroup, Proceedings of the 10th International Conference on Information Society and Technology, March 2010, Kopaonik, Serbia, pp. 240-243

12. A. Popovska-Mitrovikj, D. Mechkaroska, V. Dimitrova, V. Bakeva: Algorithm for Reducing Storage in Blockchain based on Secret Sharing, 2nd International Conference on Electrical, Communication and Computer Engineering (ICECCE 2020), Istanbul, Turkey, 12-12 June 2020, pp. 567–569

13. Popovska-Mitrovikj A., Bakeva V., Mechkaroska D.: *Fast Decoding with Cryptcodes for Burst Errors*, In: Dimitrova V., Dimitrovski I. (eds.): Communications in Computer and Information Science Series (CCIS), ISTInnovations 2020, Springer (accepted).

## Преглед на содржината

Докторската дисертација е поделена на осум глави и заклучок. Во првата глава *Квазигрупи и квазигрупни трансформации*, дадени се основните дефиниции, како и својствата на квазигрупите и квазигрупните трансформации, врз чија основа се темелат истражувањата направени во оваа докторска дисертација, за примена на квазигрупите во теоријата на кодирање и криптографијата.

Втората глава се однесува на *Случајните кодови базирани на квазигрупи*. Првичната идеја за примена на квазигрупите во случајните кодови е дадена од Данило Глигороски, Смиле Марковски и Љупчо Коцарев. Во оваа глава, најпрво е дефиниран тотално асихрон проточен шифрувач - EdonZ, врз чија основа се темелат Случајните кодови базирани на квазигрупи. Потоа опишан е Стандардниот алгоритам за кодирање/декодирање. Брзината на декодирање е најголем проблем кај овие кодови. Со цел да се забрза процесот на декодирање, Александра Поповска-Митровиќ, Смиле Марковски и Верица

Бакева, дефинираат нов алгоритам за кодирање/декодирање наречен алгоритам-за-декодирање-со-пресеци (Cut-Decoding или АДП алгоритам), а потоа истите автори прават модификација на овој алгоритам, наречена алгоритам-за-декодирање-со-4-пресеци (АД4П алгоритам). Во оваа глава опишани се АДП и АД4П алгоритмите, а со цел да се подобри и веројатноста за пакет-грешка и бит-грешка опишани се методи за намалување на бројот на грешките во овие алгоритми.

Во третата глава дадени се експерименталните резултати за случајните кодовите базирани на квазигрупи при пренос низ Гаусов канал. Експерименталните резултати добиени со Стандардниот и со АДП алгоритамот, се презентирани во мојата магистерска теза. Во истражувањето за оваа докторска дисертација, проучени се перформансите на АД4П кодовите, и истите се споредени со соодветните резултати добиени со АДП. Во сите експерименти се разгледуваат разликите меѓу пренесената и декодирана порака. Исто така, ги споредуваме експериментално добиените вредности за веројатноста за бит грешка ($BER$) и веројатноста за пакет грешка ($PER$) со двата алгоритми. Во овој дел, презентирани се и експерименталните резултати добиени при пренос на слики низ Гаусов канал. Од споредбата на овие резултати, покажано е дека со АД4П алгоритамот се добиваат подобри резултати за сите вредности на $SNR$, додека процесот на декодирање се забрзува во зависност од вредноста на $SNR$ и е од $2$ до $16$ пати побрз во однос на декодирањето со АДП алгоритамот. Во овој дел предложен е и филтер, со кој визуелно се подобруваат оштетувањата настанати од грешките при пренос на сликата низ каналот. Нови резултати во овој дел се: проучување на перформансите на АД4П кодовите при пренос низ Гаусов канал, дефиниран е филтер со цел визуелно подобрување на оштетувањата настанати од грешки што се јавуваат при пренос на сликите низ Гаусов канал; испитувани се перформансите на Случајните кодови базирани на квазигрупи при нивна примена во кодирање/декодирање на слики кои се пренесуваат низ каналот со грешки.

За да се подобрат перформансите на случајните кодови, т.е. да се забрза процесот на декодирање, во четвртата глава предложени се нови модификации на АДП и АД4П, наречени Брз-алгоритам-за-декодирање-со-пресеци (Fast-Cut-Decoding) или кратко БрзАДП и Брз-алгоритам-за-декодирање-со-4-пресеци (Fast-4-Sets-Cut-Decoding) или кратко, БрзАД4П. Декодирањето со АДП и АД4П е всушност декодирање со листа. Со новите БрзАДП и БрзАД4П се обидуваме да ја декодираме пораката користејќи пократки листи и во случај на успешно декодирање за помали вредности на $B_{max}$, избегнуваме долги листи и побавно декодирање. Во декодирањето во

старите алгоритми користиме $B_{max} = 4$ или $B_{max} = 5$, додека декодирањето со новите брзи алгоритми започнува со $B_{max} = 1$. Во оваа глава направени се експерименти за кодови со рата $1/4$ и $1/8$ со новите алгоритми, а потоа ги споредуваме добиените резултати со соодветните резултати добиени со АДП и АД4П. Од добиените резултати, можеме да извлечеме заклучок дека резултатите за $BER$ и $PER$ добиени со новите алгоритми се подобри од соодветните резултати добиени со старите верзии на овие алгоритми. Исто така, за поголеми вредности на $SNR$ (слаб шум во каналот), декодирањето со новопредложените алгоритми е многу побрзо отколку со старите алгоритми. Во овој дел презентирани се експериментални резултати за истражување на перформансите на брзите алгоритми за пренос на слики и звук низ каналот. Повторно се споредуваат добиените резултати со соодветните резултати добиени со старите алгоритми. Се добиваат слични резултати како и при пренос на обичните пораки. Направена е и анализа на процентот на пораки за кои декодирањето завршува за различни вредности на $B_{max}$. Од анализите се заклучува дека за поголеми вредности на $SNR$, имаме голем процент на пораки чие декодирање завршува со помали вредности на $B_{max}$, т.е. имаме побрзо декодирање. Исто така, во овој дел дефинираме и филтер за подобрување на квалитетот на аудио датотеките декодирани со Случајните кодови базирани на квазигрупи, на тој начин што се прави замена на погрешно декодираните нибли со нова вредност добиена од неколку претходни нибли. Нови резултати во овој дел се: дефинирање на нови модификации на АДП и АД4П, наречени БрзАДП и БрзАД4П и испитување на перформансите на Случајните кодови базирани на квазигрупи при кодирање/декодирање на обични пораки, слики и звук пренесени низ Гаусов канал, со помош на новите брзи алгоритми; дефинирање филтер за подобрување на звукот при пренос низ Гаусов канал.

Во експериментите со каналите со рафални грешки, не добиваме добри резултати со претходно објаснетите алгоритми. Поради тоа, во петтата глава, предлагаме нови алгоритми за кодирање/декодирање наречени РафаленАДП (Burst-Cut-Decoding) и РафаленАД4П (Burst-4-Sets-Cut-Decoding). Во новите алгоритми воведуваме интерливер во алгоритамот за кодирање и соодветен деинтерливер во алгоритамот за декодирање. Всушност, интерливерот и деинтерливерот се користат за справување со рафални грешки во комуникациските системи. За симулација на канал со рафални грешки го користиме Гилберт-Елиот моделот. Во нашите експерименти користиме два вида на Гилберт-Елиот канали: канал кој е бинарно симетричен во секоја состојба (добра и лоша состојба на каналот) и канал кој е Гаусов во секоја состојба

на каналот. Потоа ги споредуваме веројатностите за пакет грешка и веројатноста за бит грешка добиени со старите АДП и АД4П, со соодветните веројатности добиени со новите РафаленАДП и РафаленАД4П. Во експериментите користиме 4 комбинации за веројатностите за премин од добра во добра состојба и од лоша во лоша состојба. Анализирајќи ги добиените резултати заклучуваме дека веројатноста за бит грешка со новите рафални алгоритми е неколку (од 1 до 10) пати подобра од соодветните веројатности за бит грешка добиена со соодветните стари алгоритми за кодирање/декодирање. Истиот заклучок може да се изведе и за веројатноста за пакет грешка. Потоа, направивме експерименти за пренос на слики низ канал со рафални грешки за двата вида на Гилберт-Елиот канали. На добиените слики, потоа го додаваме филтерот предложен во третата глава. Направивме споредба на добиените слики пред и после примената на предложениот филтер. Со цел да ги примениме брзите алгоритми на RCBQ за пренесување пораки преку канал со рафални грешки, дефинираме два нови алгоритми за кодирање/декодирање - наречени БрзР-АДП и БрзР-АД4П. Во овие алгоритми применуваме интерливер на добиениот коден збор и деинтерливер на примените пораки на излезот на каналот, после делењето на две (или четири) делови. Анализирани се перформансите на БрзР-АДП и БрзР-АД4П за код со рата $R = 1/4$ и $R = 1/8$, за пренос преку Гилберт-Елиот канал. Експериментите се направени за различни комбинации на веројатности за премин од добра во добра состојба и од лоша во лоша состојба во каналот. Нови резултати во овој дел се: нови криптокодови за каналите со рафални грешки наречени РафаленАДП и РафаленАД4П со воведување интерливер/деинтерливер во новите алгоритми за кодирање/декодирање при пренос на пораки низ Гилберт-Елиот канал; истражување на перформансите на новите рафални алгоритми за пренос на обични пораки и слики низ два вида на Гилберт-Елиот канали: бинарно симетричен и Гаусов канал, два нови алгоритми за кодирање/декодирање наречени БрзР-АДП и БрзР-АД4П, во кои применуваме интерливер на добиениот коден збор и деинтерливер на примените пораки на излезот на каналот.

Шестата глава од докторската дисертација е воведен дел во втората насока на движење во оваа дисертација, а тоа е примена на квазигрупите во блок-веригите за забрзување на преносот и намалување на просторот. Во оваа глава се објаснува поимот блок-верига, како растечка низа на поврзани блокови од трансакции. Всушност, објаснети се основните поими и принципи на работа во технологијата на блок-веригите. Направено е истражување и анализа на најважните примена на блок-веригите: биткоинот, првата главна примена на блок-веригите, која претставува $P2P$ верзија на електронски кеш,

што овозможува директно електронско плаќање од една страна на друга без потреба од вклучување на трета страна; размената на пари, имот акции или која било вредност на транспарентен, без конфликтен начин, меѓу двајца сопственици врз основа на множество услови вклучени во договорот, ја дефинираат втората карактеристика наречена паметен договор; третата главна иновација во блок-веригата претставува скалирањето во блок-веригата, со цел забрзување на процесот на верификација на трансакциите во неа. Низата од блокови што ја формираат блок-веригата е безбедна, непроменлива и транспарентна. Затоа, решенијата што ги користи технологијата на блок-веригите може да биде столб на системите за обработка и процесирање на податоците во рамки на организациите. Интегрирањето на блок-веригите со масивните податоци овозможува подобро управување со огромните количини и разновидност на информациите. Примената на блок-веригите во IoT им овозможува на IoT уредите да учествуваат во трансакциите на блок-веригите и да пронаоѓаат нови стилови на дигитални итеракции. Блок-веригите во IoT може да се користат за следење на поврзаните уреди, обработка на трансакциите, координација меѓу уредите, итн.

Влијанието на квазигрупите и квазигрупните трансформации во дизајнот на криптографските примитиви, кодови за откривање и кодови за корекција на грешки, беше објасното во првата глава. Со цел да се обезбеди побрз и поедноставен процес на декодирање во мрежното кодирање, во седмата глава ќе дадеме една примена на овие квазигрупни трансформации во овој вид на кодирање. На почетокот на оваа глава, дадено е објаснување за мрежното кодирање како една ветувачка технологија која ја подобрува мрежната моќност и обезбедува голема сигурност. Најпопуларно мрежно кодирање е линеарното кодирање на мрежата. Во линеарното мрежно кодирање, во процесот на декодирање на пораките се користи Гаусов метод на елиминации. Сложеноста на декодирањето поради користење на Гаусов метод на елиминации, ни ја даде идејата за конструкција на алгоритам за мрежно кодирање базиран на квазигрупи. Во дефинирањето на алгоритамот базиран на квазигрупи ја користиме квазигрупата $(Q, *)$ од ред $2^n$ заедно со нејзината парастрофа $\backslash$, $E_l$ - трансформација со лидер $l$ за кодирање и шифрирање пораки и $D_l$ - трансформација со истиот лидер $l$ за декодирање и дешифрирање. Овој алгоритам е поедноставен и побрз отколку алгоритмите што се користат во линераното мрежно кодирање, поради тоа што во процесот на декодирање применуваме само $D$ трансформација, наместо да решаваме систем линеарни равенки користејќи метод на Гаусова елиминација. Нови резултати во овој дел се: дефинирање нов алгоритам за мрежно кодирање базиран на ква-

зигрупи, а со цел зголемување на пропусната моќ во ”пеперутка” комуникациската мрежа.

Главната книга на трансакции на блок-веригата содржи запис за секоја трансакција некогаш направена. Со зголемување на бројот на трансакции, просторот за складирање брзо се зголемува и со тоа растат трошоците за складирање на целата мрежа на блок-веригата. Поради тоа, има потреба од промена на концептот на складирање на блок-веригата. Неколку вакви предлози се дадени претходните години, со цел подобрување на перформансите на блок-веригите и намалување на трошоците за складирање, со користење на најразлични техники. Во осмата глава, ќе предложиме за намалување на просторот за складирање во блок-веригите со користење на шемата на Шамир за споделување на тајна. На почетокот на оваа глава, го објаснуваме концептот на работа на шемата на Шамир за споделување на тајна, а потоа дефинираме алгоритам за намалување на просторот за складирање на блок-веригите со помош на оваа шема, што е нов резултат во оваа глава.

# Глава 1

# Квазигрупи и квазигрупни трансформации

Во овој дел ќе дадеме краток преглед на квазигрупите. Повеќе детали за квазигрупите и нивните својства се дадени во [2], [11], [62]. Со помош на квазигрупите дефинирани се голем број стринг трансформации во [64], [66], [67], [68], [60], [12]. Тие се наречени квазигрупни стринг трансформации. Овие квазигрупни трансформации се користат за конструкција на криптографски примитиви и за дизајнирање на кодови за откривање и поправање на грешки.

## 1.1 Квазигрупи и нивни својства корисни во криптографија и теорија на кодирање

Квазигрупа $(Q, *)$ е групоид, т.е. множество $Q$ со бинарна операција $* : Q^2 \to Q$, така што за сите $u, v \in Q$, важи равенството:

$$(\forall u, v \in Q)(\exists! x, y \in Q) \ \ (x * u = v \ \& \ u * y = v) \tag{1.1}$$

Всушност, со (1.1) велиме дека групоид $(Q, *)$ е квазигрупа ако и само ако равенствата $x * u = v$ и $u * y = v$ имаат единствени решенија $x$ и $y$ за кои било фиксни $u, v \in Q$.

Понатаму ќе претпоставиме дека множеството $Q$ е конечно и дека $(Q, *)$ е дадена квазигрупата. Од операцијата $*$ може да се изведат пет нови квазигрупни операции, наречени *парастрофи*. За кодовите разгледувани во оваа докторска дисертација, потребна е само една, означена со "$\backslash$" која е дефини-

рана на следниов начин:

$$x * y = z \quad \Longleftrightarrow \quad y = x \setminus z.$$

Алгебрата $(Q, *, \setminus)$ ги задоволува идентитетите:

$$x \setminus (x * y) = y, \quad x * (x \setminus y) = y \tag{1.2}$$

и $(Q, \setminus)$ е исто така квазигрупа.

Понатаму, ќе дадеме некои својства на квазигрупите.

**Дефиниција 1.1.** *Квазигрупата $(Q, *)$ е комутативна ако и само ако го задоволува идентитетот*

$$x * y = y * x. \tag{1.3}$$

**Дефиниција 1.2.** *Квазигрупата $(Q, *)$ е асоцијативна ако и само ако го задоволува идентитетот*

$$(x * y) * z = x * (y * z) \tag{1.4}$$

**Дефиниција 1.3.** *Квазигрупата $(Q, *)$ е лева лупа ако постои лева единица $e \in Q$ така што*

$$(\forall\, x \in Q)\ (e * x = x) \tag{1.5}$$

**Дефиниција 1.4.** *Квазигрупата $(Q, *)$ е десна лупа ако постои десна единица $e \in Q$ така што*

$$(\forall\, x \in Q)\ (x * e = x) \tag{1.6}$$

**Дефиниција 1.5.** *Квазигрупата $(Q, *)$ е лупа ако постои единица $e \in Q$ така што*

$$(\forall\, x \in Q)\ (x * e = x = e * x) \tag{1.7}$$

Може да се забележи дека $(Q, *)$ е лупа акко таа е лева лупа и десна лупа.

**Дефиниција 1.6.** *Квазигрупата $(Q, *)$ е идемпотентна ако и само ако го задоволува идентитетот*

$$x * x = x \tag{1.8}$$

Квазигрупа $(Q, *)$ од ред $n$ е *без облик* ако и само ако е не-идемпотентна, не-комутативна, не-асоцијативна, нема ниту лева ниту десна единица, не содржи соодветни под-квазигрупи и не постои $k < 2n$ за кој се задоволени следните идентитети:

$$\underbrace{x(...*(x}_{k}*y)) = y,\ y = ((y*\underbrace{x)*...)*x}_{k}. \tag{1.9}$$

Условот $k < 2n$ за идентитетите (1.9) значи дека која било лева или десна транслација на квазигрупата $(Q, *)$ мора да биде од ред $k \geqslant 2n + 1$. За криптографски цели, подобро е да се избере квазигрупа без облик [17].

Табела 1.1: Број на квазигрупи од ред $n \leqslant 11$

| $n$ | $Q_n$ |
|---|---|
| 1 | 1 |
| 2 | 2 |
| 3 | 12 |
| 4 | 576 |
| 5 | 161280 |
| 6 | 812851200 |
| 7 | 61479419904000 |
| 8 | 108776032459082956800 |
| 9 | 5524751496156892842531225600 |
| 10 | 9982437658213039871725064756920320000 |
| 11 | 776966836171770144107444346734230682311065600000 |

## 1.2 Квазигрупни трансформации

Во овој дел, ќе ги опишеме квазигрупните трансформации кои ќе ги користиме во следните поглавја за дефинирање на класата на кодовите за корекција на грешки. За дадена азбуката $Q$, множеството од сите конечни стрингови од елементите на $Q$ ќе го означиме со $Q^+$, т.е. $Q^+ = \{a_1a_2 \dots a_n | a_i \in Q, n \in \mathbb{N}\}$ или $Q^+ = Q \cup Q^2 \cup Q^3 \dots$. Понатаму, ќе претпоставиме дека редот на квазигрупата е степен од 2, т.е. дека редот е $2^k$, за некое $k$. На тој начин,

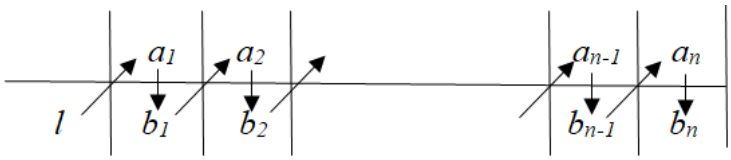
кој било елемент од $Q$ можеме да го запишеме како $k$-торка од битови. Ако $k = 4$, елементите на $Q$ се нибли.

Во [67], со користење на квазигрупната операција $*$ и нејзината парастрофа $\backslash$, дефинирани се две *квазигрупни трансформации*. Тоа се $e$-трансформација и $d$-трансформација, кои се дефинираат во $Q^+$, каде $\mid Q \mid \geqslant 2$.

- *$e$-трансформација* е функција во $Q^+$, која се дефинира на следниот начин.
  За фиксен елемент $l \in Q$, кој се нарекува *лидер* и $a_i \in Q$, $i = 1, 2, \ldots, n$.

$$e_{l,*}(a_1 \ldots a_n) = b_1 \ldots b_n \Leftrightarrow \begin{cases} b_1 & = & l * a_1, \\ b_{i+1} & = & b_i * a_{i+1}, \quad i = 1, 2, \ldots n-1 \end{cases} \tag{1.10}$$

  Графичкото претставување на $e_{l,*}$ е дадено на Слика1.1.



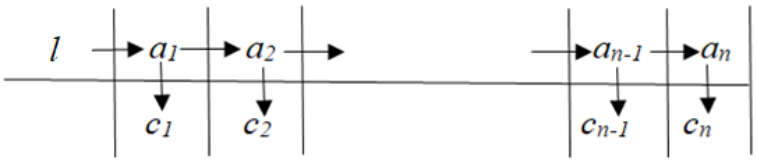

Слика 1.1: Графичко претставување на функцијата $e_{l,*}$

- *$d$-трансформација* е функција $d_{l,\backslash} : Q^+ \to Q^+$ која се дефинира на следниот начин.
  За избран лидер $l$ и $a_i \in Q$, $i = 1, 2, \ldots, n$,

$$d_{l,\backslash}(a_1 \ldots a_n) = c_1 \ldots c_n \Leftrightarrow \begin{cases} c_1 & = & l \setminus a_1, \\ c_{i+1} & = & a_i \setminus a_{i+1}, \quad i = 1, 2, \ldots n-1 \end{cases} \tag{1.11}$$

  Графичкото претставување на $d_{l,\backslash}$ е дадено на Слика 1.2.

Како последица на равенството (1.2) го имаме следново својство:




Слика 1.2: Графичко претставување на функцијата $d_{l,\backslash}$

**Својство 1.1.** ([67]) *Пресликувањата $e$ и $d$ се биекции (пермутации) и притоа за секој стринг $\alpha \in Q^+$, важи*

$$d_{l,\backslash}(e_{l,*}(\alpha)) = \alpha = e_{l,*}(d_{l,\backslash}(\alpha))$$

*т.е. $d_{l,\backslash} = e_{l,*}^{-1}$ е инверзна функција на $e$ .*

Нека $*_1, *_2, ..., *_n$ се квазигрупни операции дефинирани над $Q$ и $\backslash_1, \backslash_2, ...,$ $\backslash_n$ се соодветните парастрофни операции дефинирани над $Q$. Нека трансформациите $e_{l_1,*_1}, e_{l_2,*_2}, ..., e_{l_n,*_n}$ се дефинирани на исти начин како во (1.10), а $d_{l_1,\backslash_1}, d_{l_2,\backslash_2}, ..., d_{l_n,\backslash_n}$ како во (1.11), со избор на фиксни елементи $l_1, l_2,\ldots,l_n \in Q$.

Ги формираме следните композиции:

$$E = e_{l_n,*_n} \circ e_{l_{n-1},*_{n-1}} \circ, ..., \circ e_{l_1,*_1}$$

$$D = d_{l_1,\backslash_1} \circ d_{l_2,\backslash_2} \circ, ..., \circ d_{l_n,\backslash_n}$$

Како последица на Својство 1.1., имаме дека $E$ и $D$ се биекции кои се инверзни една на друга.

## 1.3 Криптографски својства на квазигрупните трансформации

Во овој дел ќе разгледаме некои криптографски својства на квазигрупните ($E$- и $D$-) трансформации. $E$-трансформацијата може да се употреби за дизајнирање на алгоритми за шифрирање, а $D$-трансформацијата - за алгоритми за дешифрирање. Нека $M$ е оригинална порака и $C = E(M)$ е соодветната шифрирана порака. Откривањето на пораката $M$ од пораката $C$ е невозможно

без да се знаат квазигрупите што се користат при E-трансформацијата. Нападот со груба сили (brute force) за пронаоѓање на овие квазигрупи е исто така невозможен поради големиот број квазигрупи (особено, за голем ред $n$).

– Едно од најважните криптографски својства на квазигрупните трансформациите е *рамномерната распределба* на $n$-торките (парови, тројки и така натаму) во шифрираната низа што овозможува отпорност од статистички напад. Ова својство на квазигрупната $E-$ трансформација е дадено во [67] каде што авторите го докажаа следното својство.

  Нека $(Q, *)$ е дадена конечна квазигрупа и $\mathbf{p} = (p_1, p_2, ..., p_{|Q|})$ е распределба на веројатноста на симболите од $Q$, така што $p_i > 0$ за секое $i$ и $\sum_i p_i = 1$. Докажана е следната теорема.

  **Теорема 1.** ([67]) *Нека $M = a_1 a_2 \dots a_n$ $(a_i \in Q)$ е дадена случајна низа и нека* **p** *е вектор на веројатности на симболите $a_i$, за $a_i \in Q$. Нека $C$ е добиена после $k$ примени на $e-$ трансформација на $M$. Ако $n$ е доволно голем цел број тогаш распределбата на подстринговите со должина $t$ на $C$ е рамномерна за секое $1 \leqslant t \leqslant k$.* (За $t > k$ распределбата на подстринговите на $C$ со должина $t$ не е рамномерна.)

– Друго важно криптографско својство на криптираната низа $C$ е нејзиниот *период*. Во [69] покажано е дека периодот на низи процесирани со квазигрупни трансформации расте најмалку линеарно и зголемувањето на периодот зависи од избраната квазигрупа. Повеќе за периодот на стринговите добиени со квазигрупни трансформации дадено е во [13].

## 1.4 Класификација на квазигрупи според фракталност и линеарност

Резултатите од постоечките истражувања кажуваат дека добрите криптографски својства на квазигрупно процесираната низа зависат од избраните квазигрупи. Затоа, класификациите на квазигрупите се многу важни за нивна успешна примена во криптографијата и теоријата на кодирање. Тоа е тежок проблем, бидејќи бројот на квазигрупи (дури и со мал ред) е многу голем.

Класификацијата на квазигрупите со графичко прикажување на квазигрупно процесираната низа е дадена во [14]. Во тој труд, авторите го делат множеството на сите квазигрупи од ред $4$ на две дисјунктни класи, класата на

таканаречени *фрактални* квазигрупи (ако графичката презентација на квазигрупно процесираната низа има фрактална структура) и класа на таканаречени *нефрактални* квазигрупи (ако графичката презентација на квазигрупно процесираната низа нема фрактална структура). Класата на фракталните квазигрупи не се препорачува за дизајнирање на криптографски примитиви. Бројот на фрактални квазигрупи од ред 4 е 192, бројот на нефрактални квазигрупи е 384.

Во [15], авторите даваат репрезентација на квазигрупите со векторско вредносни Булови функции. Квазигрупа $(Q, *)$ од ред $2^n$ може да биде претставена со векторско вредносна Булова функција $f : \{0,1\}^{2n} \to \{0,1\}^n$ на следниот начин. Нека произволен елемент $x$ од квазигрупата е претставен како бинарен вектор $x = (x_1, x_2, \ldots, x_n) \in \{0,1\}^n$. Тогаш, за секои $x, y \in Q$

$$x * y \equiv f(x_1, \ldots, x_{2n}) = (f_1(x_1, \ldots, x_{2n}), \ldots, f_n(x_1, \ldots, x_{2n})) \tag{1.12}$$

каде што

$$x = (x_1, x_2, \ldots, x_n),\ y = (x_{n+1}, x_{n+2}, \ldots, x_{2n})$$

и

$$f_i : \{0,1\}^{2n} \to \{0,1\}$$

се соодветните компоненти на $f$.

Со користење на Буловата репрезентација во [15] квазигрупите се поделени во две класи: линеарни квазигрупи и нелинеарни квазигрупи.

**Дефиниција 1.7.** *Квазигрупата е линеарна, ако сите функции $f_i$ за $i = 1, 2, \ldots n$ се линеарни полиноми.*

**Дефиниција 1.8.** *Квазигрупата е нелинеарна, ако постои функција $f_i$ за некое $i = 1, 2, \ldots n$ која не е линеарна.*

Нелинеарните квазигрупи се поделени во две подкласи: делумно (или слабо) нелинеарни и чисто нелинеарни квазигрупи. Делумно нелинеарни се оние квазигрупи кои имаат барем една компонента која е линеарна Булова функција и барем една компонента која е нелинеарна Булова функција. Квазигрупата е чисто нелинеарна ако сите нејзини компоненти се нелинеарни Булови функции.

# Глава 2

# Случајни кодови базирани на квазигрупи

Првичната идеја за примена на квазигрупите во случајните кодови е дадена во [16] и [17], каде што се предложени Случајните кодови базирани на квазигрупи (Random Codes Based on Quasigroups - RCBQ). RCBQ се всушност криптокодови т.е. тие се дефинирани со користење на криптографски алгоритам во самиот процес на кодирање/декодирање. Овие кодови со само еден алгоритам овозможуваат не само поправање на одреден број на грешки кои настануваат при пренос низ канал со пречки, туку истовремено обезбедуваат и тајност на информациите. Примателот на информацијата до оригиналните податоци може да дојде само доколку знае точно кои параметри се користени во процесот на кодирање, дури и ако комуникацискиот канал е без пречки. Во оваа дисертација ги разгледуваме RCBQ како кодови кои поправаат грешки и затоа тука нема да бидат анализирани нивните криптографски својства.

Алгоритмите за кодирање/декодирање предложени во [16] и [17] ќе ги нарекуваме Стандарден алгоритам на RCBQ. Тие вклучуваат неколку параметри и нивните перформанси зависат од избраните параметри. Влијанието на параметрите врз перформансите на овие кодови е проучено во [88]. Во [85] авторите ги споредуваат перформансите на овие кодови со перформансите на Ред-Милер (RMC) и Ред-Соломон (RSC) кодовите, при пренос низ бинарно симетричен канал. Од добиените резултати авторите заклучуваат дека RMC и RSC имаат подобри перформанси за декодирање во бинарен симетричен канал со веројатност за бит грешка $p < 0.05$. За поголеми вредности на $p$, со RCBQ се добиваат значително подобри резултати. Сепак, временската ефикасност на RMC и RSC е многу подобра од онаа на RCBQ.

Значи, брзината на декодирање е најголем недостаток на RCBQ и претставува предизвик за понатамошно истражување. Со цел да се подобрат перформансите на овие кодови, авторите во [89], [90], [91] [107], дефинираат нови алгоритми за кодирање/декодирање наречени: алгоритам-за-декодирање-со-пресеци (Cut-Decoding Algorithm или кратко АДП) и алгоритам-за-декодирање-со-4пресеци (4-Sets-Cut-Decoding Algorithm или кратко АД4П). Во сите овие трудови, проучувани се перформансите на случајните кодови базирани на квазигрупи при пренос низ бинарен симетричен канал.

## 2.1 Опис на случајните кодови базирани на квазигрупи

Во овој дел ќе биде опишан Стандардниот алгоритам за Случајните кодови базирани на квазигрупи кој е предложен од Глигороски, Марковски и Коцарев ([16], [17]).

### 2.1.1 Тотално асинхрон проточен шифрувач

Случајните кодови базирани на квазигрупи, се темелат на класата тотално асинхрони проточни шифрувачи ($TASC$), чиј концепт е дефиниран во [17], на следниот начин:

**Дефиниција 2.1.** *Тотално асинхрон проточен шифрувач е оној во кој низата клучеви се генерира како функција од меѓу-клучот и сите претходни букви во влезната порака.*

*Функцијата за шифрирање на еден тотално асинхрон проточен шифрувач може да биде опишана со следните равенки:*

$$k^{(i+1)} = f(k^{(i)}, m_i), c_i = h(k^{(i)}, m_i),$$

*каде $k^{(0)}$ е почетниот таен клуч, $k^{(i)}$ се меѓу-клучевите, $f$ е функцијата за определување на следниот клуч и $h$ е излезната функција. Функцијата за дешифрирање на тотално асинхрон проточен шифрувач може да се опише со следните равенки:*

$$k^{(i+1)} = f'(k^{(i)}, c_i), m_i = h'(k^{(i)}, c_i).$$

Од дефиницијата јасно е дека тотално асинхрон проточен шифрувач дава шифриран текст $c_i$ кој зависи од сите претходни букви $m_0, m_1, ..., m_i$ од влезниот текст. Авторите на Случајните кодови базирани на квазигрупи имаат дефинирано една можна имплементација на TASC со користење на квазигрупни стринг трансформации, која е наречена EdonZ.

### 2.1.2 Процес на кодирање со Стандардниот алгоритам

Нека $Q$ е множество од сите $a$-бит симболи, т.е. $Q$ има $2^a$ букви, $(Q, *)$ е дадена квазигрупа и $(Q, \backslash)$ е нејзината парастрофа. Пред да започне кодирањето со Стандардниот алгоритам за RCBQ, низата од битови добиена од изворот на информации се дели во блокови со должина $N_{block}$ бита и нека $M$ е еден таков блок. Ќе земеме дека $M = M_1 M_2 \dots M_l$, каде што $M_i \in Q$, т.е. $M_i$ се симболи од $a$ бита. Оттука е јасно дека $N_{block} = la$.

Потоа процесот на кодирање се изведува во следниве чекори:

– Пораката $M$ добиена од изворот се проширува со додавање на редундантни информации. Со тоа секоја пораката со должина $N_{block}$ се пресликува во порака со должина $N > N_{block}$. Тоа може да се изведе на многу различни начини, но најчесто се додаваат само нули кои според одредени патерни соодветно се разместуваат во пораката, при што ја добиваме проширената порака:

$$L = L^{(1)} L^{(2)} ... L^{(s)} = L_1 L_2 ... L_m,$$

каде $L_i \in Q$, а $L^{(i)}$ се подблокови од $r$ симболи. Притоа, $N = ma$ и $m = rs$. На ваков начин, се добива $(N_{block}, N)$ код со рата $R = N_{block}/N$.

– Добиената проширена порака се кодира со користење на квазигрупни трансформации, т.е. со алгоритамот за шифрирање на EdonZ, прикажан на Слика 2.1.

– На овој начин се добива кодниот збор

$$C = C_1 C_2 ... C_m,$$

каде $C_i \in Q$.

| **Шифрирање** | **Дешифрирање** |
|---|---|
| **Влез**: Клуч $k = k_1k_2 \dots k_n$ и $L = L_1L_2 \dots L_m$<br>**Излез**: коден збор $C = C_1 ... C_m$ | **Влез**: Парот $(a_1a_2 \dots a_s, k_1k_2 \dots k_n)$<br>**Излез**: Парот $(c_1c_2 \dots c_s, K_1K_2 \dots K_n)$ |
| За $j = 1$ to $m$<br>$X \leftarrow L_j$;<br>$T \leftarrow 0$;<br>За $i = 1$ до $n$<br>$X \leftarrow k_i * X$;<br>$T \leftarrow T \oplus X$;<br>$k_i \leftarrow X$;<br>$k_n \leftarrow T$<br>**Излез**: $C_j \leftarrow X$ | За $i = 1$ до $n$<br>$K_i \leftarrow k_i$;<br>За $j = 0$ до $s - 1$<br>$X, T \leftarrow a_{j+1}$;<br>$temp \leftarrow K_n$;<br>За $i = n$ до 2<br>$X \leftarrow temp \setminus X$;<br>$T \leftarrow T \oplus X$;<br>$temp \leftarrow K_{i-1}$;<br>$K_{i-1} \leftarrow X$;<br>$X \leftarrow temp \setminus X$;<br>$K_n \leftarrow T$;<br>$c_{j+1} \leftarrow X$;<br>**Излез**: $(c_1c_2 \dots c_s, K_1K_2 \dots K_n)$ |

Слика 2.1: Алгоритми за шифрирање и дешифрирање

### 2.1.3 Процес на декодирање со Стандардниот алгоритам

Декодирањето е, всушност, постапка во која треба да се вратат (ако тоа е можно) пратените кодни зборови. Бидејќи при преносот се јавуваат грешки, декодирањето е можно само со користење на редундантната информација. Во овој код, при декодирањето се користи фактот дека некои букви на одредени позиции во проширената порака се нули. Декодирањето е проточно и се декодира последователно блок по блок. Овој тип на декодирање овозможува дел од кодниот збор да биде декодиран кога е невозможно да се декодира целиот коден збор. После преносот на кодниот збор $C$ низ каналот со пречки, примената порака е $D = D^{(1)}D^{(2)}...D^{(s)} = D_1D_2 \dots D_m$, каде $D_i \in Q$, а $D^{(i)}$ се подблокови од $r$ симболи и $m = rs$. Процесот на декодирање се состои од четири чекори: $(i)$ процедура за генерирање на множества со преддефинирано Хамингово растојание, $(ii)$ инверзен алгоритам за кодирање, $(iii)$ процедура за генерирање на множествата со кандидати за декодирање и $(iv)$ правило за декодирање.

– *Процедура за генерирање на множества со преддефинирано Хамингово растојание*

Веројатноста дека во $D^{(i)}$ најмногу $t$ битови ќе бидат погрешно пренесени изнесува

$$P(p;t) = \sum_{k=0}^{t} \binom{ra}{k} p^k (1-p)^{ra-k},$$

каде $p$ е веројатноста за бит-грешка при пренос низ каналот. Нека $B_{max}$ е цел број што го означува претпоставениот максималниот број на бит-грешки што може да се појават во блокот за време на преносот. Се формираат множествата $H_i = \{\alpha | \alpha \in Q^r, \quad H(D^{(i)}, \alpha) \leqslant B_{max}\}$, за $i = 1, 2, \ldots, s$, каде $H(D^{(i)}, \alpha)$ е Хаминговото растојание помеѓу $D^{(i)}$ и $\alpha$. Бројот на елементи во множествата $H_i$ е

$$B_{checks} = 1 + \binom{ra}{1} + \binom{ra}{2} + ... + \binom{ra}{B_{max}}$$

и бројот $B_{checks}$ ја одредува комплексноста на процесот на декодирање: за да се најде елементот $C^{(i)}$ во множеството $H_i$, треба да се направат најмногу $B_{checks}$ проверки. Оттука, јасно е дека за ефикасно декодирање бројот на проверки $B_{checks}$ треба да се намали колку што е можно повеќе.

– *Инверзен алгоритам на алгоритамот за кодирање*

Инверзниот алгоритам на алгоритмот за кодирање (ICA) е всушност алгоритамот за дешифрирање на TASC даден на Слика 2.1.

– *Генерирање на множествата со кандидати за декодирање*

Множествата кандидати за декодирање $S_0$, $S_1$, $S_2$,..., $S_s$, се дефинираат итеративно. Нека $S_0 = (k_1 \ldots k_n; \lambda)$, каде $\lambda$ е празен стринг. Нека $S_{i-1}$ е дефинирано за $i \geqslant 1$. Тогаш $S_i$ е множеството од сите парови $(\delta, w_1 w_2 \ldots w_{rai})$ добиени со користење на множествата $S_{i-1}$ и $H_i$ на следниот начин ($w_j$ се битови). За секој елемент $\alpha \in H_i$ и за секој $(\beta, w_1 w_2 \ldots w_{ra(i-1)}) \in S_{i-1}$, се применува инверзниот алгоритам за кодирање со влез $(\alpha, \beta)$. Ако излез од алгоритмот е парот $(\gamma, \delta)$ и притоа двете низи $\gamma$ и $L^{(i)}$ имаат редундантни нулти симболи на исти позиции,

тогаш парот $(\delta, w_1 w_2 \dots w_{ra(i-1)} c_1 c_2 \dots c_r) \equiv$
$(\delta, w_1 w_2 \dots w_{rai})$ ($c_i \in Q$) е елемент на множеството $S_i$.

– *Правило за декодирање*

Декодирањето на примената порака $D$ е дадено со следното правило:

- Ако множеството $S_s$ содржи само еден елемент $(d_1 \dots d_n, w_1 \dots w_{ras})$ тогаш $L = w_1 \dots w_{ras}$ е декодираната (редундантна) порака. Во тој случај, велиме дека имаме *успешно декодирање*.
- Ако добиената декодирана порака не е точна тогаш имаме *недетектирана-грешка*.
- Во случај кога множеството $S_s$ содржи повеќе од еден елемент, велиме дека декодирањето на $D$ е неуспешно (се појавила грешка од тип *грешка-повеќе-кандидати*).
- Во случај кога $S_j = \emptyset$, за некое $j \in \{1, \dots, s\}$, тогаш процесот на декодирање се стопира (велиме дека се појавила грешка од типот *грешка-празно-множество*). Можеме да се заклучиме дека при преносот на $D^{(i)}$, за некое $i \leqslant j$, се појавиле повеќе од $B_{max}$ грешки и затоа $C_i \notin H_i$.

## 2.2 Алгоритам за декодирање со пресеци

А. Поповска-Митровиќ, С. Марковски и В. Бакева во својот труд [88] го проучуваат влијанието на параметрите на Случајните Кодови Базирани на Квазигрупи врз перформансите на кодовите. Тие доаѓаат до заклучок дека сите параметри кои се користат во кодот (како, патернот, должината на клучот, квазигрупата) влијаат на перформансите на Случајните кодови базирани на квазигрупи. Но, најголем недостаток на овие кодови е брзината на декодирање. Со цел да се подобри брзината на декодирање, овие тројца автори (во [89]) дефинираат нов алгоритам за кодирање/декодирање наречен алгоритам-за-декодирање-со-пресек (Cut-Decoding или АДП алгоритам) и ги проучуваат неговите перформанси (во [90], [91]) при пренос низ бинарен симетричен канал. Бидејќи декодирањето на RCBQ е всушност декодирање со листа (list decoding), брзината на декодирањето и веројатноста за точно декодирање зависи од големината на листите со можни кандидати за декодираната порака. Затоа, во АДП алгоритам направени се модификации со цел да

се намали бројот на кандидати за декодирање во сите итерации од процесот на декодирање. Во овој алгоритам, се применуваат две трансформации на редундантната порака со користење на различни параметри, а кандидатите за декодирана порака се добиваат со барање пресек на соодветните множества $S$. На овој начин процесот на декодирање за код (72,288) е 4.5 пати побрз од оригиналниот алгоритам ([89], [107]).

### 2.2.1 Кодирање со АДП алгоритамот

Во Стандардниот алгоритам за кодирање, опишан во претходната глава, користевме код $(N_{block}, N)$ со рата $R = N_{block}/N$. Во АДП алгоритамот, се користат два $(N_{block}, N/2)$ кодови со рата $2R$, со кои се кодира/декодира иста порака од $N_{block}$ битови. Кодирањето се состои од следните чекори:

– Влезната порака $M = M_1 M_2 ... M_l$ се проширува со додавање на $\nu$ редундантни нулти симболи (на ист начин како и во Стандардниот алгоритам за кодирање) и на тој начин се добива редундантната порака $L = L^{(1)} L^{(2)} ... L^{(s/2)} = L_1 L_2 ... L_{m/2}$ од $N/2$ бита, каде што $L^{(i)}$ е подблок од $r$ симболи од азбуката $Q$, а $L_i \in Q$. Притоа, $N = am$, $m = rs$, а $m = l + \nu$.

– Потоа, за кодирање два пати го применуваме алгоритамот за шифрирање даден на Слика 2.1, на иста редундантна порака $L$, користејќи различни параметри (различни клучеви или квазигрупи).

– На овој начин го добиваме кодниот збор за влезната порака $M$ како конкатенација на два кодни зборови од $N/2$ битови, т.е.,

$$C = C_1 C_2 ... C_{m/2} C_{m/2+1} ... C_m,$$

каде $C_i \in Q$.

### 2.2.2 Декодирање со АДП алгоритамот

Кодираната порака се пренесува низ канал со пречки и ја добиваме излезната порака $D = D^{(1)} D^{(2)} ... D^{(s)}$, каде што $D^{(i)}$ се подблокови од $r$ симболи. Пораката $D$ се дели на две пораки $D_1 = D^{(1)} D^{(2)} ... D^{(s/2)}$ и $D_2 = D^{(s/2+1)} D^{(s/2+2)} ... D^{(s)}$ со еднаква должина. Потоа, овие две пораки се декодираат паралелно со соодветните параметри. Процесот на декодирање се состои од

истите 4 чекори како и кај Стандардниот алгоритам, со тоа што е направена модификација во процедурата за генерирање на множествата со кандидати за декодирање. За секој подблок од двете пораки се прават паралелно чекорите во Табела 2.1.

| | |
|---|---|
| 1. Ги формираме множествата $H_i^{(1)}$ | 1. Ги формираме множествата $H_i^{(2)}$ |
| 2. За секое $\alpha_1 \in H_i^{(1)}$, го применуваме ICA $(\alpha_1, k_1)$ и нека $(\beta_1, k_1')$=ICA$(\alpha_1, k_1)$. | 2. За секое $\alpha_2 \in H_i^{(2)}$, го применуваме ICA $(\alpha_2, k_2)$ и нека $(\beta_2, k_2')$=ICA$(\alpha_2, k_2)$. |
| 2.1. Провериме дали во $\beta_1$ и $L^{(i)}$ редундантни симболи се на иста позиција. Ако се, тогаш $(k_1', \beta_1) \in S_i^{(1)}$ | 2.1. Провериме дали во $\beta_2$ и $L^{(i)}$ редундантни симболи се на иста позиција. Ако се, тогаш $(k_2', \beta_2) \in S_i^{(2)}$ |

Табела 2.1: Генерирање множества со кандидати за декодирање

– Нека во двата паралелни процеси на декодирање се добиени множествата $S_i^{(1)}$ и $S_i^{(2)}$ со кандидати за декодирање, на ист начин како и во Стандардниот алгоритам на RCBQ.

– Нека

$$V_j = \{w_1 w_2 \dots w_{ri} | (\delta, w_1 w_2 \dots w_{rai}) \in S_i^{(j)}\}, j = 1, 2$$

и $V = V_1 \cap V_2$.

– За секој $(\delta, w_1 w_2 \dots w_{rai}) \in S_i^{(1)}$, ако $w_1 w_2 \dots w_{rai} \notin V$, тогаш

$$S_i^{(1)} \leftarrow S_i^{(1)} \setminus \{(\delta, w_1 w_2 \dots w_{rai})\}.$$

Исто така, за секој $(\delta, w_1 w_2 \dots w_{rai}) \in S_i^{(2)}$, ако $w_1 w_2 \dots w_{rai} \notin V$, тогаш

$$S_i^{(2)} \leftarrow S_i^{(2)} \setminus \{(\delta, w_1 w_2 \dots w_{rai})\}.$$

Ова значи дека од $S_i^{(1)}$ се елиминираат сите елементи чиј втор дел не е еднаков на вториот дел на некој елемент во $S_i^{(2)}$, и обратно. Во следната итерација и двата процеси ги користат соодветните редуцирани множества $S_i^{(1)}$ и $S_i^{(2)}$.

– Постапката се повторува за секое $i \leqslant s/2$.

Правилото за декодирање во АДП алгоритамот е дефинирано на следниот начин.

- Ако после последната итерација, редуцираните множества $S_{s/2}^{(1)}$ и $S_{s/2}^{(2)}$ имаат само еден елемент со иста втора компонента $w_1 \dots w_{ras/2}$, тогаш $L = w_1 \dots w_{ras/2}$ е декодираната редундантна порака. Во тој случај, велиме дека имаме *успешно декодирање*.

- Ако добиената декодирана порака не е точна тогаш имаме *недетектирана-грешка*.

- Ако во некоја итерација се добие $S_i^{(1)} = \emptyset$, $S_i^{(2)} \neq \emptyset$ или $S_i^{(2)} = \emptyset$, $S_i^{(1)} \neq \emptyset$, тогаш декодирањето на пораката продолжува само со непразното множество $S_i^{(2)}$ или $S_i^{(1)}$, со користење на стандардниот алгоритам за декодирање на RCBQ.

- Ако во некоја итерација $S_i^{(1)} = S_i^{(2)} = \emptyset$, тогаш процесот на декодирање се стопира и велиме дека се појавила грешка од типот *грешка-празно-множество*.

- Ако после последната итерација, редуцираните множества $S_{s/2}^{(1)}$ и $S_{s/2}^{(2)}$ имаат повеќе од еден елемент, тогаш имаме неуспешно декодирање со *грешка-повеќе-кандидати*.

- За решавање на проблемот со поголемиот број на неуспешни декодирања со *грешка-повеќе-кандидати* се користи една хевристика во правилото за декодирање за елиминација на овој тип грешки [107]. Имено, од експериментите со RCBQ може да се види дека кога декодирањето завршува со повеќе елементи во редуцираните множества со кандидати за декодирање добиени во последната итерација, скоро секогаш во овие множества се наоѓа и точната порака. Затоа, во овој случај може случајно да се избере една порака од едно од множествата во последната итерација и таа порака да се прифати како декодирана порака.

Во трудот [91] авторите предложиле методи за намалување на бројот на *грешка-празно-множество* и *грешка-повеќе-кандидати* во АДП алгоритамот:

1. Со цел намалување на бројот на неуспешни декодирања со *грешка-празно-множество* дефиниран е метод со враќање во АДП. Всушност, ако во $i$-тата итерација на процесот на декодирање, се добие дека $S_i^{(1)} = S_i^{(2)} = \emptyset$, (т.е., двете редуцирани множества се празни), тоа значи дека во некој претходен чекор $j < i$ се изгубил точниот блок. Ова се случува ако бројот на грешки за време на пренесувањето на некој блок е поголем од $B_{max}$. Некои од овој тип грешки ќе бидат елиминирани ако се поништат неколку итерации од процесот на декодирање и дел од нив се извршуваат со поголема вредност на $B_{max}$.

2. Предложена е слична модификација со враќање во АДП алгоритамот, за намалување на бројот на неуспешни декодирања со *грешка-повеќе-кандидати*. Ако процесот на декодирање заврши со *грешка-повеќе-кандидати*, тогаш со цел бројот на кандидати да се намали на еден, некоја итерација се извршува со помала вредност на $B_{max}$. Имено, во случајот кога декодирањето завршува со повеќе елементи во редуцираните множества со кандидати за декодирање во последната итерација, тогаш се поништуваат неколку итерации и првата од поништените итерации се извршува со помала вредност на $B_{max}$. Во следните итерации се користи претходната вредност на $B_{max}$.

-

## 2.3 Алгоритам-за-декодирање-со-4пресеци

Подобрување на брзината на декодирање со АДП ја дава идејата за користење на пресеци од повеќе множества $S_i$, со цел да добиеме поголемо зголемување на брзината на декодирање. Па така, авторите на АДП во [92] прават модификации на тој алгоритам и користат четири трансформации на редундантната порака. Со овој нов алгоритам, наречен алгоритам-за-декодирање-со-4пресеци (АД4П) се добива поголемо подобрување на брзината на декодирање. Исто така, со цел да се подобрат и веројатностите за пакет-грешка и бит-грешка дефинирани се неколку методи за генерирање на редуцираните множества со кандидати за декодирање.

### 2.3.1 Кодирање со АД4П алгоритмите

Влезната порака $M = M_1M_2...M_l$ се проширува со додавање редундантни $\nu$ нулти симболи (на ист начин како и во Стандардниот алгоритам и во АДП) и на тој начин се добива редундантната порака $L = L^{(1)}L^{(2)}...L^{(s/4)}$ $= L_1L_2...L_{m/4}$ од $N/4$ бита, каде што $L^{(i)}$ е подблок од $r$ симболи од азбуката $Q$, и $L_i \in Q$. Притоа, $N = am$, $m = rs$ и $m = l + \nu$.

Во оваа модификација на алгоритамот-за-декодирање-со-4пресеци наместо $(N_{block}, N)$ код со рата $R$, користиме четири $(N_{block}, N/4)$ кодови со рата $4R$, кои што ја кодираат/декодираат истата порака од $N_{block}$ бита.

Значи, во процесот на кодирање го применуваме алгоритамот за криптирање, даден на Слика 2.1, на истата редундантна порака $L$ четири пати користејќи различни параметри (различни клучеви или квазигрупи) и на тој начин се добива коден збор $C$ како конкатенација на четири кодни зборови од $N/4$ битови.

### 2.3.2 Декодирање со АД4П алгоритмите

Во [92], авторите предлагаат 4 различни верзии на алгоритам-за-декодирање-со-4пресеци. Најдобри резултати се добиени со користење на алгоритам-за-декодирање-со-4пресеци #3. Во нашите експерименти, ја користиме само оваа верзија и тука накратко ќе ја објасниме. После преносот низ каналот со шум, ја делиме излезната порака $D = D^{(1)}D^{(2)}...D^{(s)}$ на четири пораки $D^1 = D^{(1)}D^{(2)}...D^{(s/4)}$, $D^2 = D^{(s/4+1)}D^{(s/4+2)}...D^{(s/2)}$, $D^3 = D^{(s/2+1)}D^{(s/2+2)}...$ $D^{(3s/4)}$ и $D^4 = D^{(3s/4+1)}D^{(3s/4+2)}...D^{(s)}$ со еднакви должини и ги декодираме четирите пораки паралелно со соодветните параметри. Слично, како и во АДП алгоритамот, во секоја итерација на процесот на декодирање, ги редуцираме множествата кандидати за декодирање добиени во четирите процеси на декодирање, на следниот начин. Нека $S_i^{(1)}$, $S_i^{(2)}$, $S_i^{(3)}$ и $S_i^{(4)}$ се множества кандидати за декодирање добиени во $i$-тата итерација на четирите паралелни процеси на декодирање, $i = 1, \ldots, s/4$. Нека $V_1 = \{w_1w_2 \ldots w_{rai} | (\delta,$ $w_1w_2 \ldots w_{rai}) \in S_i^{(1)}\}$,..., $V_4 = \{w_1w_2 \ldots w_{rai} | (\delta, w_1w_2 \ldots w_{rai}) \in S_i^{(4)}\}$ и $V = V_1 \cap V_2 \cap V_3 \cap V_4$. Ако $V = \emptyset$, тогаш $V = (V_1 \cap V_2 \cap V_3) \cup (V_1 \cap V_2 \cap V_4) \cup$ $(V_1 \cap V_3 \cap V_4) \cup (V_2 \cap V_3 \cap V_4)$. Потоа, од $S_i^{(1)}$, $S_i^{(2)}$, $S_i^{(3)}$ и $S_i^{(4)}$ се отстрануваат сите елементи чиј втор дел не се совпаѓа со некој елемент од $V$.

После последната итерација, ако сите редуцирани множества $S_{s/4}^{(1)}$, $S_{s/4}^{(2)}$, $S_{s/4}^{(3)}$, $S_{s/4}^{(4)}$, имаат само еден елемент со иста втора компонента $w_1 \ldots w_{ras/4}$,

тогаш $L = w_1 \dots w_{ras/4}$ , тогаш оваа компонента е декодираната порака $L$. Во овој случај, велиме дека имаме *успешно декодирање*. Ако декодираната порака не е точната порака, тогаш велиме дека се појавила *недетектирана грешка*. Ако редуцираните множества добиени во последната итерација имаат повеќе од еден елемент тогаш велиме дека се појавила *грешка-повеќе-кандидати*. Ако добиеме $S_i^{(1)} = S_i^{(2)} = S_i^{(3)} = S_i^{(4)} = \emptyset$ во некоја итерација од алгоритам-за-декодирање-со-4пресеци, тогаш процесот ќе заврши (се појавува *грешка-празно-множество*). Но, ако во една итерација добиеме најмалку едно непразно множество кандидати за декодирање, тогаш декодирањето продолжува со непразните множества (редуцираните множества се добиени само со пресек на непразните множества). За решавање на проблемот со поголемиот број на неуспешни декодирања со *грешка-повеќе-кандидати* се користи истата хевристика како и во АДП.

# Глава 3

# Експериментални резултати за RCBQ низ Гаусов канал

Експерименталните резултати добиени со Стандардниот алгоритам и со АДП алгоритамот, со кои се проучуваат перформансите на Случајните кодови базирани на квазигрупи при пренос низ Гаусов канал, беа презентирани во мојата магистерска теза и објавени во трудот [76]. Во овој дел, дадени се експериментални резултати добиени со АД4П, а потоа ги споредуваме овие резултати со соодветните добиени со АДП. Прво, накратко ќе го објасниме Гаусовиот канал за пренос на податоци.

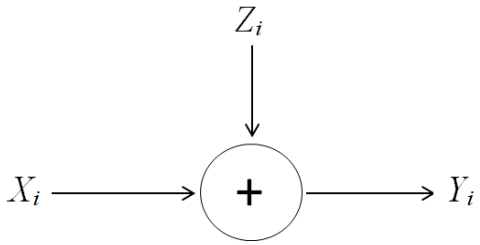
## 3.1 Гаусов канал

Најзначаен непрекинат канал за пренос на податоци е Гаусовиот канал опишан на Слика 3.1. Тој е временски дискретен канал, каде излезот во време $i$ е $Y_i$ и тој излез се добива како сума од влезот $X_i$ и шумот $Z_i$. Шумот е составен од независни и еднакво распределени случајни променливи $Z_i$ со Гаусова распределба со дисперзија $N$ и го нарекуваме *додаден бел Гаусов шум* (*Additive White Gaussian Noise* (AWGN)), а каналот се нарекува Гаусов (AWGN) канал.

Всушност,

$$Y_i = X_i + Z_i \quad Z_i \sim \mathcal{N}(0, N).$$

Шумот $Z_i$ претпоставуваме дека е независен од сигналот $X_i$. Ова е добар модел со кој може да се опишат повеќе видови на комуниципски канали. Ако дисперзијата на шумот е нула, тогаш приемникот го прима пренесениот сигнал коректно. Бидејќи $X$ може да прими која било реална вредност, каналот




Слика 3.1: Гаусов канал

може да пренесе произволен реален број без грешка.

Ако дисперзијата на шумот не е нула и нема ограничување на влезот, можеме да избереме бесконечно подмножество од произволни влезови, така што добиените излези ќе се разликуваат помеѓу себе со мала веројатност на грешка. Ваква шема има бесконечен капацитет [6].

Најчесто ограничување на влезот е ограничување на моќноста (P). За кој било коден збор $(x_1, x_2, ..., x_n)$ пренесен преку каналот, имаме:

$$\frac{1}{n}\sum x_i^2 \leqslant P$$

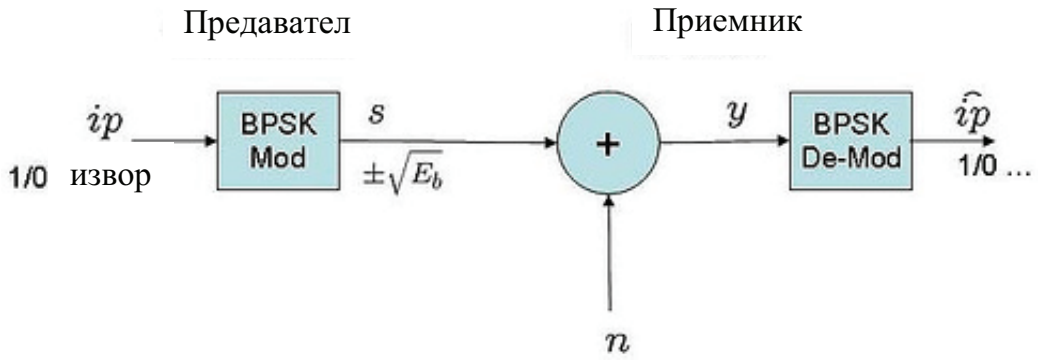
Да претпоставиме дека сакаме да испратиме 1 бит преку каналот. За даденото ограничување на моќноста, најдобро што може да направиме е да испратиме сигнал чија јачина е една од двете аналогни нивоа $+\sqrt{P}$ или $-\sqrt{P}$. Приемникот го гледа соодветниот примен сигнал $Y$ и се обидува да одлучи на кое од двете нивоа е испратениот сигнал. Претпоставувајќи дека двете нивоа се еднакво веројатни (ова е случај ако сакаме да испратиме точно 1 бит информација), оптималното правило за декодирање е да одлучиме дека е пратен сигнал со јачина $+\sqrt{P}$, ако $Y > 0$ или сигнал со јачина $-\sqrt{P}$, ако $Y < 0$.

Дигиталните модулации се користат при пренос на податоците во мобилните телефони, во научните и геомагнетните инструменти, итн. Која било дигитална модулациска шема користи конечен број различни симболи, за да се претстават дигиталните податоци. Една од овие модулации е и PSK (Phase-shift keying). Таа користи конечен број на фази, секоја поврзана со единствен модел (патерн) на бинарни цифри. Обично секоја фаза кодира

еднаков број на битови. Секоја $n$-торка од битови (за фиксно $n$) формира симбол, кој е претставен со одредена фаза. Демодулаторот кој е специјално дизајниран за множеството симболи користени од страна на модулаторот, ја определува фазата на применият сигнал и го пресликува назад во симболите кои го претставуваат, со што се враќаме на оригиналните податоци.

Најпрост облик на PSK модулаторот е BPSK (Binary Phase-shift keying) кој користи само 2 фази. Со BPSK, бинарните цифри 1 и 0 можат да бидат претставени со аналогните нивоа $+\sqrt{E_b}$ и $-\sqrt{E_b}$, соодветно. Овде $P = E_b$ е ограничување на моќноста.

Овој модел е прикажан на Слика 3.2 ([98]).




Слика 3.2: Едноставен блок дијаграм со BPSK предавател-приемник

Симулацијата на оваа модулација е изведена преку AWGN каналот. Притоа, веројатноста за бит-грешка во каналот се пресметува со менување на вредноста на $SNR$ (Signal-to-noise-ratio) на AWGN каналот. Значи, веројатноста за бит-грешка $P_b$ е функција од $SNR$. Вредностите за $P_b$ за вредности на $SNR$ во интервал од $-3$ до 10 децибели се прикажани во Табела 3.1. Од табелата, може да се воочи дека веројатноста за бит-грешка $P_b$ се намалува со зголемување на $SNR$.

| $SNR$ | $P_b$ |
|---|---|
| $-3$ | $0.1584$ |
| $-2$ | $0.1306$ |
| $-1$ | $0.1038$ |
| $0$ | $0.0786$ |
| $1$ | $0.0563$ |
| $2$ | $0.0375$ |
| $3$ | $0.0229$ |
| $4$ | $0.0125$ |
| $5$ | $0.0060$ |
| $6$ | $0.0024$ |
| $7$ | $0.000773$ |
| $8$ | $0.000191$ |
| $9$ | $0.0000336$ |
| $10$ | $0.00000387$ |

Табела 3.1: Веројатност за бит грешка $P_b$

## 3.2 Избирање параметри за оптимален RCBQ

Во овој дел, накратко ќе објасниме како беа направени експериментите добиени со Стандардниот алгоритам и АДП во [76] и како ги избравме параметрите за оптимален RCBQ. Експериментите со случајните кодови базирани на квазигрупи, се изведуваат на следниот начин:

- Најпрво, пораката добиена од изворот се проширува со користење на патерн за додавање на редундантни нулти нибли. Користени се шест различни патерни.

- Проширената порака се кодира со користење на алгоритамот за кодирање (стандардниот или АДП).

- На кодираната порака се врши $BPSK$ модулација со $0 \to -1$ и $1 \to 1$.

- Сигналот се пренесува низ Гаусов канал во кој дејствуваат пречки и поради тоа добиениот сигнал на излезот од каналот не мора да биде ист со влезниот.

- Потоа, се врши демодулација на излезниот сигнал на следниот начин:

  1) ако применият сигнал е поголем од 0, тогаш претпоставуваме дека на влезот од каналот е пуштен бит 1;

  2) ако применият сигнал е помал од 0, тогаш претпоставуваме дека на влезот од каналот е пуштен бит 0;

- Демодулираната порака се декодира со користење на претходно дефинираното правило за декодирање.

- Добиената порака се споредува со почетната и се пресметуваат веројатноста за бит-грешка $BER$ и веројатноста за пакет-грешка $PER$.

Веројатноста за пакет-грешка $PER$ на комуникацискиот канал се пресметува како однос на бројот на погрешно пренесени блокови (пакети) и вкупниот број на блокови пренесени низ каналот. Погрешно пренесени блокови се појавуваат во следните случаи:

1. Ако во последното множество $S_s$ од кандидати за декодирање, има само еден елемент, тој елемент (декодираната порака) се споредува со влезната порака. Ако тие двете се еднакви, тогаш имаме точно декодирање. Доколку декодираната порака се разликува барем во еден бит од влезната, тогаш се јавува недетектирана грешка и пакетот е погрешно пренесен, т.е., се појавува пакет-грешка.

2. Пакет-грешки се јавуваат и во другите видови на неуспешно декодирање, т.е., при *грешка-празно-множество* и *грешка-повеќе-кандидати*.

Вкупниот број на погрешно пренесени блокови е збирот од погрешно пренесените блокови во претходните два случаи.

Веројатноста за бит-грешка $BER$ на комуникацискиот канал се пресметува како однос на бројот на погрешно пренесени битови и вкупниот број на битови пренесени низ каналот. Вкупниот број на погрешно пренесени битови е збир на погрешно пренесените битови во следните два случаи:

1. При успешно декодирање се споредуваат декодираната и влезната порака и се определува бројот на битови во кои тие се разликуваат.

2. Во случај на неуспешно декодирање, се разгледуваат следните два случаи:

2.1. При појавување на *грешка-празно-множество*, некое од множествата $S_i = \emptyset$. Во тој случај се земаат сите елементи од множеството $S_{i-1}$ и се наоѓа максималниот заеднички префикс на неговите елементи. Нека тој префикс има должина $k$. Потоа овој заеднички префикс се споредува со првите $k$ бита од влезната порака (која е со должина од $m$ бита). Ако тие се разликуваат во $t$ бита, тогаш бројот на погрешно пренесени битови е $m - k + t$.

2.2 Ако се појави *грешка-повеќе-кандидати*, тогаш од елементите на последното множество $S_s$ повторно го наоѓаме максималниот заеднички префикс и постапката за определување на бројот на погрешно пренесени битови се врши на ист начин како и кај грешка-празно-множество.

Со цел да се провери влијанието на патернот врз перформансите на кодот, во мојата магистерска теза, направени се експерименти за неколку различни патерни за додавање на редундантните нулти нибли. Експериментите се извршуваат за различни вредности на $SNR$ во интервал од $-3$ до $10$ децибели. Од направените експерименти ги најдовме патерните со кои се добиваат најдобри резултати (најмали BER и PER) и со помош на тие патерни направени се експерименти во оваа докторска дисертација.

Исто така, со цел да се провери влијанието на клучот врз перформансите на кодот, направени се експерименти со различна должина на клучот. Анализирајќи ги добиените резултати (во мојата магистерска теза) заклучено е дека клучот со должина 10 дава најдобри резултати, па поради тоа таа должина на клуч се зема како параметар во конструкцијата на оптимален RCBQ.

За да се провери дали изборот на квазигрупата има влијание на перформансите на кодот, направени се и експерименти со циклична квазигрупа од ред 16 (за клуч со должина 10). Потоа, направени се експерименти со квазигрупа од ред 16 која е добиена со директен производ на квазигрупа од ред 2. Експериментални резултати за вака добиената квазигрупа се полоши и од резултатите добиени со цикличната квазигрупа. Цикличната квазигрупа и квазигрупата добиена со директен производ на квазигрупа од ред 2 се примери на фрактални квазигрупи. Од друга страна, квазигрупата дадена во Табела 3.2 е пример на нефрактална квазигрупа и резултатите добиени со оваа квазигрупа се сосема задоволителни. Исто така, дадената квазгигрупа е и без облик, што е добро заради криптографски цели. Од претходните примери, може да се заклучи дека изборот на квазигрупата има огромно влијание над перформансите на кодот [76].

## 3.3 Експериментални резултати добиени со АД4П

Направивме многу експерименти за АД4П и ги најдовме оптималните параметри на исти начин како што е објаснето во Поглавје 3.2. Најдобри резултати за кодот $(72, 576)$ со рата $R = 1/8$ се добиени со употреба на следните параметри:

- Во АДП - патерн за редундантност: 1100 1100 1000 0000 1100 1000 1000 0000 1100 1100 1000 0000 1100 1000 1000 0000 0000 0000 за рата $1/4$ и два различни клучеви од 10 нибли.
- Во АД4П - патерн за редундантност: 1100 1110 1100 1100 1110 1100 1100 1100 0000 за рата $1/2$ и четири различни клучеви од 10 нибли.
- Во сите експерименти ја користиме азбуката од нибли $Q = \{0, 1, \ldots, 9, a, b, c, d, e, f\}$ и квазигрупата $(Q, *)$ дадена во Табела 3.2.

Табела 3.2: Квазигрупа со ред 16 употребена во експериментите

| $*$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | $a$ | $b$ | $c$ | $d$ | $e$ | $f$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 3 | $c$ | 2 | 5 | $f$ | 7 | 6 | 1 | 0 | $b$ | $d$ | $e$ | 8 | 4 | 9 | $a$ |
| 1 | 0 | 3 | 9 | $d$ | 8 | 1 | 7 | $b$ | 6 | 5 | 2 | $a$ | $c$ | $f$ | $e$ | 4 |
| 2 | 1 | 0 | $e$ | $c$ | 4 | 5 | $f$ | 9 | $d$ | 3 | 6 | 7 | $a$ | 8 | $b$ | 2 |
| 3 | 6 | $b$ | $f$ | 1 | 9 | 4 | $e$ | $a$ | 3 | 7 | 8 | 0 | 2 | $c$ | $d$ | 5 |
| 4 | 4 | 5 | 0 | 7 | 6 | $b$ | 9 | 3 | $f$ | 2 | $a$ | 8 | $d$ | $e$ | $c$ | 1 |
| 5 | $f$ | $a$ | 1 | 0 | $e$ | 2 | 4 | $c$ | 7 | $d$ | 3 | $b$ | 5 | 9 | 8 | 6 |
| 6 | 2 | $f$ | $a$ | 3 | $c$ | 8 | $d$ | 0 | $b$ | $e$ | 9 | 4 | 6 | 1 | 5 | 7 |
| 7 | $e$ | 9 | $c$ | $a$ | 1 | $d$ | 8 | 6 | 5 | $f$ | $b$ | 2 | 4 | 0 | 7 | 3 |
| 8 | $c$ | 7 | 6 | 2 | $a$ | $f$ | $b$ | 5 | 1 | 0 | 4 | 9 | $e$ | $d$ | 3 | 8 |
| 9 | $b$ | $e$ | 4 | 9 | $d$ | 3 | 1 | $f$ | 8 | $c$ | 5 | 6 | 7 | $a$ | 2 | 0 |
| $a$ | 9 | 4 | $d$ | 8 | 0 | 6 | 5 | 7 | $e$ | 1 | $f$ | 3 | $b$ | 2 | $a$ | $c$ |
| $b$ | 7 | 8 | 5 | $e$ | 2 | $a$ | 3 | 4 | $c$ | 6 | 0 | $d$ | $f$ | $b$ | 1 | 9 |
| $c$ | 5 | 2 | $b$ | 6 | 7 | 9 | 0 | $e$ | $a$ | 8 | $c$ | $f$ | 1 | 3 | 4 | $d$ |
| $d$ | $a$ | 6 | 8 | 4 | 3 | $e$ | $c$ | $d$ | 2 | 9 | 1 | 5 | 0 | 7 | $f$ | $b$ |
| $e$ | $d$ | 1 | 3 | $f$ | $b$ | 0 | 2 | 8 | 4 | $a$ | 7 | $c$ | 9 | 5 | 6 | $e$ |
| $f$ | 8 | $d$ | 7 | $b$ | 5 | $c$ | $a$ | 2 | 9 | 4 | $e$ | 1 | 3 | 6 | 0 | $f$ |

Во сите експерименти со овие алгоритми употребуваме $B_{max} = 5$.

Во Табела 3.3, претставени се експериментални резултати за веројатноста за бит-грешка $BER_{cut}$, $BER_{4-sets}$, добиени со АДП и АД4П, соодветно и соодветните веројатности за пакет-грешка $PER_{cut}$ и $PER_{4-sets}$. За $SNR$

помали од $-1$, користењето на АДП нема смисла бидејќи веројатностите за бит-грешка се поголеми отколку веројатноста за бит-грешка во каналот.

Табела 3.3: Експериментални резултати со $R = 1/8$

| $SNR$ | $BER_{cut}$ | $BER_{4-sets}$ | $PER_{cut}$ | $PER_{4-sets}$ |
|---|---|---|---|---|
| $-2$ | / | 0.06548 | / | 0.11283 |
| $-1$ | 0.05591 | 0.02225 | 0.10491 | 0.03831 |
| 0 | 0.01232 | 0.00449 | 0.02376 | 0.00835 |
| 1 | 0.00224 | 0.00066 | 0.00425 | 0.00136 |
| 2 | 0.00037 | 0.00008 | 0.00058 | 0.00014 |

Анализирајќи ги резултати од Табела 4.3, можеме да заклучиме дека за сите вредности на $SNR$, резултатите за $BER_{4-sets}$ се подобри во однос на соодветните резултати на $BER_{cut}$. Слични заклучоци може да да се добијат за споредбата на $PER_{4-sets}$ и $PER_{cut}$.

## 3.4 Експериментални резултати за слики

Во овој дел, ги презентираме експерименталните резултати добиени при пренос на слики низ Гаусов канал за различни вредности на $SNR$ (signal-to-noise ratio). Ги споредуваме резултатите добиени со користење на претходно спомнати алгоритми за RCBQ. Во сите експерименти (за различни вредности на $SNR$ во каналот) се разгледуваат разликите меѓу пренесената и декодираната слика. Исто така, ги споредуваме експериментално добиените вредности за веројатност за бит грешка ($BER$ ) и веројатност за пакет грешки ($PER$) и времетраењето на процесите на декодирање со двата алгоритми.

Во сите алгоритми за декодирање за RCBQ, кога се појавува *грешка-празно-множество*, процесот на декодирање завршува порано и е декодиран само дел од пораката. Поради тоа во експериментите со слики го употребуваме следното решение. Во случаите, кога се појавува *грешка-празно-множество*, т.е. во некоја итерација, сите редуцирани множества се празни, ги земаме стринговите без редундантни симболи од сите елементи во множествата од претходната итерација и наоѓаме нивен најдолг заеднички префикс. Ако овој подстринг има $k$ симболи тогаш со цел да се добие декодирана порака од $l$

симболи, ги земаме овие $k$ симболи и додаваме $l-k$ нулти симболи на крајот од пораката.

Во ова поглавје ги презентираме и ги споредуваме добиените резултати користејќи ги алгоритмите за RCBQ со модификациите за намалување на бројот на неуспешно декодирање.

Сите експерименти се направени за код $(72, 576)$ со рата $R = 1/8$ со $B_{max} = 5$ користејќи ги параметрите како во Поглавје 3.3.

Во експериментите за пренос на слики ја користиме сликата "Лена" дадена на Слика 3.3а), додека втората Слика 3.3б), е сликата шифрирана со алгоритамот (даден во делот 2.1.2) кој се користи во процесот на кодирање во RCBQ. На Сликите 3.4 – 3.8, презентирани се сликите добиени за $SNR = -3$, $SNR = -2$, $SNR = -1$, $SNR = 0$ and $SNR = 1$, соодветно. На секоја слика, првата слика е добиена после пренос низ канал без користење на код за корекција на грешки, втората слика е добиена со користење на АДП и третата слика со користење на АД4П.

а) б)

Слика 3.3: Оригинална и шифрирана слика

Од сликите, можеме да видиме дека АДП и АД4П корегираат многу грешки кои се појавуваат за време на преносот. Исто така, со АД4П се добиваат помалку оштетувања (линии) на сликите во однос на АДП.

Вредностите на $BER$ и $PER$ добиени со двата алгоритми (презентирани во Табела 3.4 и Табела 3.5) ги потврдуваат нашите заклучоци од сликите. Во табелите со $BER_{cut}$ и $PER_{cut}$ се означуваат веројатностите за АДП и со $BER_{4-sets}$ и $PER_{4-sets}$ соодветните веројатности добиени со АД4П. Од овие табели може да се види дека за сите вредности од $SNR$, $BER_{4-sets}$ е повеќе од 3 пати помала од $BER_{cut}$. Исто така, веројатноста за пакет грешки добиени со АД4П е повеќе од 2 пати помала од веројатноста добиена со АДП.

Слика 3.4: $SNR = -3$

Слика 3.5: $SNR = -2$

Слика 3.6: $SNR = -1$

Како што објаснивме порано, кога ќе се појави *грешка-празно-множество* на крајот на пораката додаваме $l - k$ нулти симболи и со тоа на сликата се добиваат хоризонтални црни линии. Хоризонталните бели и сиви линии се добиваат во случај на *грешка-повеќе-кандидати* кога случајно избраната порака од редуцираните множества во последната итерација се разликува од оригиналната порака или во случај на недетектирана грешка.

Слика 3.7: $SNR = 0$

Слика 3.8: $SNR = 1$

Табела 3.4: Експериментални резултати за $BER$

| $SNR$ | $BER_{cut}$ | $BER_{4-sets}$ |
|---|---|---|
| $-3$ | 0.31351 | 0.10411 |
| $-2$ | 0.13257 | 0.03487 |
| $-1$ | 0.04029 | 0.01233 |
| 0 | 0.00990 | 0.00306 |
| 1 | 0.00161 | 0.00040 |

Од друга страна, добиените слики без употреба на кодови за корекција на грешки ги немаат овие линии, но во целата слики има точки кои ги означуваат погрешно пренесените симболи.

Во Табела 3.6 и Табела 3.7, се дадени веројатностите за *грешка-празно-множество* $PER_{null}$) и *грешка-повеќе-кандидати* ($PER_{more-candidate}$). Од овие табели можеме да заклучиме дека со АДП добиваме многу поголем број на

Табела 3.5: Експериментални резултати за $PER$

| $SNR$ | $PER_{cut}$ | $PER_{4-sets}$ |
|---|---|---|
| $-3$ | 0.52898 | 0.24045 |
| $-2$ | 0.24265 | 0.08019 |
| $-1$ | 0.07429 | 0.02622 |
| 0 | 0.01675 | 0.00645 |
| 1 | 0.00316 | 0.00082 |

неуспешни декодирања со *грешка-празно-множество* отколку со АД4П, но бројот на *грешка-повеќе-кандидати* е помал. Затоа, сликите декодирани со АДП имаат повеќе црни линии од сликите што се декодираат со АД4П. Исто така, повеќето од линиите во третите слики (добиени со АД4П) се бели или сиви.

Табела 3.6: Експериментални резултати со АДП

| $SNR$ | $PER_{null}$ | $PER_{more-candidate}$ |
|---|---|---|
| $-3$ | 0.52719 | 0.00096 |
| $-2$ | 0.24155 | 0.00041 |
| $-1$ | 0.07402 | 0.00027 |
| 0 | 0.01675 | 0 |
| 1 | 0.00316 | 0 |

Табела 3.7: Експериментални резултати со АД4П

| $SNR$ | $PER_{null}$ | $PER_{more-candidate}$ |
|---|---|---|
| $-3$ | 0.07278 | 0.10588 |
| $-2$ | 0.02691 | 0.03131 |
| $-1$ | 0.01181 | 0.00796 |
| 0 | 0.00247 | 0.00288 |
| 1 | 0.00041 | 0.00027 |

Исто така, ја анализиравме и брзината на двата алгоритми и заклучивме

дека АД4П е побрз од АДП. Брзината на алгоритмите зависи од вредноста на $SNR$ и се зголемува со зголемување на $SNR$. На пример, за $SNR = 1$, АД4П е два пати побрз од АДП, а за $SNR = 2$, тој е 16 пати побрз. Резултатите од овој дел се објавени во [1].

Со цел да се исчистат некои оштетувања (хоризонтални линии) на сликите, во следниот дел предлагаме филтрер кој визуелно ги подобрува пикслите оштетени од *грешка-празно-множество* и *грешка-повеќе-кандидати*.

### 3.4.1 Филтер за слики декодирани со криптокодови базирани на квазигрупи

Во сите експерименти, на добиените слики беа воочливи оштетувања (во облик на линии) кои се резултат на некорегираните грешки при пренос. Со цел да се поправат овие оштетувања и да се добијат појасни слики, во овој дел предлагаме филтер за нивна корекција. Но, за да се поправи оштетувањето, филтерот мора да лоцира каде се наоѓа оштетувањето. Лоцирањето на *грешка-празно-множество* е лесно бидејќи додаваме нулти симболи на местото на недекодираниот дел од пораката. Со цел да се лоцираат *грешките-повеќе-кандидати*, во експериментите со предложениот филтер го менуваме правилото за декодирање за овој вид на грешки. Наместо случајно селектирање на порака од редуцираните множества во последната итерација, земаме порака со сите нулти симболи како декодирана порака. Сега, еден пиксел се смета за оштетен ако припаѓа на нулти подблок со најмалку четири последователни нулти нибли. Основната идеја во дефиницијата на овој филтер е да се замени вредноста на интензитетот на оштетениот пиксел со нова вредност добиена од фиксен број околни пиксели. Во овој процес, ја употребуваме медијаната на ненултите вредности на околните пиксли, така што нашиот филтер е филтер со медијана.

За секој оштетен пиксел на позиција $(i, j)$, во филтерот го употребуваме следниот алгоритам:

1. се зема $3 \times 3$ регион центриран околу пикселот $(i, j)$;

2. се сортираат ненултите вредности на пикселите во регионот во растечки редослед;

3. медијаната на овие вредности се зема како нова вредност на пикселот $(i, j)$.

На Слика 3.9 – Слика 3.12, претставени се сликите добиени со АДП и АД4П пред и после примената на предложениот филтер за $SNR = -2$, $SNR = -1$, $SNR = 0$ и $SNR = 1$, соодветно. На секоја слика, првите две слики се добиени со АДП (без и со филтерот, соодветно), а последните две со АД4П (со и без филтерот).

Од презентираните слики може да забележиме дека со предоженият филтер се добива големо подобрување на сликите за сите разгледани вредности на $SNR$. Исто така, филтерот дава подобри резултати за сликите добиени со АДП отколку со АД4П. Причината за ова е поголемиот број на *недетектирани грешки* добиени при декодирање со АД4П. Резултатите од овој дел се објавени во [77].

Слика 3.9: $SNR = -2$

Слика 3.10: $SNR = -1$

Слика 3.11: $SNR = 0$

Слика 3.12: $SNR = 1$

# Глава 4

# Брзи алгоритми за декодирање базирани на квазигрупи

Декодирањето со АДП и АД4П е всушност декодирање со листа. Брзината на декодирањето зависи од големината на листата (кратки листи даваат побрзо декодирање). Големината на листата зависи од $B_{max}$ (максималниот претпоставен број на бит грешки во еден блок). За помали вредности на $B_{max}$, добиваме пократки листи. Но, не знаеме однапред колку грешки ќе се појават за време на преносот на еден блок. Ако бројот на грешки во еден блок е поголем од $B_{max}$, грешките нема да бидат корегирани. Од друга страна, ако $B_{max}$ е многу големо, имаме долги листи и процесот на декодирање е многу бавен. Исто така, големи вредности на $B_{max}$ доведуваат до завршување на процесот на декодирање со *грешка-повеќе-кандидати*. Затоа, со сите алгоритми за декодирање за RCBQ, *грешка-повеќе-кандидати* може да се добие и кога веројатноста за бит грешка во каналот е многу мала и бројот на бит грешки во блокот не е поголема од $B_{max}$ (или нема грешки за време на преносот).

Со цел да се реши овој проблем, ги предлагаме следните модификации на АДП и АД4П, наречени Брз-алгоритам-за-декодирање-со-пресеци (Fast-Cut-Decoding) или кратко БрзАДП и Брз-алгоритам-за декодирање-со-4-пресеци (Fast-4-Sets-Cut-Decoding algorithm) или кратко БрзАД4П [86].

Овде, наместо фиксна вредност на $B_{max}$ како во двата претходни алгоритми, започнуваме со $B_{max} = 1$. Ако имаме успешно декодирање, постапката е завршена. Ако не, ја зголемуваме вредноста на $B_{max}$ за 1 и го повторуваме процесот на декодирање со новата вредност на $B_{max}$, итн. Процесот на декодирање завршува со $B_{max} = 4$ (за рата $1/4$) или со $B_{max} = 5$ (за рата $1/8$).

Всушност, во новите алгоритми се обидуваме да ја декодираме пораката

користејќи пократки листи и во случај на успешно декодирање за помали вредности на $B_{max}$ ($B_{max} < 4$), избегнуваме долги листи и побавно декодирање. Исто така, го намалуваме и бројот на *грешка-повеќе-кандидати*.

## 4.1 Експериментални резултати

Во овој дел, ќе ги презентираме експерименталните резултати за новите брзи алгоритми и ќе ги споредиме со веќе постоечките.

## 4.2 Експериментални резултати за БрзАДП и БрзАД4П

Во овој дел дадени се експериментални резултати добиени со новите БрзАДП и БрзАД4П за рата $R = 1/4$ и $R = 1/8$, за различни вредности на $SNR$ во Гаусов канал [86]. Овие резултати ги споредуваме со соодветните резултати добиени со АДП и АД4П. Со цел да ја покажеме ефикасноста на новите алгоритми го претставуваме и процентот на пораки за кои декодирањето завршило со $B_{max} = 1, 2, 3, 4$ или $5$.

Прво, ќе ги претставиме резултатите за код $(72, 288)$ со рата $R = 1/4$. За овој код, направени се експерименти со АДП и БрзАДП користејќи ги следните параметри:

– Патерн за редундантност 1100 1100 1000 0000 1100 1000 1000 0000 1100 1100 1000 0000 1100 1000 1000 0000 0000 0000 за рата $1/2$

– Два различни клучеви од 10 нибли и

– Квазигрупата дадена во Табела 3.2 и соодветната парастрофа.

Во експериментите со АДП употребивме $B_{max} = 4$ и за БрзАДП, максимална вредност за $B_{max}$ е 4. Во Табела 4.1, се претставени експериментални резултати за веројатноста за бит-грешка $BER_{cut}$, а веројатноста за пакет-грешка $PER_{cut}$ (добиени со АДП) и соодветните веројатности $BER_{f-cut}$ и $PER_{f-cut}$ (добиени со БрзАДП). Резултатите за $BER_{cut}$ и $PER_{cut}$ за АДП се дадени во Глава 3. За вредности на $SNR$ помали од 0, кодирањето нема смисла бидејќи веројатноста за бит-грешка добиена со АДП е поголема од веројатноста за бит-грешка во каналот (без кодирање) и поради тоа резултатите за помалите вредности од 0 не се претставени во табелите.

Табела 4.1: Експериментални резултати за $R = 1/4$

| $SNR$ | $BER_{cut}$ | $BER_{f-cut}$ | $PER_{cut}$ | $PER_{f-cut}$ |
|---|---|---|---|---|
| 0 | 0.07153 | 0.06122 | 0.10001 | 0.08993 |
| 1 | 0.01830 | 0.01493 | 0.02722 | 0.02275 |
| 2 | 0.00249 | 0.00155 | 0.00410 | 0.00274 |
| 3 | 0.00073 | 0.00006 | 0.00230 | 0.00014 |
| 4 | 0.00052 | 0.00001 | 0.00252 | 0.00007 |

Анализирајќи ги резултатите во Табела 4.1, можеме да заклучиме дека за сите вредности на $SNR$, резултатите за $BER_{f-cut}$ се подобри отколку соодветните резултати за $BER_{cut}$. За $SNR = 4$, $BER_{f-cut}$ е дури 50 пати помала од $BER_{cut}$. Исти заклучоци се добиваат и со споредување на $PER_{f-cut}$ и $PER_{cut}$. Во Табела 4.2, го претставуваме процентот на пораките чие декодирање со БрзАДП завршува со $B_{max} = 1$, $B_{max} = 2$, $B_{max} = 3$ или $B_{max} = 4$. Од резултатите дадени овде, може да се заклучи дека за помали вредности на $SNR$ (0 или 1), за декодирање на голем процент на пораки потребно е $B_{max} = 3$ или 4. Од друга страна, за $SNR = 4$, декодирањето на повеќе од 90% на пораките успешно завршува со $B_{max} = 1$. Од овие резултати, може да се заклучи дека за поголеми вредности на $SNR$ (слаб шум во каналот) декодирањето со новиот предложени алгоритам е многу побрзо отколку со стариот алгоритам.

Табела 4.2: Процент на пораки декодирани со различни вредности на $B_{max}$

| $SNR$ | $B_{max} = 1$ | $B_{max} = 2$ | $B_{max} = 3$ | $B_{max} = 4$ |
|---|---|---|---|---|
| 0 | 2.74% | 26.35% | 37.28% | 33.63% |
| 1 | 14.66% | 45.83% | 28.98% | 10.53% |
| 2 | 44.62% | 41.90% | 11.41% | 2.07% |
| 3 | 76.45% | 20.50% | 2.79% | 0.26% |
| 4 | 93.68% | 5.89% | 0.42% | 0.01% |

Понатаму, претставени се резултатите за код $(72, 576)$ со рата $R = 1/8$. Ги споредуваме експерименталните резултати добиени со АДП, БрзАДП, АД4П

и БрзАД4П. Во експериментите ги употребуваме следните параметри:

- Во АДП и БрзАДП: патерн за редундантност 1100 1100 1000 0000 1100 1000 1000 0000 1100 1100 1000 0000 1100 1000 1000 0000 0000 0000 за рата $1/4$ и два различни клучеви од 10 нибли.

- Во АД4П и БрзАД4П: патерн за редундантност 1100 1110 1100 1100 1110 1100 1100 1100 0000 за рата $1/2$ и четири различни клучеви од 10 нибли.

- Во сите експерименти употребивме иста квазигрупа на $Q$ дадена во Табела 3.2.

Овде, во експериментите со старите алгоритми употребуваме $B_{max} = 5$, а за новите алгоритми, максималната вредност на $B_{max}$ е 5.

Во Табела 4.3, претставени се експериментални резултати за веројатноста за бит-грешка $BER_{cut}$, $BER_{f-cut}$, $BER_{4-sets}$, $BER_{f-4sets}$ добиени со АДП, БрзАДП, АД4П и БрзАД4П, соодветно и соодветните веројатности за пакет-грешка $PER_{cut}$, $PER_{f-cut}$, $PER_{4-sets}$, $PER_{f-4sets}$. За $SNR$ помали од $-1$, користењето на АДП нема смисла бидејќи веројатностите за бит-грешка се поголеми отколку веројатноста за бит-грешка во каналот.

Табела 4.3: Експериментални резултати со $R = 1/8$

| $SNR$ | $BER_{cut}$ | $BER_{f-cut}$ | $BER_{4-sets}$ | $BER_{f-4sets}$ |
|---|---|---|---|---|
| $-2$ | / | / | 0.06548 | 0.04905 |
| $-1$ | 0.05591 | 0.03920 | 0.02225 | 0.00795 |
| 0 | 0.01232 | 0.00872 | 0.00449 | 0.00074 |
| 1 | 0.00224 | 0.00069 | 0.00066 | 0.00013 |
| 2 | 0.00037 | 0 | 0.00008 | 0 |
| $SNR$ | $PER_{cut}$ | $PER_{f-cut}$ | $PER_{4sets}$ | $PER_{f-4sets}$ |
| $-2$ | / | / | 0.11283 | 0.09086 |
| $-1$ | 0.10491 | 0.07171 | 0.03831 | 0.01656 |
| 0 | 0.02376 | 0.01598 | 0.00835 | 0.00151 |
| 1 | 0.00425 | 0.00187 | 0.00136 | 0.00021 |
| 2 | 0.00058 | 0 | 0.00014 | 0 |

Во Табела 4.4, даден е процентот на пораки чие декодирање завршува со $B_{max} = 1$, $B_{max} = 2$, $B_{max} = 3$, $B_{max} = 4$ или $B_{max} = 5$ за двата нови алгоритми.

Табела 4.4: Процент на декодирани пораки со различни вредности на $B_{max}$

| | Брз-алгоритам-за-декодирање-со-пресеци | | | | |
|---|---|---|---|---|---|
| $B_{max}$ / $SNR$ | 1 | 2 | 3 | 4 | 5 |
| $-2$ | / | / | / | / | / |
| $-1$ | 0% | 1.57% | 25.32% | 44.46% | 28.64% |
| 0 | 0.07% | 13.80% | 49.38% | 27.88% | 8.86% |
| 1 | 1.71% | 48.12% | 39.16% | 9.43% | 1.58% |
| 2 | 18.84% | 65.41% | 13.88% | 1.68% | 0.18% |
| | Брз-алгоритам-за-декодирање-со-4пресеци | | | | |
| $B_{max}$ / $SNR$ | 1 | 2 | 3 | 4 | 5 |
| $-2$ | 0% | 1.14% | 7.68% | 51.39% | 39.79% |
| $-1$ | 0% | 6.99% | 32.13% | 49.54% | 11.02% |
| 0 | 3.39% | 28.74% | 48.76% | 17.62% | 1.48% |
| 1 | 20.24% | 51.47% | 25.88% | 2.33% | 0.09% |
| 2 | 57.86% | 37.67% | 4.37% | 0.10% | 0.01% |

Од резултатите дадени во Табела 4.3 и Табела 4.4, можеме да извлечеме слични заклучоци за рата $1/8$ како и за рата $1/4$. Всушност, можеме да заклучиме дека за сите вредности на $SNR$, резултатите за $BER$ и $PER$ добиени со новите алгоритми се подобри од соодветните резултати добиени со старите верзии на овие алгоритми. Повторно, ова подобрување е позначајно за поголеми вредности на $SNR$, т.е. за послаб шум во каналот. Уште повеќе, за $SNR \geqslant 2$, вредностите за $BER$ и $PER$ добиени со новите алгоритми се еднакви на 0.

Од Табелата 4.4, можеме да видиме дека со новите алгоритми, ако вредностите на $SNR$ се помали тогаш имаме поголем процент на пораки за чие декодирање е потребно $B_{max} = 4$ или 5. Од друга страна, за $SNR = 2$, декодирањето на повеќе од 80% од пораките успешно завршуваат со $B_{max} = 1$ или $B_{max} = 2$. Оттука можеме да заклучиме дека за поголеми вредности на

$SNR$ (слаб шум во каналот), декодирањето со новопредложени алгоритми е многу побрзо отколку со старите алгоритми. Резултатите од овој дел се објавени во [86].

## 4.3 Експериментални резултати за слики

Во овој дел, ги истражуваме перформансите на брзите алгоритми за пренос на слики. За оваа цел, направивме многу експерименти и во нашите експерименти ги користиме Гаусов канал и Случајните кодови базирани на квазигрупи. Ги споредуваме добиените резултати користејќи четири алгоритми за случајните кодови базирани на квазигрупи: АДП, БрзАДП, АД4П и БрзАД4П, со модификации за намалување на бројот на неуспешни декодирања. Сите експерименти се направени за код $(72, 576)$ со рата $R = 1/8$ и различни вредности за $SNR$ во Гаусов канал.

Во експериментите ги користиме истите параметри (дадени во претходното поглавје) како и за обичните пораки.

Овде, во експериментите со старите алгоритми користиме $B_{max} = 5$, а во БрзАДП и БрзАД4П максималната вредност на $B_{max}$ е 5. Во случај на *грешка-празно-множество* или *грешка-повеќе-кандидати*, ја применуваме истата хевристика како во Поглавје 3.3.

Исто како и претходно, во експериментите за пренесување на слика, ја користиме сликата "Лена", дадена на Слика 3.3 а). Сликата се пренесува преку каналот (за различни вредности на $SNR$) и се применува соодветниот алгоритам за декодирање. На Слика 4.1 – Слика 4.4, ги претставуваме сликите добиени за $SNR = -2$, $SNR = -1$, $SNR = 0$ и $SNR = 1$, соодветно. Во секоја од нив, првата слика се добива со употреба на АДП, втората слика се добива со БрзАДП, третата слика со АД4П и четвртата со употреба на БрзАД4П.

Ако ги споредиме сликите, можеме да заклучиме дека сликите добиени со БрзАДП и БрзАД4П се појасни споредено со соодветните слики добиени со АДП и АД4П. Исто така, сликите добиени со БрзАД4П се појасни споредено со соодветните слики добиени со БрзАДП, за сите вредности на $SNR$.

Во Табела 4.5, претставени се резултатите за веројатноста за бит-грешка $BER_{cut}$, $BER_{f-cut}$, $BER_{4-sets}$, $BER_{f-4sets}$ добиени со АДП, БрзАДП, АД4П и БрзАД4П, соодветно, и соодветните веројатности за пакет-грешка $PER_{cut}$, $PER_{f-cut}$, $PER_{4-sets}$, $PER_{f-4sets}$.

Слика 4.1: $SNR = -2$

Слика 4.2: $SNR = -1$

Слика 4.3: $SNR = 0$

Во Табелата 4.6, претставен е процентот на пораките за кои декодирањето завршува со $B_{max} = 1$, $B_{max} = 2$, $B_{max} = 3$, $B_{max} = 4$ или $B_{max} = 5$ со БрзАДП. Анализирајќи ги резултатите може да заклучиме дака добиените резултати се слични како и за обичните пораки. Од резултатите дадени овде, можеме да видиме дека за поголеми вредности на $SNR$ (1 или 2), имаме поголем процент на пораки за чие декодирањето потребно е $B_{max} = 2$. Ова значи дека за поголеми вредности на $SNR$ (слаб шум во каналот) декодирањето со новиот предложен алгоритам е побрз споредено со стариот АДП.

Слика 4.4: $SNR = 1$

Табела 4.5: Експериментални резултати за $R = 1/8$

| $SNR$ | $BER_{cut}$ | $BER_{f-cut}$ | $BER_{4-sets}$ | $BER_{f-4sets}$ |
|---|---|---|---|---|
| $-2$ | 0.13257 | 0.14189 | 0.03487 | 0.04886 |
| $-1$ | 0.04029 | 0.03947 | 0.01233 | 0.00823 |
| 0 | 0.00990 | 0.00594 | 0.00306 | 0.00062 |
| 1 | 0.00161 | 0.00140 | 0.00040 | 0 |
| $SNR$ | $PER_{cut}$ | $PER_{f-cut}$ | $PER_{4-sets}$ | $PER_{f-4sets}$ |
| $-2$ | 0.24265 | 0.24952 | 0.08019 | 0.09063 |
| $-1$ | 0.07429 | 0.07127 | 0.02622 | 0.01634 |
| $-0$ | 0.01675 | 0.01332 | 0.00645 | 0.00151 |
| 1 | 0.00316 | 0.00307 | 0.00082 | 0 |

Табела 4.6: Процент на пораки декодирани со БрзАДП за различни вредност на $B_{max}$

| $SNR$ | $B_{max} = 1$ | $B_{max} = 2$ | $B_{max} = 3$ | $B_{max} = 4$ | $B_{max} = 5$ |
|---|---|---|---|---|---|
| $-2$ | 0% | 0.12% | 6.04% | 33.92% | 59.91% |
| $-1$ | 0% | 1.61% | 25.89% | 43.27% | 29.24% |
| 0 | 0.08% | 14.16% | 48.15% | 28.93% | 8.68% |
| 1 | 1.99% | 47.68% | 39.29% | 9.37% | 1.68% |
| 2 | 18.76% | 66.03% | 13.43% | 1.68% | 0.07% |

Во Табелата 4.7, претставен е процентот на пораките за кои декодирањето завршува со $B_{max} = 1$, $B_{max} = 2$, $B_{max} = 3$, $B_{max} = 4$ и $B_{max} = 5$ со БрзАД4П. Од добиените резултати, можеме да видиме дека за поголеми вредности на $SNR$ (0 или 1), имаме повисок процент на пораки за чие декодирање е потребно $B_{max} = 1$ или 2. Може да заклучиме дека за поголеми вредности на $SNR$ (слаб шум во каналот) декодирањето со новиот предложен алгоритам е многу побрзо споредено со стариот АД4П.

Табела 4.7: Процент на пораки декодирани со БрзАД4П за различни вредности на $B_{max}$

| $SNR$ | $B_{max} = 1$ | $B_{max} = 2$ | $B_{max} = 3$ | $B_{max} = 4$ | $B_{max} = 5$ |
|---|---|---|---|---|---|
| $-2$ | 0.03% | 1.15% | 7.84% | 51.11% | 39.87% |
| $-1$ | 0.36% | 7.28% | 32.74% | 48.81% | 10.82% |
| 0 | 3.53% | 28.00% | 49.77% | 17.19% | 1.51% |
| 1 | 20.54% | 51.37% | 25.84% | 2.14% | 0.10% |
| 2 | 57.79% | 37.70% | 4.49% | 0.03% | 0% |

Со овие нови алгоритми, БрзАДП и БрзАД4П, добиваме подобри резултати за веројатноста на пакет-грешка и бит-грешка споредено со претходно дефинираните (АДП и АД4П) алгоритми на RCBQ за пренос низ Гаусов канал. Исто така, од презентираните проценти на пораките чие декодирање завршува за различни вредности на $B_{max}$, можеме да заклучиме дека за поголеми вредности на $SNR$, БрзАДП и БрзАД4П овозможуваат побрзо декодирање.

## 4.4 Експериментални резултати за пренос на аудио датотеки

Во ова поглавје, ги истражуваме перформансите на брзите алгоритми при пренос на аудио датотеки. Во нашите експерименти го користиме Гаусовиот канал и $RCBQ$ како код за корекција на грешки. Слично, како и за сликите, ги споредуваме резултатите за код $(72, 576)$ со рата $R = 1/8$, добиени со користење на четирите алгоритми за $RCBQ$: АДП, БрзАДП, АД4П и БрзАД4П и за различни вредности на $SNR$ во Гаусов канал. Во овие експерименти ја користиме аудио датотеката кој се состои од еден 16-битен канал со рата на

семплирање од $44100Hz$ и е дел од Бетовеновата "Ода на радоста" во траење од $4.3$ секунди.

Во експериментите ги користиме следните параметри:

- Во АДП и во БрзАДП - патерн на редундантност: 1100 1100 1000 0000 1100 1000 1000 0000 1100 1100 1000 0000 1100 1000 1000 0000 0000 0000, за рата $1/4$ и два различни клучеви од 10 нибли.
- Во АД4П и во БрзАД4П - патерн на редундантност: 1100 1110 1100 1100 1110 1100 1100 1100 0000 за рата $1/2$ и четири различни клучеви од 10 нибли.
- Во сите експерименти ја употребуваме истата квазигрупа на $Q$ дадена во Табела 3.2.

Во експериментите со АДП и АД4П употребуваме $B_{max} = 5$, а во експериментите со БрзАДП и БрзАД4П максималната вредност на $B_{max}$ е 5.

Во сите алгоритми за декодирање на RCBQ, кога се појавува *грешка-празно-множество*, процесот на декодирање завршува порано и само дел од пораката е декодиран. Поради тоа во експериментите со аудио датотеките го користиме следното решение. Во случаевите кога се појавува *грешка-празно-множество*, т.е. сите редуцирани множества се празни во некоја итерација, ги земаме стринговите без редундантни симболи од сите елементи во множествата од претходната итерација и го наоѓаме нивниот максимален заеднички префикс. Ако овој префикс има $k$ симболи тогаш со цел да се добие декодираната порака со $l$ симболи, ги земаме овие $k$ симболи и додаваме $l - k$ нулти симболи на крајот на пораката.

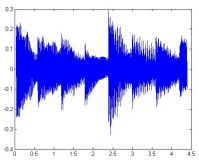
Во Табела 4.8, ги претставуваме експерименталните резултати за веројатноста за бит грешка $BER_{cut}$, $BER_{f-cut}$, $BER_{4sets}$, $BER_{f-4sets}$ добиена со АДП, БрзАДП, АД4П и БрзАД4П, соодветно и соодветните веројатности за пакет грешка $PER_{cut}$, $PER_{f-cut}$, $PER_{4sets}$, $PER_{f-4sets}$. За $SNR$ помало од $-1$, користењето на АДП нема смисла бидејќи веројатностите за бит грешка се поголеми од веројатноста за бит грешка во каналот.

Од експерименталните резултати дадени во Табела 4.8 можеме да заклучиме дека за сите вредности на $SNR$, резултатите за $PER$ и $BER$ добиени со брзите алгоритми се подобри од резултатите добиени со АДП и АД4П, особено за поголеми вредности на $SNR$ (слаб шум). Освен за $SNR = -2$ , за сите други $SNR$, БрзАД4П дава најдобри резултати. Исто така, декодирањето со овие алгоритми е многу побрзо во споредба со старите алгоритми.

Табела 4.8: Експериментални резултати за $BER$ и $PER$

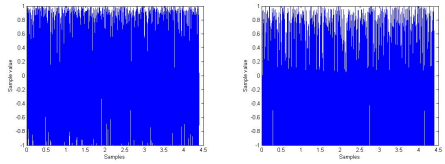
| $SNR$ | $BER_{cut}$ | $BER_{f-cut}$ | $BER_{4sets}$ | $BER_{f-4sets}$ |
|---|---|---|---|---|
| $-2$ | / | / | 0.03658 | 0.04782 |
| $-1$ | 0.04741 | 0.04019 | 0.01122 | 0.00825 |
| 0 | 0.01114 | 0.00713 | 0.00281 | 0.00081 |
| 1 | 0.00175 | 0.00086 | 0.00052 | 0.00003 |
| $SNR$ | $PER_{cut}$ | $PER_{f-cut}$ | $PER_{4sets}$ | $PER_{f-4sets}$ |
| $-2$ | / | / | 0.08451 | 0.08917 |
| $-1$ | 0.08623 | 0.07589 | 0.02499 | 0.01644 |
| 0 | 0.02208 | 0.01418 | 0.00632 | 0.00165 |
| 1 | 0.00383 | 0.00177 | 0.00113 | 0.00012 |

За сите експерименти, ги разгледуваме разликите и помеѓу вредностите на примероците (семпловите) на оригиналниот и декодираниот сигнал. Ги претставуваме овие резултати на графици каде што бројот на примероци во низата примероци што го сочинуваат аудио сигналот е на $x$ - оската, а вредноста на примерокот е на $y$ -оската. На Слика 4.5 претставени се оригиналните аудио примероци. Графиците за декодираните аудио датотеки за разгледуваните вредности на $SNR$ се дадени на Слика 4.6 - Слика 4.9. На Слика 4.6 (за $SNR = -2$) претставени се само 2 графици, првиот за АД4П и вториот за БрзАД4П. За другите вредности на $SNR$, на Слика 4.7 - Слика 4.9, претставени се 4 графици (за декодираните аудио датотеки употребувајќи ги сите 4 спомнати алгоритми) во следниот редослед: АДП, БрзАДП, АД4П и БрзАД4П.

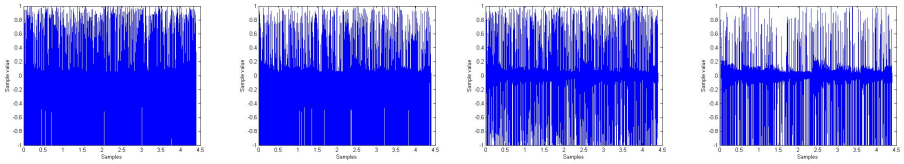


Слика 4.5: Оригинални аудио примероци

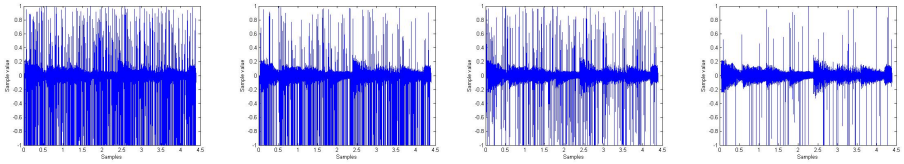
Од овие слики како и од табелите за $PER$ и $BER$ можеме да извлечеме некои заклучоци. Од графиците очигледно е дака за сите вредности на $SNR$,



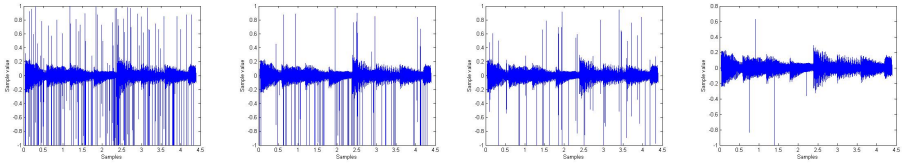
Слика 4.6: Резултати за $SNR = -2$



Слика 4.7: Резултати за $SNR = -1$



Слика 4.8: Резултати за $SNR = 0$



Слика 4.9: Резултати за $SNR = 1$

резултатите добиени со користење на АД4П се подобри од резултатите добиени со АДП и резултатите добиени со новите брзи алгоритми се подобри отколку соодветните резултати добиени со старите верзии на овие алгоритми.

Сите аудио датотеки добиени во нашите експерименти за пренос низ Гаусов канал со различни $SNR$ може да се најдат на следниот линк:
`https://www.dropbox.com/sh/5zq5ly6qtiho8d6/AACTQBgUDopFq9psdbaMb8BKa?`

`dl=0`.

Ако некој ги слуша овие аудио датотеки, тој/таа ќе забележи дека како што се намалува $SNR$, се зголемува шумот, но оригиналната мелодија може да се слушне целосно во позадина. Ова е видливо и на графиците. Ако ги споредиме графиците на декодираните датотеки, добиени со употреба на сите 4 алгоритми, со графикот за оригиналните аудио примероци, можеме да видиме дека примероците од оригиналната аудио датотека се содржани во сите графици. Ова е причината зошто сè уште можеме да ја слушнеме оригиналната мелодија во позадината испреплетена со бучните шумови. За да се исчистат некои од овие шумови, во следниот дел предлагаме филтер што може да поправи дел од штетите направени од *грешка-празно-множество* и *грешка-повеќе-кандидати*.

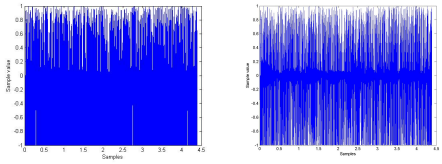
### 4.4.1 Филтер за подобрување на квалитетот на аудио датотеките декодирани со RCBQ

За поправка на оштетувањето настанато при пренос на аудио датотеките, филтерот треба да лоцира каде се појавува оштетувањето. Постапката е иста како и за сликите. Лоцирањето на *грешка-празно-множество* е лесно бидејќи додаваме нулти симболи на местото на некодираниот дел од пораката. За да лоцираме *грешка-повеќе-кандидати* го менуваме правилото за декодирање за овој вид на грешки. Во овој случај, место случајно да избереме порака од редуцираните множества во последната итерација, земаме порака со сите нулти симболи како декодирана порака.

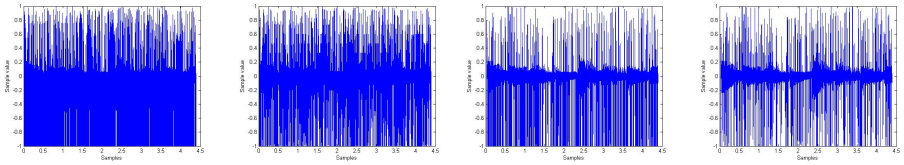
Основната идеја во дефинирањето на овој филтер е да ги замени оштетените (погрешно декодирани) нибли со нова вредност добиена од вредностите на неколку претходни нибли. Значи, ги земаме сите декодирани пораки како една листа од нибли и една нибла се смета за погрешно декодиран симбол, ако припаѓа на нулта под-листа со најмалку четири последователни нулти нибли. Потоа, секој погрешно декодиран симбол (нибла) го заменуваме со медијаната на претходните $2k + 1$ нибли во листата. Ако погрешната нибла е на почетокот на листата и нема претходни $2k + 1$ нибли, тогаш се зема медијана од сите претходни нибли (до почетокот на листата). За корегирање на ниблата, ги користиме само претходните $2k + 1$ нибли, бидејќи следните нибли се нули (погрешно декодираниот симбол припаѓа на нулта под-листа со најмалку четири последователни нулта нибли) и веројатно тие се погрешно декодирани. Направивме експерименти за $2k + 1$ еднакво на $3$, $5$, $7$ и $9$ и

резултатите беа слични, но малку подобри резултати се добиваат за $2k + 1$ еднакво на 7 или 9. Овде ги презентираме резултатите добиени со медијана од 7 претходни нибли. Да забележиме дека земаме непарен број претходни нибли бидејќи се пресметува медијаната на овие нибли и ако овој број е парен тогаш медијаната може да биде број кој не е во $Q$.

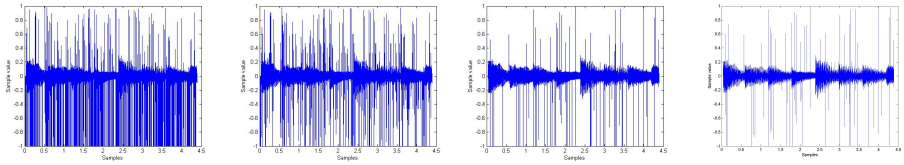
На Сликите 4.10 - 4.13, претставени се графиците на примероците (семплови) на аудио датотеките добиени со БрзАДП и БрзАД4П пред и после употребата на предложениот филтер за $SNR = -2$, $SNR = -1$, $SNR = 0$ и $SNR = 1$, соодветно. На секоја слика, првите два графици се однесуваат на датотеките добиени со БрзАДП (без и со користење на филтерот, соодветно) и последните две слики се за датотеките добиени со БрзАД4П (без и со користење на филтерот, соодветно). Исто така, аудио датотеките добиени после примената на предложениот филтер можат да се најдат на следниот линк: `https://www.dropbox.com/sh/5zq5ly6qtiho8d6/AACTQBgUDopFq9psdbaMb8BKa?dl=0`.



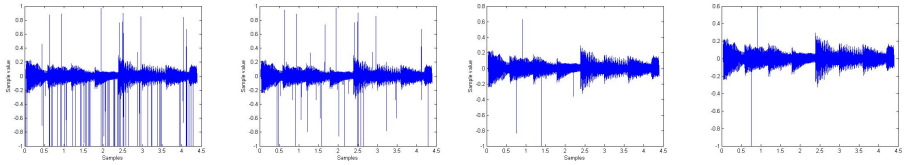
Слика 4.10: $SNR = -2$



Слика 4.11: $SNR = -1$



Слика 4.12: $SNR = 0$



Слика 4.13: $SNR = 1$

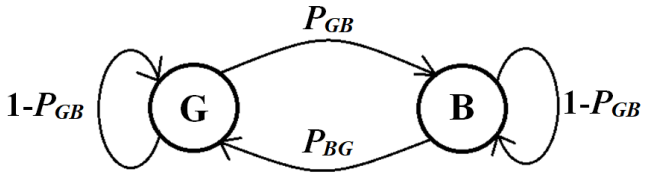
Од дадените графици и дадените аудио датотеки можеме да забележиме дека предложениот филтер дава големо подобрување на звукот за разгледаните вредности на $SNR$. Резултатите од овој дел се објавени во [78].

# Глава 5

# Гилберт-Елиот канал

## 5.1 Гилберт-Елиот модел за рафални грешки

Гилберт-Елиот моделот за рафални грешки е модел на канал воведен од Едгар Гилберт (Edgar Gilbert) и Е.О.Елиот (E. O. Elliott). Овој модел е базиран на верига на Марков со две состојби $G$(добра) и $B$(лоша). Кога каналот е во добра состојба, веројатноста за погрешен пренос на еден бит е мала, а кога е во лоша состојба, оваа веројатност е голема. Овој модел е широко употребуван за опишување на рафални (burst) грешки во каналите за пренос, кој овозможува симулации на дигиталните грешки во комуникациските врски. Овој модел е прикажан на Слика 5.1, каде што $G$ ја претставува добрата состојба, а $B$ лошата состојба на каналот. Веројатноста за премин од добра во лоша состојба е $P_{GB}$ и веројатноста за премин од лоша во добра состојба е $P_{BG}$ ([57], [61]).




Слика 5.1: Гилберт-Елиот модел за рафални грешки

Експериментите во оваа докторска дисертација се направени со два вида

на Гилберт-Елиот канали. Првиот вид на Гилберт-Елиот канал е канал кој е бинарно симетричен во секоја состојба (БСК), со веројатност за бит грешка $P_e(G)$ кога каналот е во добра состојба и $P_e(B)$ кога каналот е во лоша состојба. Кај вториот вид на Гилберт-Елиот канал, каналите се Гаусови, каде вредноста на $SNR_G$, кога каналот е во добра состојба, е голема, а кога каналот е во лоша состојба, вредноста на $SNR_B$ е мала.

## 5.2 Нови криптокодови за каналите со рафални грешки

Во експериментите со каналите со рафални грешки, не добиваме добри резултати со претходно објаснетите алгоритми и поради тоа, во [79] предлагаме нови алгоритми за кодирање/декодирање наречени РафаленАДП (Burst-Cut-Decoding) и РафаленАД4П (Burst-4-Sets-Cut-Decoding). Познато е дека интерливерот и деинтерливерот се користат за справување со рафални грешки во комуникациски систем. Поради тоа, во новите алгоритми се воведува интерливер во алгоритамот за кодирање и соодветен деинтерливер во алгоритамот за декодирање.

Процесот на кодирање во новите алгорими се состои од следниве чекори:

– Влезната порака $M = M_1M_2...M_l$ се проширува со додавање редундантни $\nu$ нулти симболи (на ист начин како во АДП и АД4П) и на тој начин се добива редундантната порака $L = L^{(1)}L^{(2)}...L^{(s/2)} = L_1L_2...L_{m/2}$ од $N/2$ бита (во АДП) т.е. $L = L^{(1)}L^{(2)}...L^{(s/4)} = L_1L_2...L_{m/4}$ од $N/4$ бита (во АД4П), каде што $L^{(i)}$ е подблок од $r$ симболи од азбуката $Q$, и $L_i \in Q$. Притоа, $N = am$, $m = rs$ и $m = l + \nu$.

– Потоа, за кодирање два (четири) пати го применуваме алгоритамот за шифрирање даден на Слика 2.1, на иста редундантна порака $L$, користејќи различни параметри, со што добиваме два (четири) кодни збора.

– На секој коден збор поединечно применуваме интерливер. Интерливерот ги преуредува (по редици) $m$-те нибли од кодниот збор во матрица од ред $k \times (k/m)$. Излезот на интерливерот е измешана порака добиена со читање на оваа матрицата по колони.

– Потоа, ги конкатенираме двата (четирите) излези од интерливерот и на тој начин го добиваме кодниот збор за влезната порака $M$ т.е. $C = C_1C_{\frac{m}{k}+1}C_{\frac{2m}{k}+1}...C_2C_{\frac{m}{k}+2}...C_{\frac{(k-1)m}{k}}C_m$ каде $C_i \in Q$.

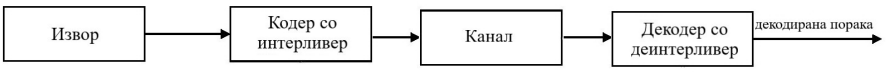
Процесот на декодирање се состои од следниве чекори:

– После преносот на кодираната порака низ каналот со рафални грешки, ја добиваме излезната порака $D = D^{(1)}D^{(2)}...D^{(s)}$, каде што $D^{(i)}$ се подблокови од $r$ симболи. Пораката $D$ се дели на две (четири) пораки со еднаква должина.

– Пред паралелното декодирање, се применува деинтерливерот на секоја порака поединечно. Поточно, секој од деловите на излезната порака се преуредува (по редици) во матрица од ред $(k/m) \times k$, а потоа пораките кои се декодираат се добиваат со читање по колони на добиените матрици.

– Овие измешани пораки се декодираат со соодветниот алгоритам за декодирање на RCBQ (АДП или АД4П).

Процесот на кодирање/декодирање на новите алгоритми шематски е претставен на Слика 5.2.




Слика 5.2: Кодирање/декодирање со новите алгоритми

## 5.3 Експериментални резултати за Гилберт-Елиот канал

Во овој дел ги презентираме експерименталните резултати добиени со RCBQ за пренос низ канал со рафални грешки. За симулација на каналот користиме Гилберт-Елиот модел. Ги споредуваме вредностите на веројатноста за пакет-грешка ($PER$) и веројатноста за бит-грешка ($BER$) добиени со АДП, АД4П, РафаленАДП и РафаленАД4П. Експериментите се направени за код $(72, 288)$ со рата $1/4$ користејќи ги АДП и РафаленАДП и код $(72, 576)$ со рата $1/8$ користејќи ги сите алгоритми (АДП, АД4П, РафаленАДП и РафаленАД4П). Во експериментите се користат следните параметри:

– за кодот $(72, 288)$ во АДП и РафаленАДП, користени се следните параметри:

  – патерн за редундантност: 1100 1110 1100 1100 1110 1100 1100 1100 0000 за код со рата $1/2$ и два различни клучеви од 10 нибли.

– за кодот $(72, 576)$ параметрите што се користат се:

  – во АДП и РафаленАДП- патерн за редундантност 1100 1100 1000 00001100 1000 1000 0000 1100 1100 1000 0000 1100 1000 1000 0000 0000 0000, за код со рата $1/4$ и два различни клучеви од 10 нибли.

  – во АД4П и РафаленАД4П - патерн за редундантност 1100 1110 1100 1100 1110 1100 1100 1100 0000 за рата $1/2$ и четири различни клучеви од 10 нибли.

Во сите експерименти употребуваме иста квазигрупа $Q$ дадена во Табела 3.2.

За новите рафални алгоритми, направивме експерименти за различни вредности на $k$, т.е. за сите делители на 36. Всушност, бројот на нибли во двата (или четирите) конкатенирани кодни зборови во АДП/РафаленАДП (или АД4П/ РафаленАД4П) изнесува 36 (или 72). Најдобри резултати се добиени за $k = 9$ и затоа само тие ќе бидат претставени во оваа дисертација.

Во Поглавје 5.3.1, претставени се добиените експериментални резултати за Гилберт-Елиот моделот со бинарно симетрични канали за различни вредности на веројатноста на бит-грешка и различни веројатности на премин. Експерименталните резултати за различни вредности на $SNR$ и веројатности за премин на Гилберт-Елиот модел со Гаусови канали, се дадени во Поглавје 5.3.2.

### 5.3.1 Експерименти за Гилберт-Елиот модел со БСК канали

Во сите експерименти за Гилберт-Елиот модел со бинарно симетрични канали ја користиме веројатноста за бит-грешка во добра состојба $P_e(G) = 0.01$ и неколку различни вредности на веројатноста за бит-грешка кога каналот е во лоша состојба $P_e(B) \in \{0.2; 0.16; 0.13; 0.1\}$.

Табела 5.1: Експериментални резултати за $R = 1/4$

,

| $P_e(B)$ | $PER_{cut}$ | $PER_{b-cut}$ | $BER_{cut}$ | $BER_{b-cut}$ |
|---|---|---|---|---|
| | $P_{GG} = 0.8$ | | $P_{BB} = 0.8$ | |
| 0.1 | 0.14069 | 0.06818 | 0.10453 | 0.05030 |
| 0.13 | 0.34014 | 0.18173 | 0.26055 | 0.13535 |
| 0.16 | 0.58078 | 0.32466 | 0.45424 | 0.24698 |
| 0.2 | 0.81271 | 0.51087 | 0.67081 | 0.40820 |
| | $P_{GG} = 0.5$ | | $P_{BB} = 0.5$ | |
| 0.1 | 0.13464 | 0.04665 | 0.09799 | 0.03376 |
| 0.13 | 0.32711 | 0.12283 | 0.24417 | 0.08970 |
| 0.16 | 0.56365 | 0.23394 | 0.43088 | 0.17073 |
| 0.2 | 0.82358 | 0.43581 | 0.65848 | 0.33149 |
| | $P_{GG} = 0.2$ | | $P_{BB} = 0.8$ | |
| 0.1 | 0.20470 | 0.14177 | 0.14853 | 0.10266 |
| 0.13 | 0.46781 | 0.34619 | 0.35206 | 0.25621 |
| 0.16 | 0.73048 | 0.59497 | 0.57155 | 0.45788 |
| 0.2 | 0.942468 | 0.84951 | 0.79485 | 0.68579 |
| | $P_{GG} = 0.8$ | | $P_{BB} = 0.2$ | |
| 0.1 | 0.05709 | 0.00921 | 0.04201 | 0.00560 |
| 0.13 | 0.14292 | 0.01555 | 0.10386 | 0.01066 |
| 0.16 | 0.27728 | 0.03333 | 0.20678 | 0.02259 |
| 0.2 | 0.48898 | 0.07250 | 0.37152 | 0.05260 |

Во Табела 5.1, претставени се експерименталните резултати за веројатноста за бит-грешка $BER_{cut}$ и веројатноста за пакет-грешка $PER_{cut}$ (добиени со АДП) и соодветните веројатности $BER_{b-cut}$ и $PER_{b-cut}$ (добиени со РафаленАДП) за код со рата $1/4$ и следните комбинации на веројатностите на премин од добра во добра состојба $P_{GG}$ и од лоша во лоша состојба $P_{BB}$:

- $P_{GG} = 0.8$ и $P_{BB} = 0.8$
- $P_{GG} = 0.5$ и $P_{BB} = 0.5$
- $P_{GG} = 0.2$ и $P_{BB} = 0.8$
- $P_{GG} = 0.8$ и $P_{BB} = 0.2$

Анализирајќи ги резултатите од Табела 5.1, можеме да заклучиме дека за сите вредности на $P_e(B)$ и за сите вредности на $P_{BB}$, резултатите за $BER_{b-cut}$ се подобри од соодветните резултати за $BER_{cut}$. Всушност,

- за $P_{GG} = 0.8$ и $P_{BB} = 0.8$, $BER_{b-cut}$ е од 1.6 до 2.5 пати подобра од $BER_{cut}$;

– за $P_{GG} = 0.5$ и $P_{BB} = 0.5$, $BER_{b-cut}$ е од 1.9 до 2.8 пати подобра од $BER_{cut}$;

– за $P_{GG} = 0.2$ и $P_{BB} = 0.8$, $BER_{b-cut}$ е околу 1.2 пати подобра од $BER_{cut}$;

– за $P_{GG} = 0.8$ и $P_{BB} = 0.2$, $BER_{b-cut}$ е од 7.5 до 10 пати подобра од $BER_{cut}$.

Табела 5.2: Експериментални резултати за $R = 1/8$

| | $P_{GG} = 0.8$ $P_{BB} = 0.8$ | | | |
|---|---|---|---|---|
| $P_e(B)$ | $BER_{cut}$ | $BER_{b-cut}$ | $BER_{4sets}$ | $BER_{b-4sets}$ |
| 0.1 | 0.09183 | 0.04224 | 0.01796 | 0.00601 |
| 0.13 | 0.23069 | 0.11561 | 0.08060 | 0.02616 |
| 0.16 | 0.41820 | 0.22276 | 0.20798 | 0.07405 |
| 0.2 | 0.64522 | 0.39519 | 0.45947 | 0.20217 |
| $P_e(B)$ | $PER_{cut}$ | $PER_{b-cut}$ | $PER_{4sets}$ | $PER_{b-4sets}$ |
| 0.1 | 0.15790 | 0.07445 | 0.03578 | 0.01252 |
| 0.13 | 0.37334 | 0.19549 | 0.16957 | 0.05429 |
| 0.16 | 0.62600 | 0.35476 | 0.41101 | 0.15207 |
| 0.2 | 0.85671 | 0.56624 | 0.75302 | 0.34288 |
| | $P_{GG} = 0.5$ $P_{BB} = 0.5$ | | | |
| $P_e(B)$ | $BER_{cut}$ | $BER_{b-cut}$ | $BER_{4sets}$ | $BER_{b-4sets}$ |
| 0.1 | 0.08336 | 0.02759 | 0.01619 | 0.00412 |
| 0.13 | 0.21774 | 0.07662 | 0.07504 | 0.01353 |
| 0.16 | 0.20663 | 0.15975 | 0.20663 | 0.04268 |
| 0.2 | 0.63793 | 0.31081 | 0.47000 | 0.01298 |
| $P_e(B)$ | $PER_{cut}$ | $PER_{b-cut}$ | $PER_{4sets}$ | $PER_{b-4sets}$ |
| 0.1 | 0.22753 | 0.16028 | 0.07099 | 0.03729 |
| 0.13 | 0.52584 | 0.39026 | 0.29586 | 0.17713 |
| 0.16 | 0.80105 | 0.66229 | 0.66993 | 0.45636 |
| 0.2 | 0.97227 | 0.89840 | 0.95542 | 0.82632 |
| | $P_{GG} = 0.8$ $P_{BB} = 0.2$ | | | |
| $P_e(B)$ | $BER_{cut}$ | $BER_{b-cut}$ | $BER_{4sets}$ | $BER_{b-4sets}$ |
| 0.1 | 0.03391 | 0.00320 | 0.00576 | 0.00044 |
| 0.13 | 0.0926 | 0.00866 | 0.02041 | 0.00124 |
| 0.16 | 0.18494 | 0.02014 | 0.06019 | 0.00256 |
| 0.2 | 0.34766 | 0.04408 | 0.16918 | 0.00803 |
| $P_e(B)$ | $PER_{cut}$ | $PER_{b-cut}$ | $PER_{4sets}$ | $PER_{b-4sets}$ |
| 0.1 | 0.05889 | 0.00583 | 0.01094 | 0.00079 |
| 0.13 | 0.16503 | 0.01684 | 0.04133 | 0.00230 |
| 0.16 | 0.31617 | 0.03578 | 0.12802 | 0.00518 |
| 0.2 | 0.55465 | 0.07913 | 0.34569 | 0.01569 |

Ист заклучок можеме да добиеме со споредување на $PER_{b-cut}$ и $PER_{cut}$. Во Табела 5.2, дадени се експериментални резултати за код со рата $1/8$, за истата комбинација на веројатностите на премин од добра во добра состојба

$P_{GG}$ и од лоша во лоша состојба $P_{BB}$, како и за кодот со рата $1/4$. Со $BER_{cut}$, $BER_{b-cut}$, $BER_{4sets}$ и $BER_{b-4sets}$ се означени веројатностите за бит-грешка добиени со АДП, РафаленАДП, АД4П, РафаленАД4П, соодветно. Исто така, со $PER_{cut}$, $PER_{b-cut}$, $PER_{4sets}$ и $PER_{b-4sets}$ се означени соодветните веројатности за пакет-грешка.

Од резултатите во Табела 5.2, можеме да заклучиме дека за сите вредности на $P_e(B)$ и за сите вредности на $P_{GG}$ и $P_{BB}$, резултатите за $BER_{b-cut}$ се подобри од соодветните резултати за $BER_{cut}$ и резултатите за $BER_{b-4sets}$ се подобри од соодветните резултати добиени за $BER_{4sets}$. Исто така, ако ги споредуваме резултатите за рафалните алгоритмите, можеме да заклучиме дека РафаленАД4П дава од 2 до 7 пати подобри резултати во споредба со РафаленАДП. Истиот заклучок се добива за соодветните веројатности за пакет-грешка.

### 5.3.2 Експерименти за Гилберт-Елиот модел со Гаусови канали

Во овој дел, дадени се експерименталните резултати за Гилберт-Елиот модел со Гаусови канали со $SNR_G = 4$ и различни вредности на $SNR_B \in \{-3, -2, -1\}$ и со истите веројатности на премин како и во експериментите со бинарно симетрични канали. Прво, во Табела 5.3 претставени се експерименталните резултати за код со рата $1/4$, во која ги употребуваме истите ознаки како претходно.

Табела 5.3: Експериментални резултати за $R = 1/4$

| $P_e(B)$ | $PER_{cut}$ | $PER_{b-cut}$ | $BER_{cut}$ | $BER_{b-cut}$ |
|---|---|---|---|---|
| | $P_{GG} = 0.8$, | | $P_{BB} = 0.8$ | |
| -3 | 0.56322 | 0.32366 | 0.44058 | 0.24605 |
| -2 | 0.46867 | 0.35419 | 0.35223 | 0.26129 |
| -1 | 0.16467 | 0.08179 | 0.12212 | 0.05911 |
| | $P_{GG} = 0.5$ | | $P_{BB} = 0.5$ | |
| -3 | 0.56322 | 0.32366 | 0.44058 | 0.24605 |
| -2 | 0.32754 | 0.12348 | 0.24303 | 0.08867 |
| -1 | 0.15243 | 0.05609 | 0.10993 | 0.03993 |
| | $P_{GG} = 0.2$ | | $P_{BB} = 0.8$ | |
| -3 | 0.73127 | 0.57783 | 0.57146 | 0.4417 |
| -2 | 0.46867 | 0.35419 | 0.35223 | 0.26129 |
| -1 | 0.22775 | 0.16561 | 0.16732 | 0.12115 |
| | $P_{GG} = 0.8$ | | $P_{BB} = 0.2$ | |
| -3 | 0.27282 | 0.03513 | 0.20121 | 0.02449 |
| -2 | 0.15056 | 0.01980 | 0.10891 | 0.01393 |
| -1 | 0.06732 | 0.00986 | 0.04852 | 0.00589 |

Од Табела 5.3 можеме да заклучиме дека резултатите добиени со новиот РафаленАДП, се од 2 до 8 пати подобри од соодветните резултати добиени со стариот АДП.

Табела 5.4: Експериментални резултати за $R = 1/8$

| $P_e(B)$ | $BER_{cut}$ | $BER_{b-cut}$ | $BER_{4sets}$ | $BER_{b-4sets}$ |
|---|---|---|---|---|
| | $P_{GG} = 0.8,$ | | $P_{BB} = 0.8$ | |
| -3 | 0.40468 | 0.21957 | 0.202399 | 0.07334 |
| -2 | 0.23806 | 0.11587 | 0.08246 | 0.02539 |
| -1 | 0.10796 | 0.04945 | 0.02266 | 0.00871 |
| | $P_{GG} = 0.5$ | | $P_{BB} = 0.5$ | |
| -3 | 0.38576 | 0.15412 | 0.20372 | 0.043289 |
| -2 | 0.21698 | 0.07825 | 0.07806 | 0.01490 |
| -1 | 0.09497 | 0.03024 | 0.02002 | 0.00385 |
| | $P_{GG} = 0.2$ | | $P_{BB} = 0.8$ | |
| -3 | 0.53986 | 0.42327 | 0.35535 | 0.22462 |
| -2 | 0.33082 | 0.24201 | 0.15007 | 0.08345 |
| -1 | 0.15119 | 0.10676 | 0.03686 | 0.02265 |
| | $P_{GG} = 0.8$ | | $P_{BB} = 0.2$ | |
| -3 | 0.17904 | 0.02085 | 0.05540 | 0.00328 |
| -2 | 0.09442 | 0.01037 | 0.02074 | 0.00134 |
| -1 | 0.04009 | 0.00503 | 0.00703 | 0.00060 |
| $P_e(B)$ | $PER_{cut}$ | $PER_{b-cut}$ | $PER_{4sets}$ | $PER_{b-4sets}$ |
| | $P_{GG} = 0.8,$ | | $P_{BB} = 0.8$ | |
| -3 | 0.61031 | 0.34835 | 0.40048 | 0.15092 |
| -2 | 0.38256 | 0.19693 | 0.17461 | 0.05457 |
| -1 | 0.17821 | 0.08640 | 0.04860 | 0.01771 |
| | $P_{GG} = 0.5$ | | $P_{BB} = 0.5$ | |
| -3 | 0.60865 | 0.26116 | 0.40710 | 0.09454 |
| -2 | 0.37028 | 0.13637 | 0.16748 | 0.03204 |
| -1 | 0.16402 | 0.05501 | 0.04169 | 0.00835 |
| | $P_{GG} = 0.2$ | | $P_{BB} = 0.8$ | |
| -3 | 0.78650 | 0.65300 | 0.65797 | 0.44758 |
| -2 | 0.53269 | 0.40408 | 0.31820 | 0.18130 |
| -1 | 0.26101 | 0.18613 | 0.08208 | 0.04910 |
| | $P_{GG} = 0.8$ | | $P_{BB} = 0.2$ | |
| -3 | 0.30645 | 0.03809 | 0.11880 | 0.00604 |
| -2 | 0.16345 | 0.01886 | 0.04428 | 0.00252 |
| -1 | 0.07063 | 0.00907 | 0.01404 | 0.00115 |

Во Табела 5.4, дадени се експериментални резултати за веројатноста за бит-грешка и веројатноста за пакет-грешка за код со рата $1/8$. Притоа, користени се истите ознаки како и претходно. Од оваа табела, можеме да направиме слични заклучоци за кодот за рата $1/8$, како за кодот со рата $1/4$. Оттука, можеме да заклучиме дека за сите вредности на $SNR$, резултатите за $BER$ и $PER$ добиени со новите алгоритми се подобри во споредба со соодветните резултати добиени со старите верзии на алгоритмите. Исто така, со

РафаленАД4П се добиваат од 2 до 8 пати подобри резултати отколку со РафаленАДП. Резултатите од овој дел се објавени во [79].

## 5.4 Експериментални резултати за слики

Во овој дел претставени се експерименталните резултати добиени со $RCBQ$ за пренос на слики низ канали со рафални грешки. Всушност, ги истражуваме перформансите на РафаленАДП и РафаленАД4П за пренос на слики низ Гилберт-Елиот канал. Направивме експерименти со двата вида на Гилберт-Елиот каналите (бинарно симетрични канали и Гаусови канали). Сите експерименти се направени за код $(72, 576)$ со рата $R = 1/8$, $B_{max} = 5$ и со истите параметри како и обичните пораки во претходното поглавје.

Прво, на Сликата 5.3 претставени се слики за $P_{GG} = 0.2$, $P_{BB} = 0.8$ добиени после пренос низ канал без користење на код за корекција на грешки. Првата слика е добиена после пренос низ Гилберт-Елиот канал со БСК со веројатноста за бит-грешка во добра состојба $P_e(G) = 0.01$ и веројатноста за бит-грешка кога каналот е во лоша состојба $P_e(B) = 0.16$, а втората со Гаусови канали со $SNR_G = 4$ кога каналот е во добра состојба и $SNR_B = -3$ кога каналот е во лоша состојба.

Слика 5.3: Без користење на код за корекција на грешки

### 5.4.1 Експерименти за слики при пренос низ Гилберт-Елиот моделот со бинарно симетрични канали

Во сите експерименти за Гилберт-Елиот модел со бинарно симетрични канали ја користиме веројатноста за бит-грешка во добра состојба $P_e(G) = 0.01$ и неколку различни вредности на веројатноста за бит-грешка кога каналот е

во лоша состојба $P_e(B) \in \{0.16; 0.13; 0.1\}$ и следните комбинации на веројатностите на премин од добра во добра состојба $P_{GG}$ и од лоша во лоша состојба $P_{BB}$:

- $P_{GG} = 0.2$ и $P_{BB} = 0.8$
- $P_{GG} = 0.8$ и $P_{BB} = 0.2$

На Слика 5.4 претставени се сликите добиени за $P_{GG} = 0.2$, $P_{BB} = 0.8$ и за сите соодветни вредности за $P_e(B)$ добиени со РафаленАДП, а на Слика 5.5 истите слики добиени после примена на филтерот предложен во Поглавје 3.4.1. Сликите добиени за $P_{GG} = 0.8$ и $P_{BB} = 0.2$ дадени се на Слика 5.6 и Слика 5.7.

Слика 5.4: Слики добиени со РафаленАДП без користење на филтер за $P_{GG} = 0.2$, $P_{BB} = 0.8$

Слика 5.5: Слики добиени со РафаленАДП со користење на филтер за $P_{GG} = 0.2$, $P_{BB} = 0.8$

Сликите добиени со РафаленАД4П за $P_{GG} = 0.2$, $P_{BB} = 0.8$ претставени се на Слика 5.8 (без филтер) и Слика 5.9 (со филтер), додека сликите за $P_{GG} = 0.8$, $P_{BB} = 0.2$ претставени се на Слика 5.10 (без филтер) и Слика 5.11 (со филтер).

Слика 5.6: Слики добиени со РафаленАДП без користење на филтер за $P_{GG} = 0.8$, $P_{BB} = 0.2$

Слика 5.7: Слики добиени со РафаленАДП со користење на филтер за $P_{GG} = 0.8$, $P_{BB} = 0.2$

Слика 5.8: Слики добиени со РафаленАД4П без користење на филтер за $P_{GG} = 0.2$, $P_{BB} = 0.8$

Од Сликата 5.4 и Сликата 5.8 (за $P_{GG} = 0.2$, $P_{BB} = 0.8$) можеме да заклучиме дека РафаленАД4П дава подобри резултати во споредба со РафаленАДП за сите разгледани вредности на $P_e(B)$. Споредувајќи ги сликите добиени пред и после примена на филтерот, можеме да заклучиме дека филтерот дава значително подобрување на квалитетот на сликите декодирани со двата алгоритми. Но, очигледно е дека филтерот дава подобри резултати за сликите добиени со РафаленАДП во споредба со РафаленАД4П. Причината за ова е

Слика 5.9: Слики добиени со РафаленАД4П со користење на филтер за $P_{GG} = 0.2$, $P_{BB} = 0.8$

Слика 5.10: Слики добиени со РафаленАД4П без користење на филтер за $P_{GG} = 0.8$, $P_{BB} = 0.2$

Слика 5.11: Слики добиени со РафаленАД4П со користење на филтер за $P_{GG} = 0.8$, $P_{BB} = 0.2$

поголемиот број на недетектирани грешки во експериментите со РафаленАД4П. Всушност, филтерот не може да ги детектира овој вид на грешки. Сликите за $P_{GG} = 0.8$, $P_{BB} = 0.2$ (Слика 5.6, Слика 5.7, Слика 5.10, Слика 5.11) се појасни поради помалиот број на грешки во каналот со овие веројатности на премин. Поради тоа, во овие слики нема голема разлика помеѓу сликите декодирани со двата алгоритми и по примената на филтерот.

### 5.4.2 Експерименти за слики при пренос низ Гилберт-Елиот моделот со Гаусови канали

Во овој дел, дадени се експерименталните резултати за Гилберт-Елиот модел со Гаусови канали за $SNR_G = 4$ и различни вредности на $SNR_B \in \{-3, -2, -1\}$ и со истите веројатности на премин како и во експериментите со бинарно симетрични канали.

Сликите добиени со РафаленАДП за $P_{GG} = 0.2$, $P_{BB} = 0.8$ за сите $SNR_B$ дадени се на Слика 5.12. Сликите на Слика 5.13 се добиени од сликите дадени на Слика 5.12 после примена на филтерот. Сликите за $P_{GG} = 0.8$, $P_{BB} = 0.2$ дадени се на Слика 5.14 (без филтер) и Слика 5.15 (со филтер).

Слика 5.12: Слики добиени со РафаленАДП без користење на филтер за $P_{GG} = 0.2$, $P_{BB} = 0.8$

Слика 5.13: Слики добиени со РафаленАДП со користење на филтер за $P_{GG} = 0.2$, $P_{BB} = 0.8$

На Слика 5.16 (без филтер) и на Слика 5.17 (со филтер) претставени се сликите добиени со РафаленАД4П за $P_{GG} = 0.2$, $P_{BB} = 0.8$. Сликите добиени со овој алгоритам за $P_{GG} = 0.8$, $P_{BB} = 0.2$ се дадени на Слика 5.18 (без филтер) и на Слика 5.19 (со филтер).

Анализирајќи ги сликите дадени во овој дел (за пренос низ Гилберт-Елиот со Гаусови канали) можеме да ги изведеме истите заклучоци како и за сликите

Слика 5.14: Слики добиени со РафаленАДП без користење на филтер за $P_{GG} = 0.8$, $P_{BB} = 0.2$

Слика 5.15: Слики добиени со РафаленАДП со користење на филтер за $P_{GG} = 0.8$, $P_{BB} = 0.2$

Слика 5.16: Слики добиени со РафаленАД4П без користење на филтер за $P_{GG} = 0.2$, $P_{BB} = 0.8$

низ Гилберт-Елиот со бинарно симетрични канали. Резултатите од овој дел се дадени во [80].

Слика 5.17: Слики добиени со РафаленАД4П со користење на филтер за $P_{GG} = 0.2$, $P_{BB} = 0.8$

Слика 5.18: Слики добиени со РафаленАД4П без користење на филтер за $P_{GG} = 0.8$, $P_{BB} = 0.2$

Слика 5.19: Слики добиени со РафаленАД4П со користење на филтер за $P_{GG} = 0.8$, $P_{BB} = 0.2$

## 5.5 Брз-алоритам-за-декодирање со криптокодови за рафални грешки

Во овој дел, сакаме да ги примениме брзите алгоритми на RCBQ дадени во Глава 4., за пренесување пораки преку канал со рафални грешки. Всушност, ние ги комбинираме претходните две идеи во два нови алгоритми за кодирање/декодирање наречени БрзР-алгоритам-за-декодирање-со-пресеци

(FastB-Cut-Decoding или БрзР-АДП) и БрзР-алгоритам-за-декодирање-со-4-пресеци (FastB-4-Sets-Cut-Decoding или БрзР-АД4П). Имено, на почетокот оригиналната порака ја кодираме со еден од алгоритмите за кодирање (БрзАДП или БрзАД4П). После тоа, применуваме интерливер на добиениот коден збор (пред соединувањето). Добиената порака со интерливерот се пренесува преку рафален канал. На примените пораки на излезот на каналот, после делењето на две (или четири) делови, го применуваме деинтерливерот на секој дел и потоа сите делови се декодираат со помош на соодветниот алгоритам за декодирање.

### 5.5.1 Експериментални резултати

Во овој дел претставени се експерименталните резултати добиени со БрзР-АДП и БрзР-АД4П за рата $R = 1/4$ и $R = 1/8$, за пренос преку канал со рафални грешки. Во нашите експерименти го користиме Гилберт-Елиот моделот каде и во двете состојби каналот е Гаусов, а вредноста на $SNR_G$ во добра состојба е висока, а вредноста на $SNR_B$ во лоша состојба е мала.

Ги споредуваме резултати добиените со БрзР-АДП и БрзР-АД4П со соодветните резултати за РафаленАДП и РафаленАД4П. Исто така, за да ја покажеме ефикасноста на брзите алгоритмите, ги презентираме процентите од пораките за кои декодирањето завршило со $B_{max} = 1, 2, 3, 4$ или $5$.

Во експериментите ги употребуваме следните параметри за кодот.

– За кодот $(72, 288)$ во РафаленАДП и БрзР-АДП, параметрите се:

  – патерн за редундантност: 1100 1110 1100 1100 1110 1100 1100 1100 0000 за рата $1/2$ и два различни клучеви од 10 нибли.

– За кодот $(72, 576)$, се употребуваат следните параметри:

  – во РафаленАДП и БрзР-АДП - патерн за редундантност: 1100 1100 1000 00001100 1000 1000 0000 1100 1100 1000 0000 1100 1000 1000 0000 0000 0000, за рата $1/4$ и два различни клучеви од 10 нибли,
  – во РафаленАД4П и БрзР-АД4П - патерн за редундантност: 1100 1110 1100 1100 1110 1100 1100 1100 0000 за рата $1/2$ и четири различни клучеви од 10 нибли.

Во сите експерименти ја употребуваме истата квазигрупа на $Q$ дадена во Табела 3.2.

Во експериментите со РафаленАДП за кодот $(72, 288)$ користиме $B_{max} = 4$, а во експериментите со алгоритмот БрзР-АДП максимална вредност на $B_{max}$ е 4. За кодот $(72, 576)$ во алгоритмите РафаленАДП и РафаленАД4П користиме $B_{max} = 5$, а во БрзР-АДП и БрзР-АД4П максималната вредност на $B_{max}$ е 5.

Во сите експерименти вредноста на $SNR$ во добра состојба е $SNR_G = 4$. Направивме експерименти за различни вредности на $SNR$ во лоша состојба $SNR_B \in \{-3, -2, -1\}$ и за следниве комбинации на веројатности за премин од добра во добра состојба $P_{GG}$ и од лоша во лоша состојба $P_{BB}$ во Гилберт-Елиот моделот:

- $P_{GG} = 0.8$ и $P_{BB} = 0.8$
- $P_{GG} = 0.5$ и $P_{BB} = 0.5$
- $P_{GG} = 0.2$ и $P_{BB} = 0.8$
- $P_{GG} = 0.8$ и $P_{BB} = 0.2$

Во Табела 5.5, дадени се експерименталните резултати за веројатноста за бит-грешка, $BER$ и веројатноста за пакет-грешка $PER$ за кодот $(72, 288)$. Со $BER_{b-cut}$ и $PER_{b-cut}$ ги означуваме веројатностите добиени со РафаленАДП, а со $BER_{fb-cut}$ и $PER_{fb-cut}$ веројатностите добиени со БрзР-АДП. Резултатите за $BER_{b-cut}$ и $PER_{b-cut}$ за РафаленАДП, се презентирани во Поглавје 5.3.2.

Табела 5.5: Експериментални резултати за код $(72, 288)$

| $SNR_B$ | $PER_{fb-cut}$ | $PER_{b-cut}$ | $BER_{fb-cut}$ | $BER_{b-cut}$ |
|---|---|---|---|---|
| | | $P_{GG} = 0.8$ | $P_{BB} = 0.8$ | |
| $-3$ | 0.27657 | 0.32366 | 0.20524 | 0.24605 |
| $-2$ | 0.15431 | 0.17727 | 0.11143 | 0.13127 |
| $-1$ | 0.06588 | 0.08179 | 0.04531 | 0.05911 |
| | | $P_{GG} = 0.5$ | $P_{BB} = 0.5$ | |
| $-3$ | 0.20838 | 0.23135 | 0.14832 | 0.16945 |
| $-2$ | 0.10023 | 0.12348 | 0.06993 | 0.08867 |
| $-1$ | 0.04097 | 0.05609 | 0.02839 | 0.03993 |
| | | $P_{GG} = 0.2$ | $P_{BB} = 0.8$ | |
| $-3$ | 0.57402 | 0.57783 | 0.43486 | 0.44170 |
| $-2$ | 0.35088 | 0.35419 | 0.25447 | 0.26129 |
| $-1$ | 0.15315 | 0.16561 | 0.10795 | 0.12115 |
| | | $P_{GG} = 0.8$ | $P_{BB} = 0.2$ | |
| $-3$ | 0.02254 | 0.03513 | 0.01529 | 0.02449 |
| $-2$ | 0.00994 | 0.01980 | 0.00669 | 0.01393 |
| $-1$ | 0.00382 | 0.00986 | 0.00253 | 0.00589 |

Анализирајќи ги резултатите од Табела 5.5, можеме да заклучиме дека за сите вредности на $SNR_B$ и сите комбинации на веројатностите за премин резултатите за $BER_{fb-cut}$ и $PER_{fb-cut}$ се малку подобри од соодветните резултати за $BER_{b-cut}$ и $PER_{b-cut}$.

Кога декодирањето во БрзР-АДП завршува $B_{max} = 1$ или $B_{max} = 2$ декодирањето со овој алгоритам е многу побрзо отколку со РафаленАДП. Затоа, во Табела 5.6 го даваме процентот на пораки за кои декодирањето завршува со $B_{max} = 1$, $B_{max} = 2$, $B_{max} = 3$ или $B_{max} = 4$ во експериментите со БрзР-АДП. Од резултатите дадени таму, можеме да видиме дека процентите зависат не само од вредноста на $SNR_B$, туку и од веројатностите за премин од лоша во лоша и од добра до добра состојба. За $P_{GG} = 0.8, P_{BB} = 0.2$ и сите вредности на $SNR_B$ за повеќе од 75% од пораките декодирањето успешно завршува со $B_{max} = 1$ или $B_{max} = 2$. За $P_{GG} = 0.8, P_{BB} = 0.8$ или $P_{GG} = 0.5, P_{BB} = 0.5$ и $SNR_B = -1$ повеќе од 50% од декодирањето е успешно завршено со $B_{max} = 1$ или $B_{max} = 2$. Во другите експерименти, каде што веројатноста каналот да биде во лоша состојба

е поголема, имаме помал процент на успешно декодирање со $B_{max} = 1$ или $B_{max} = 2$. Значи, можеме да заклучиме дека новиот алгоритам ја подобрува брзината на декодирање на RCBQ за пренесување преку Гилберт-Елиот канал со помала веројатност за лоша состојба.

Табела 5.6: Процент на пораки декодирани за различни вредности на $B_{max}$

| $SNR_B$ | $B_{max} = 1$ | $B_{max} = 2$ | $B_{max} = 3$ | $B_{max} = 4$ |
|---|---|---|---|---|
| | | $P_{GG} = 0.8$ | $P_{BB} = 0.8$ | |
| $-3$ | 15.79% | 18.23% | 17.71% | 48.26% |
| $-2$ | 20.06% | 22.56% | 22.34% | 35.04% |
| $-1$ | 24.95% | 29.59% | 24.25% | 21.21% |
| | | $P_{GG} = 0.5$ | $P_{BB} = 0.5$ | |
| $-3$ | 5.11% | 20.39% | 26.30% | 48.19% |
| $-2$ | 8.92% | 29.10% | 30.34% | 31.64% |
| $-1$ | 18.17% | 37.82% | 27.70% | 16.31% |
| | | $P_{GG} = 0.2$ | $P_{BB} = 0.8$ | |
| $-3$ | 0.17% | 3.28% | 10.95% | 85.60% |
| $-2$ | 0.42% | 8.55% | 21.45% | 69.58% |
| $-1$ | 2.17% | 20.99% | 32.60% | 44.24% |
| | | $P_{GG} = 0.8$ | $P_{BB} = 0.2$ | |
| $-3$ | 42.42% | 32.80% | 16.32% | 8.46% |
| $-2$ | 50.97% | 32.00% | 12.66% | 4.37% |
| $-1$ | 60.10% | 29.56% | 8.13% | 2.22% |

Во Табела 5.7 и Табела 5.8, ги претставуваме експерименталните резултати за кодот $(72, 576)$ со рата $R = 1/8$. Ја споредуваме веројатноста на бит-грешката ($BER$) и пакет-грешката ($PER$) добиени со РафаленАДП, БрзР-АДП, РафаленАД4Пи БрзР-АД4П.

Со $BER_{b-cut}$, $PER_{b-cut}$, $BER_{b-4sets}$, $PER_{b-4sets}$ ги означуваме веројатностите за пакет-грешка и бит-грешка добиени со рафалните алгоритми (РафаленАДП и РафаленАД4П) и со $BER_{fb-cut}$, $PER_{fb-cut}$, $BER_{fb-4sets}$ и $PER_{fb-4sets}$ соодветните веројатности добиени со алгоритмите на БрзР алгоритмите (БрзР-АДП и БрзР-АД4П).

Од резултатите дадени во Табела 5.7 и Табела 5.8, можеме да заклучиме дека за сите комбинации на веројатности за премин и сите вредности на

$SNR_B$ за $BER$ и $PER$ добиени со новите брзи алгоритми се подобри од соодветните резултати добиени со рафалните алгоритмите. Споредувајќи ги резултатите од сите алгоритми, можеме да видиме дека најдобрите резултати (за сите параметри на каналот) се добиени со алгоритмот БрзР-АД4П. За $P_{GG} = 0.8, P_{BB} = 0.2$ и сите разгледани вредности на $SNR_B$, вредностите на $BER$ и $PER$ добиени со овие алгоритми се еднакви на 0. Исто така, од времетраењето на експериментите заклучивме дека БрзР-АД4П е многу побрз од другите три алгоритми разгледани во овој дел.

Во Табела 5.9, даден е процентот на пораки кај кои декодирањето со БрзР-АДП и БрзР-АД4П завршува со $B_{max} = 1$, $B_{max} = 2$, $B_{max} = 3$, $B_{max} = 4$ или $B_{max} = 5$.

Табела 5.7: Експериментални резултати за $BER$ за кодот $(72, 576)$

| $SNR_B$ | $BER_{fb-cut}$ | $BER_{b-cut}$ | $BER_{fb-4sets}$ | $BER_{b-4sets}$ |
|---|---|---|---|---|
| | $P_{GG} = 0.8,$ | | $P_{BB} = 0.8$ | |
| $-3$ | 0.06381 | 0.09182 | 0.01421 | 0.03798 |
| $-2$ | 0.02148 | 0.03935 | 0.00322 | 0.01631 |
| $-1$ | 0.00624 | 0.01347 | 0.00044 | 0.00518 |
| | $P_{GG} = 0.5$ | | $P_{BB} = 0.5$ | |
| $-3$ | 0.03417 | 0.05539 | 0.00588 | 0.02263 |
| $-2$ | 0.01257 | 0.02198 | 0.00138 | 0.00853 |
| $-1$ | 0.00250 | 0.00707 | 0.00022 | 0.00251 |
| | $P_{GG} = 0.2$ | | $P_{BB} = 0.8$ | |
| $-3$ | 0.16994 | 0.19376 | 0.06586 | 0.08592 |
| $-2$ | 0.06408 | 0.08687 | 0.01560 | 0.03244 |
| $-1$ | 0.01697 | 0.02736 | 0.00258 | 0.01085 |
| | $P_{GG} = 0.8$ | | $P_{BB} = 0.2$ | |
| $-3$ | 0.00200 | 0.00477 | 0 | 0.00225 |
| $-2$ | 0.00063 | 0.00174 | 0 | 0.00091 |
| $-1$ | 0.00013 | 0.00075 | 0 | 0.00026 |

Табела 5.8: Експерименталните резултати за $PER$ за код $(72, 576)$

| $SNR_B$ | $PER_{fb-cut}$ | $PER_{b-cut}$ | $PER_{fb-4sets}$ | $PER_{b-4sets}$ |
|---|---|---|---|---|
| | | $P_{GG} = 0.8,$ | $P_{BB} = 0.8$ | |
| $-3$ | 0.11470 | 0.16734 | 0.02657 | 0.06552 |
| $-2$ | 0.04155 | 0.07273 | 0.00662 | 0.02931 |
| $-1$ | 0.01202 | 0.02556 | 0.00079 | 0.00900 |
| | | $P_{GG} = 0.5$ | $P_{BB} = 0.5$ | |
| $-3$ | 0.06509 | 0.10765 | 0.01202 | 0.03910 |
| $-2$ | 0.02304 | 0.04198 | 0.00324 | 0.01419 |
| $-1$ | 0.00540 | 0.01375 | 0.00043 | 0.00446 |
| | | $P_{GG} = 0.2$ | $P_{BB} = 0.8$ | |
| $-3$ | 0.29551 | 0.35628 | 0.12025 | 0.14653 |
| $-2$ | 0.11874 | 0.16381 | 0.02988 | 0.05688 |
| $-1$ | 0.03283 | 0.05357 | 0.00490 | 0.01879 |
| | | $P_{GG} = 0.8$ | $P_{BB} = 0.2$ | |
| $-3$ | 0.00374 | 0.00886 | 0 | 0.00410 |
| $-2$ | 0.00115 | 0.00382 | 0 | 0.00158 |
| $-1$ | 0.00029 | 0.00158 | 0 | 0.00043 |

Од Табела 5.9, можеме да видиме дека во сите експерименти добивме подобри проценти (поголеми проценти за помали $B_{max}$ и помали проценти за поголеми $B_{max}$) со БрзР-АД4П отколку so БрзР-АДП. Во скоро сите случаи, процентот на пораки кај кои за декодирање со БрзР-алгоритам-за-декодирање-со-4-пресеци е потребно $B_{max} = 5$ со овие алгоритми е под $8\%$ и за повеќе од $60\%$ од пораките декодирањето завршува со $B_{max} \leqslant 3$. Исклучок од ова се само случаите кога $P_{GG} = 0.2, P_{BB} = 0.8, SNR_B \leqslant -2$ и $P_{GG} = 0.8, P_{BB} = 0.8, SNR_B = -3$. Исто така, за $P_{GG} = 0.8, P_{BB} = 0.2$ повеќе од половина од декодирањата завршиле со $B_{max} = 1$. Ова значи дека за овие параметри на каналот алгоритмот БрзР-АД4П е побрз од РафаленАД4П.

Табела 5.9: Процент на пораки декодирани со различни вредности на $B_{max}$

| | БрзР-алгоритам-за-декодирање-со-пресеци | | | | |
|---|---|---|---|---|---|
| $SNR_B$ \ $B_{max}$ | 1 | 2 | 3 | 4 | 5 |
| | $P_{GG} = 0.8 \quad P_{BB} = 0.8$ | | | | |
| $-3$ | 3.49% | 14.38% | 24.80% | 28.57% | 28.76% |
| $-2$ | 5.16% | 22.52% | 32.39% | 24.98% | 14.96% |
| $-1$ | 8.10% | 34.92% | 34.77% | 16.15% | 6.06% |
| | $P_{GG} = 0.5 \quad P_{BB} = 0.5$ | | | | |
| $-3$ | 0.22% | 9.41% | 33.84% | 33.83% | 22.70% |
| $-2$ | 0.78% | 20.39% | 43.13% | 25.25% | 10.45% |
| $-1$ | 2.62% | 39.57% | 40.81% | 13.28% | 3.72% |
| | $P_{GG} = 0.2 \quad P_{BB} = 0.8$ | | | | |
| $-3$ | 0% | 0.18% | 6.57% | 29.51% | 63.75% |
| $-2$ | 0% | 1.56% | 20.51% | 41.17% | 36.76% |
| $-1$ | 0.04% | 7.75% | 40.79% | 35.68% | 15.74% |
| | $P_{GG} = 0.8 \quad P_{BB} = 0.2$ | | | | |
| $-3$ | 17.09% | 49.87% | 24.72% | 6.55% | 1.78% |
| $-2$ | 25.58% | 52.37% | 17.68% | 3.64% | 0.73% |
| $-1$ | 36.12% | 51.60% | 10.25% | 1.68% | 0.35% |
| | БрзР-алгоритам-за-декодирање-со-4-пресеци | | | | |
| $SNR_B$ \ $B_{max}$ | 1 | 2 | 3 | 4 | 5 |
| | $P_{GG} = 0.8 \quad P_{BB} = 0.8$ | | | | |
| $-3$ | 23.22% | 18.02% | 22.20% | 25.42% | 11.13% |
| $-2$ | 27.61% | 25.09% | 26.56% | 17.04% | 3.70% |
| $-1$ | 35.64% | 32.21% | 24.14% | 7.33% | 0.68% |
| | $P_{GG} = 0.5 \quad P_{BB} = 0.5$ | | | | |
| $-3$ | 7.13% | 20.07% | 35.07% | 30.54% | 7.19% |
| $-2$ | 12.36% | 31.98% | 38.33% | 15.50% | 1.84% |
| $-1$ | 23.95% | 42.21% | 28.65% | 4.85% | 0.34% |
| | $P_{GG} = 0.2 \quad P_{BB} = 0.8$ | | | | |
| $-3$ | 0.17% | 2.12% | 8.19% | 45.62% | 43.90% |
| $-2$ | 0.68% | 6.96% | 25.23% | 50.45% | 16.68% |
| $-1$ | 3.18% | 19.99% | 44.45% | 28.76% | 3.62% |
| | $P_{GG} = 0.8 \quad P_{BB} = 0.2$ | | | | |
| $-3$ | 54.50% | 31.98% | 11.89% | 1.55% | 0.09% |
| $-2$ | 63.89% | 29.17% | 6.44% | 0.48% | 0.01% |
| $-1$ | 73.37% | 23.83% | 2.68% | 0.12% | 0% |

Со овие нови алгоритми, БрзР-АДП и БрзР-АД4П, добивме подобри резултати за веројатностите на пакет-грешка и бит-грешка отколку со претходно дефинираните (РафаленАДП и РафаленАД4П) алгоритми на RCBQ за пренос преку канали со рафални грешки. Исто така, од презентираните проценти на пораки кај кои декодирањето завршува со различни вредности од $B_{max}$ можеме да заклучиме дека за некои параметри на каналот, БрзР-АДП и БрзР-АД4П алгоритмите овозможуваат побрзо декодирање. Резултатите од овој дел се прифатени за печатење во [87].

# Глава 6

# Блок-вериги (Blockchain)

## 6.1 Технологијата на блок-веригите - поим и основи

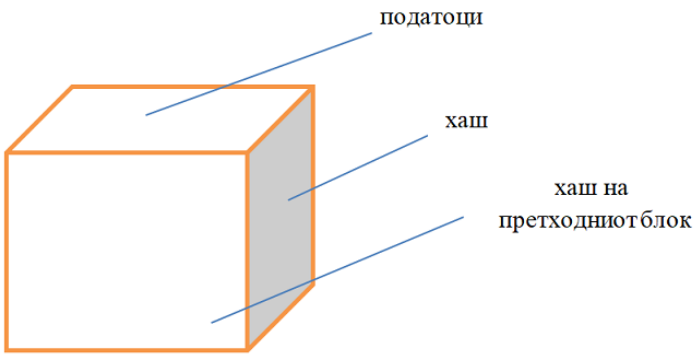
*Блок-веригата* претставува децентрализирана главна книга од трансакции, распределена помеѓу сите компјутери во една $P2P$ мрежа каде што за секој кој е конектиран на мрежата, се видливи сите детали за трансакциите. Всушност, блок-веригата е растечка низа на поврзани блокови. Секој блок се состои од валидни трансакции, временски печат, хаш поинтер до претходниот блок во ланецот и број што се генерира само еднаш и кој учествува во формирањето на хаш вредноста на блокот (nonce), како што е прикажано на Слика 6.1.


Слика 6.1: Едноставна блок-верига

Кога некој бара трансакција или кога две страни разменуваат податоци, како што се пари, договор или кое било средство кое може да е дигитално

опишано, бараната трансакција се шири во $P2P$ мрежата која се состои од компјутери наречени јазли. Оваа мрежа од компјутери ја валидира трансакцијата и статусот на корисникот користејќи познати алгоритми.

Во зависност од мрежните параметри, трансакцијата или ќе се верификува веднаш или се запишува во безбеден запис и се става во списокот на трансакции што чекаат за верификација. Секој блок се идентификува со хаш и содржи хедер (референца до хашот на претходниот блок) и група од трансакции (Слика 6.2).



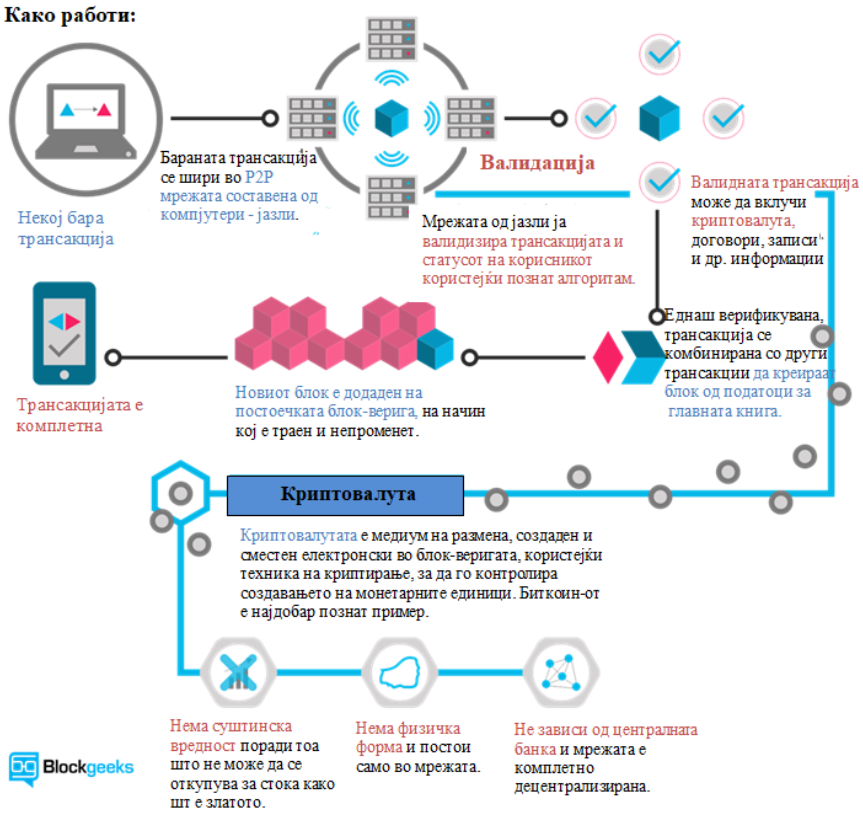

Слика 6.2: Еден блок во блок-веригата

Секвенцата од поврзани блокови креира безбедна и независна верига. Блоковите мора прво да се валидираат па потоа да бидат додадени во блок-веригата. Кога блокот се верификува, се дистрибуира низ мрежата (се додава во постоечката верига) и секој јазол го додава блокот во мнозинската блок-верига. Тогаш трансакцијата е комплетна. Шематски приказ на технологијата на блок-верига е даден на Слика 6.3.

Кога злонамерниот корисник се обидува да подметне изменет блок во синџирот, хаш функцијата на тој блок и на сите следни блокови ќе се промени. Овие промени ќе бидат детектирани од другите јазли и тие ќе го отфрлат блокот од мнозинската блок-верига, спречувајќи измама во ланецот, прикажано на Слика 6.4.

Постојат четири типа на блок-вериги:

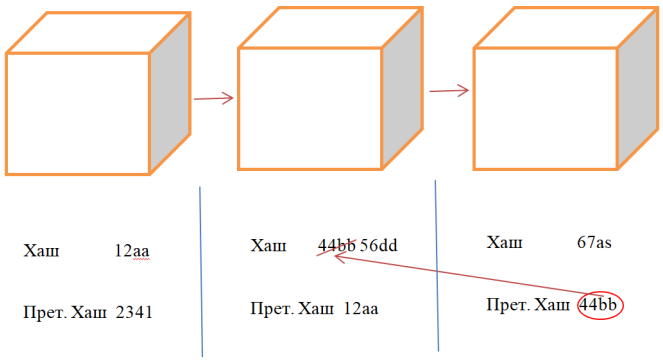
– *Јавна блок-верига* - во која нема никакви ограничувања за пристап. Сите трансакции што се одвиваат кај јавната блок-верига се целосно тран-




Слика 6.3: Технологија на блок-веригата [50]

спарентни, т.е. секој има пристап до секоја трансакција некогаш направена. Секој со интернет конекција може да праќа трансакции, а исто така да биде и валидатор (т.е. да учествува во извршувањето на консенсус протоколот). Најпопуларните иновации на јавната блок-верига се дигиталните валути *биткоин* и *етериум* [19].

– *Приватна блок-верига* - на корисниците во оваа блок-верига потребна им е дозвола да се приклучат. Трансакциите се приватни и достапни само на корисниците на кои им е дадена дозвола за да се приклучат. Приватната блок-верига е поцентрализирана и субјектите кои управуваат со веригата имаат значителна контрола над учесниците во истиот.




Слика 6.4: Изменет блок во блок-веригата

– *Дозволена блок-верига* - за разлика од приватната блок-верига каде што еден субјект ја контролира целата блок-верига, дозволената блок-верига е управувана од група субјекти, т.е. овој вид може да се смета за поткатегорија на приватната блок-верига. Предноста на оваа блок-верига е тоа што може да обедини група на бизниси кои работаат заедно, но исто така и кои се натпреваруваат меѓу себе, а со тоа овозможувајќи им да бидат поефикасни, и индивидуално и колективно, соработувајќи во некои аспекти од нивните бизниси.

– *Хибридна блок-верига* - комбинирање на предноста за приватност на приватната или дозволената блок-верига со предностите за безбедност и транспарентност на јавната блок-верига. Ова придонесува за значителна флексибилност при изборот на субјектите, кои податоци да ги направат јавни и транспарентни, а кои приватни при формирањето на блок-веригата.

## 6.2 Биткоин

Во 2008 година, поединец или група луѓе, потпишани под името Сатоши Накамото, го издадоа трудот “Bitcoin: A Peer-To-Peer Electronic Cash System” [83]. Биткоинот, првата главна примена на блок-веригите, претставува $P2P$ верзија на електронски кеш, што овозможува директно онлајн плаќање од една страна на друга, без потреба од вклучување на трета страна. Оваа

криптовалута е прва децентрализирана валута која користи криптографија за безбедни трансакции. Секој кој ќе го инсталира отворениот изворен (open source) програм што го имплементира овој нов протокол може да биде дел од биткоин $P2P$ мрежата [7].

Основните карактеристики на биткоинот се:

– *Децентрализираност*, што значи не постоење на централна банка. Емитерите на оваа валута се корисниците или носителите на компјутерите кои "рударат" ("mining" компјутери) и кои ја верификуваат секоја трансакција.

– Тајноста обезбедена со криптографијата со јавен клуч ја дава *довербата* на оваа валута.

– *Автентикација* на трансакциите од еден до друг јазол во блок-веригата е направена со дигиталниот потпис.

– Дигиталниот потпис на пораката исто така обезбедува *интегритет* низ преносот.

Секоја трансакција се состои од:

– *влез* (ги покажува трансакциите со кои испраќачот прима биткоини)

– *количина* (сумата на биткоини кои испраќачот ги испраќа до примачот)

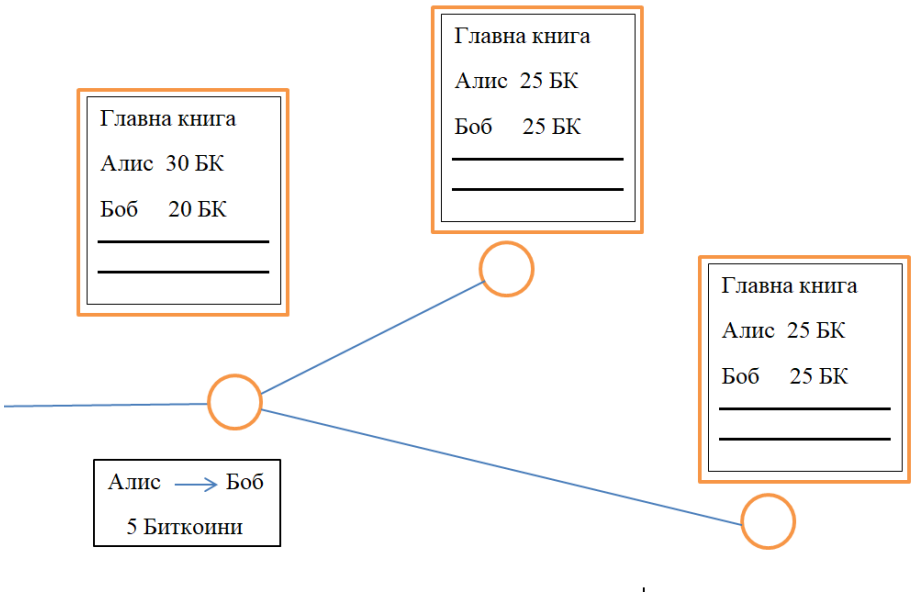
– *излез* (биткоин адресата на примачот)

### 6.2.1 Како функционира биткоинот?

За да Алис испрати пари (5 биткоини) на Боб, таа емитува порака со нејзината сметка (account) и сумата од 5 биткоини. Секој јазол што ја прима пораката прави ажурирање (update) на копијата на главната книга и потоа ја препраќа пораката за трансакцијата, како што е прикажано на Слика 6.5.

Корисникот во биткоин мрежата бара посебна лозинка за да може да ги прими испратените пари. Овој вид на безбедност го дава *дигиталниот потпис*. Со негова помош се докажува автентичноста на пораката преку математички алгоритам што заштитува од копирање и измама во дигиталната сфера.

Дигиталниот потпис користи два поврзани клуча:



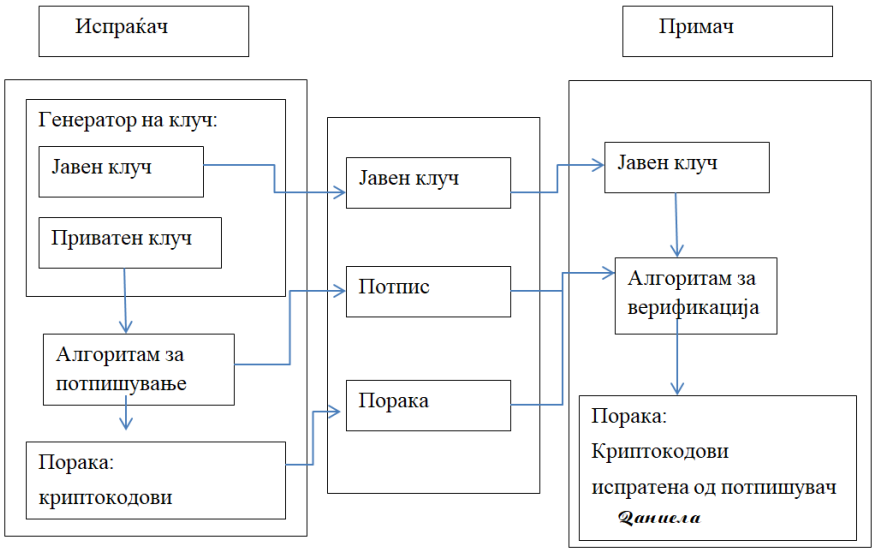

Слика 6.5: Испраќање порака

– *Приватен клуч* кој се користи да го креира потписот

– *Јавен клуч* кој се користи за проверка на припадноста на пораката.

Испраќачот генерира приватен клуч и јавен клуч. Потоа, ја потпишува пораката со потпис и го испраќа својот јавен клуч, потписот и пораката до мрежата. Јазолот или приемникот потоа проверува со употреба на алгоритам за верификација дали пораката е потпишана од испраќачот, што може да ја направи само сопственикот на приватниот клуч на јавниот клуч што е испратен. Процесот на генерирање и верификација на потписот шематски е прикажан на Слика 6.6.

Бидејќи потписот зависи од пораката, тој ќе биде различен за секоја трансакција, Слика 6.7. Оваа зависност значи дека никој не може да ја промени пораката додека поминува низ мрежата, бидејќи секоја промена на пораката ќе го уништи потписот. На овој начин се обезбедува интегритет при преносот.




Слика 6.6: Дигитален потпис

### 6.2.2 Трансакции и главна книга

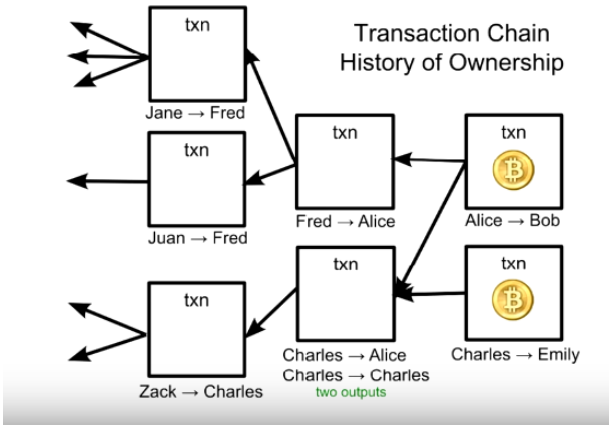
За да може Алис да прати 5 биткоини на Боб, таа мора да се повика на оние трансакции каде што Алис добива биткоини. Тоа се *влезни трансакции*. Другите јазли кои ја верификуваат оваа трансакција ги проверуваат сите влезни трансакции на Алис. Во биткоин мрежата се користи правилото, дека сите влезни трансакции мора да се искористат. Ако излезот е помал од влезните трансакции тогаш разликата од влезот и излезот како влезна трансакција повторно се враќа на испраќачот. Ланецот на трансакциите е прикажан на Слика 6.8.

Биткоин *рударите* ги верификуваат трансакциите, ги стават во блокови од трансакции и одлучуваат кој блок е следниот, т.е. *биткоин системот* ги групира трансакциите во блокови и ги поврзува во блок-веригата. Бидејќи, многу луѓе можат да создадат блокови во исто време, кој блок ќе влезе прв во блок-веригата се решава со употреба на хаш функцијата. Факторот што ја одредува веројатноста дека компјутерот *рудар* ќе го потврди блокот на трансакции е познат како *тежина (difficulty) фактор*.

| | | Дигитален потпис |
|---|---|---|
| Алис →Боб | 5 биткоини | 546373728... |
| Алис →Бил | 10 биткоини | 535372728... |
| Алис →Ким | 100 биткоини | 662772827... |

различен секој пат

Слика 6.7: Трансакциски пораки




Слика 6.8: Историја на сопственост на трансакциски ланец [34]

### 6.2.3 Хаширање во биткоин системот

Хаш функцијата е математички процес која прима еден влез, стринг со произволна должина, извршува операција над него и враќа излезен податок со фиксна должина. Биткоинот користи SHA-256 и RIPEMD-160 хаш функции. Всушност, кога се користи хашот во биткоин системот, тој се пресметува два пати. Почесто се користи хаш функцијата SHA-256, додека RIPEMD-160 се користи за креирање на кратки хашови за биткоин адреса. Во Табела 6.1 даден е пример на двојно хаширање на стрингот “bitcoin”.

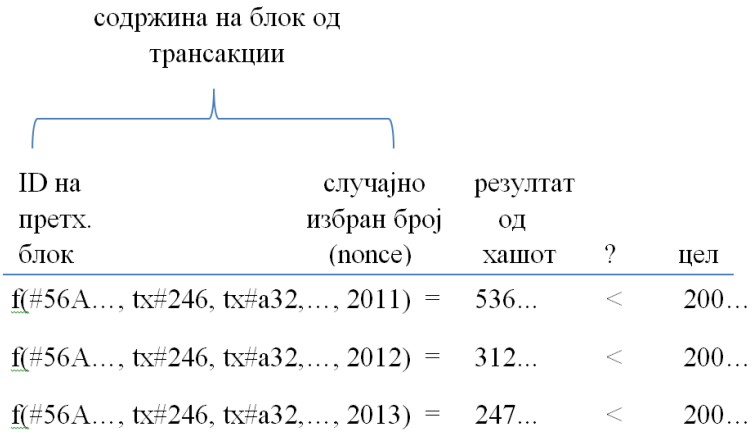
Табела 6.1: Пример на двојно хаширање на стрингот ”bitcoin”

| $SHA256$ | |
|---|---|
| *првата рунда со SHA-256* | *6b88c087247aa2f07ee1c5956b8e1a9f 4c7f892a70e324f1bb3d161e05ca107b* |
| *втората рунда со SHA-256* | *a23b7f87e4250b3a64b737f349c06422 f752f419cbb25ae9169a6cf1e23f4462* |
| *SHA-256 & RIPEMD-160* | |
| *првата рунда со SHA-256* | *6b88c087247aa2f07ee1c5956b8e1a9f 4c7f892a70e324f1bb3d161e05ca107b* |
| *втората рунда со RIPEMD-160* | *b67f99610e811d5eba9e 337877a8f55f766d7401f* |

### 6.2.4 Тежина факторот во биткоин системот

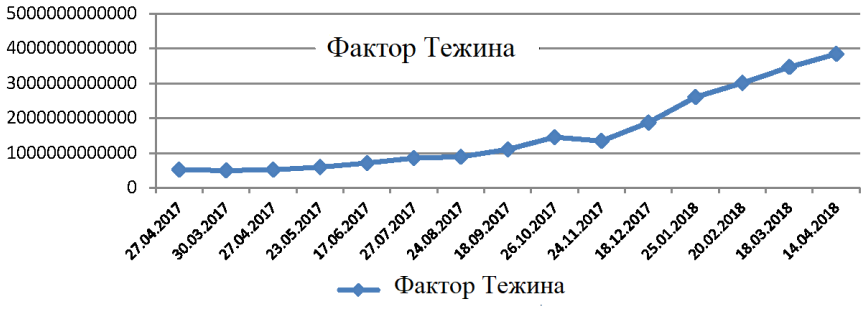
Излезот од хаш функцијата на блокот мора да биде под специфична вредност наречена *цел*. *Тежина* е мерка за тоа колку е тешко да се најде хашот под оваа цел. Вредноста на тежина факторот се менува на секои 2016 блокови. Ова ќе произведе, во просек, еден блок секој 10 минути (рата на блокот). Тековниот тежина фактор и *хаш ратата* (брзината со која компјутерот извршува операција во биткоин системот) го определуваат бројот на генерирани биткоини што може една индивидуа да ги постигне во текот на денот. Првиот рудар компјутер што ќе ја реши загатката (да најде хаш вредност под дадената цел), ќе направи емитување на неговиот блок и со тоа неговата група на трансакции е прифатена како следна во блок-веригата, Слика 6.9.

Во овој дел направивме анализи на тоа како факторот тежина се менува во месец август 2017 и годишно во периодот од април 2017 до април 2018 година. На Слика 6.10 и на Слика 6.11 прикажани се резултатите од овие анализи.




Слика 6.9: Хаш функција


Слика 6.10: Факторот тежина во август 2017




Слика 6.11: Факторот тежина од април 2017 до април 2018

### 6.2.5 Земји каде што биткоинот е легален

Биткоинот не е поврзан со ниту една влада бидејќи е независна валута. Дали тој е легален или не, зависи од земјата [22], [53].

- Во *САД* биткоинот се користи делумно во зависност од државата. Секој поединец кој рудари биткоини е предмет на оданочување. Од ноември 2013 година, биткоинот се смета за "легално средство за размена".
- Во *Австралија* легално е да се тргува, рударува и да се купуваат биткоини низ целата земја. Биткоинот се смета за вистински пари од 1 јули 2017 година.
- Во 2017 година *јапонската влада* го легализира биткоинот како метод на плаќање. Сега, земјата се обидува да ги заштити биткоин трансакциите.
- Рускиот заменик министер за финансии размислува да го легализира биткоинот и другите криптовалути во 2018 година, иако сè уште нема официјален извештај. *Русија* исто така работи на развој на КриптоРубља- нејзина сопствена криптовалута [28].
- Во *Белгија*: "Министерот за финансии посочи дека интервенцијата на владата во однос на Биткоин системот не се чини неопходна во моментот".

– Употребата на биткоинот во *Велика Британија* не е регулирана и биткоинот се третира како "приватни пари" што треба да се оданочуваат со ДДВ при тргување.

– *Франција* има одредени регулативи за употреба на биткоинот. Секоја употреба на биткоинот мора да содржи податоци за корисникот, што не е карактеристика на биткоинот.

– *Аргентина, Италија* и *Холандија* не го забрануваат, но тие се скептични кон биткоинот. Аргентинската влада манифестира скептизам заради невозможноста да се воспостави контрола врз новата валута од страна на властите, што се толкува како плодна почва за нелегални трансакции.

– Биткоинот во *Македонија* е се уште нелегален.

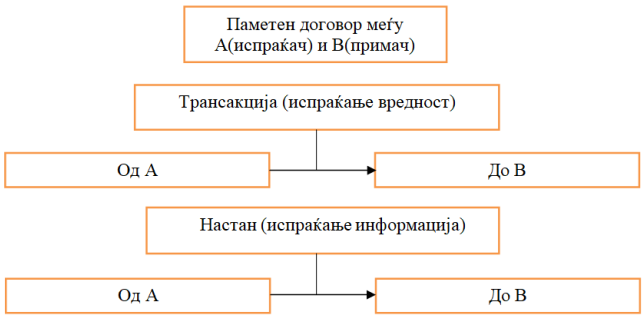
## 6.3 Паметни договори

Разменувањето на вредност, имот, акции или што било од вредност на транспарентен, без конфликтен начини, меѓу двајца сопственици врз основа на множество на услови вклучени во договор, го дефинираат паметниот договор. Овој договор е контролиран со децентрализираната согласност на блок-веригата. Како и традиционалниот договор, со паметниот договор се дефинирани правилата и казните околу него. Покрај тоа, паметниот договор автоматски ги спроведува тие обврски [26].

Секој паметен договор се состои од програмски код, датотека за складирање и биланс на сметки. Датотеката за складирање на договорот се чува во блок-веригата, додека програмскиот код на договорот го извршува мрежата на рудари во блок-веригата. Кодот на договорот се извршува секогаш кога еден корисник или некој друг договор испраќа порака. Договорот може исто така да прима и испраќа пари на други договори или корисници во билансот на сметката [10]. На Слика 6.12 е претставен еден едноставен модел на паметен договор.

### 6.3.1 Примена на паметните договори

Во овој дел, ќе дадеме неколку примени на паметните договори, анализирани во трудот [74]:




Слика 6.12: Едноставен модел на паметен договор

– **Предности на технологијата на блок-веригите во музичката индустрија**

Блок-веригата може да најде примена во музичката дистрибуција. Користејќи ја оваа технологија музичарите и музичките компании кои поседуваат музички права можат да добијат соодветни средства секогаш кога нивната музика се користи за комерцијални цели. Главната книга со трансакции на блок-веригата може да оствари директна врска помеѓу музичарите и обожавателите. Музиката со единствен ID и временски печат може да се направи јавна на главната книга. На овој начин, може да се спречат проблемите со преземање, копирање и модификација на дигиталната содржина. Секој запис на главната книга на блок-веригата има метаподатоци со информации за сопственоста и правата, така што секој може да ги види и верификува. Монетизацијата на музиката е направена со паметен договор кој овозможува плаќање на направените трансакции. На овој начин, корисниците можат да го изберат записот и веднаш да им платат на сопствениците со помош на криптовалута [40].
Онлајн музичката платформа на Бенџи Роџерс е една од компаниите што направија глобално децентрализирана книга на блок-верига како решение за проблемите со сопственоста, плаќањето и транспарентноста во музичката индустрија. Записите во оваа книга со трансакции се ”dot-Blockchain” - нов динамичен формат на датотека (еден вид зип датотека) кој ги носи сите информации за секој запис.

– **Паметните договори во осигурителните полиси**

Процесот на побарувањата во осигурителните полиси сè уште е рачен со голема количина човечка вклученост и може да треба многу време додека да се плати полисата. Паметните договори можат да го автоматизираат овој процес. Предностите на паметниот договор се: намалување на трошоците, зголемување на транспарентноста и довербата. Условите за влез на паметниот договор се запишуваат во блок-веригата. На пример, во случај на природна непогода, процесот на побарување се активира автоматски, без потреба од човечко дејство [47].

Еден пример на примена на паметните договори во осигурителните полиси претставува Етериск (Etherisc) кој од неодамна започна со продажба на токени со помош на технологијата на блок-веригите.

– **Примена на блок-веригата во системот за патенти**

Системот за патенти е инструмент кој што им помага на иноваторите да ги користат своите производи и да ги заштитат своите права над нив. Бидејќи системот за патенти сè уште има некои слабости, користењето на технологијата на блок-веригите би можело да помогне да се поправат некои од неговите недостатоци [3]. Ова може да се направи поради следниве две карактеристики на блок-веригата:

  – *Хаширање* - сите хашеви се единствени, па дури и мала промена во влезот ќе резултира во различен хаш. Истиот хаш се произведува само со хаширање на идентичен влез.
  – *Доказ за постоење* - сите хашеви се запишуваат во блок-веригата со создавање на запис дека хашот постоел во дадено време. Секој може да го потврди записот, но никој не може да ја види неговата содржина. Со цел сопствениците на патентот да го докажат постоењето на тој документ во дадено време, тие треба повторно да хашираат идентична копија. Овој процес може да им помогне на иноваторите да ја заштитат нивната работа со складирање на хашот од нивниот патент во блок-веригата.

## 6.4 Анализа на можностите за подобрување на блок-веригите

Бидејќи процесот на верификација на секоја трансакција е многу бавен, има потреба од воведување на третата главна иновација во блок-веригата наречена скалирање во блок-веригата. Скалираната блок-верига го прави процесот побрз преку истражување на:

– колку потврди се потребни да се валидира една трансакција и да се подели работата ефикасно и

– границата на износот на трансакции кои мрежата на блок-веригата може да ги процесира.

Оваа модификација, т.е. скалирањето, не ја загрозува безбедноста на блок-веригата. Скалираната блок-верига се очекува да биде доволно брза да влијае на IoT и да биде во паралела со главните посредници при плаќањето (на пример VISA и PayPal) во светот на банкарството [19].

### 6.4.1 Проблемот на двојно трошење

Скалираната блок-верига го прави процесот побрз без да влијае на безбедноста на мрежата. Во процесот на рудање, секој 10 минути се прави нов блок кој се вклучува во блок-веригата. Кога трансакцијата е вклучена во блок што се дистрибуира во мрежата на блок-веригата, таа трансакција се минира на длабочина од еден блок (една потврда за трансакцијата). Со секој нов блок во блок-веригата, длабочината на постојаните блокови се зголемува за еден. Трансакцијата е потврдена кога длабочината на блокот е на одредено ниво (шест е вообичаен број на потврди) [44].

Двојно трошење значи користење на исти биткоини повеќе од еднаш. Откако еднаш ќе се потврди трансакцијата, невозможно е да се троши двојно. Веројатноста за успешно двојно трошење се пресметува според Пуасонова распределба [83]:

$$P = 1 - \sum_{k=0}^{z} \frac{\lambda^k e^{-\lambda}}{k!}(1 - (\frac{q}{p})^{z-k})$$

каде

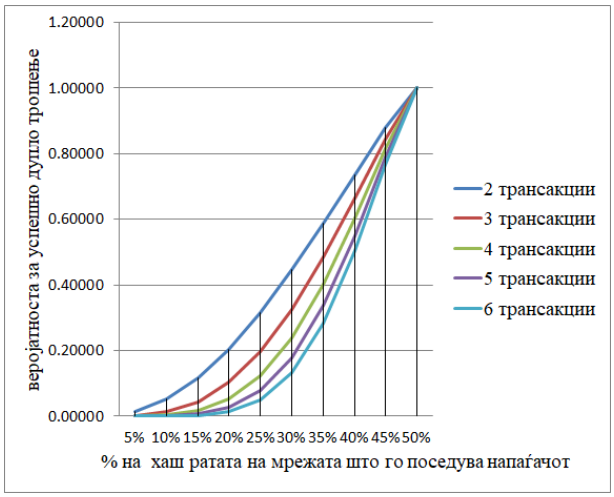
– $\lambda = \frac{zq}{p}$ е средна вредност;

– $z$ број на потврди на трансакцијата - број на блокови со кои "искреniот" јазол (јазол кој што се однесува на начинот на кој очекуваме да се однесуваат јазлите т.е. не се обидува да ја модифицира историјата, правилно ги опслужува блоковите и трансакциите, правилно ги пренесува пораки, податоци и сл.) има предност пред напаѓачот;

– $q$ е веројатност напаѓачот да го најде следниот блок (хаш ратата на напаѓачот);

– $p = 1-q$ е веројатноста искреniот јазол да го најде следниот блок.

На Слика 6.13 и Табела 6.2 и претставена е веројатноста за успешно двојно трошење како функција од хаш ратата на напаѓачот $q$, за различни вредности на бројот на потврди $n$ [39].




Слика 6.13: Веројатност за успешно двојно трошење

Бројот $6$ е избран како претпоставка дека напаѓачот не може да поседува повеќе од $10\%$ од целата хаш рата и ризикот од $0.1\%$ или помалку може да се прифати [97]. Но, на пример, ако хаш ратата е околу $40\%$, тогаш се потребни 90 потврди за да има помалку од $0.1\%$ шанса за успешен напад (нападот за двојно трошење на трансакцијата е наречен "алтернативен историски напад").

Табела 6.2: Веројатност за успешно двојно трошење

| Хаш рата на напаѓачот во % | 2 потвр. | 3 потвр. | 4 потвр. | 5 потвр. | 6 потвр. |
|---|---|---|---|---|---|
| 5% | 0.01265 | 0.00164 | 0.00022 | 0.00003 | 0.00004 |
| 10% | 0.05098 | 0.01317 | 0.00346 | 0.00091 | 0.00024 |
| 15% | 0.11504 | 0.04423 | 0.01725 | 0.00678 | 0.00268 |
| 20% | 0.20393 | 0.10324 | 0.05300 | 0.02742 | 0.01425 |
| 25% | 0.31544 | 0.19612 | 0.12351 | 0.07836 | 0.04994 |
| 30% | 0.44572 | 0.32458 | 0.23913 | 0.17735 | 0.13211 |
| 35% | 0.58881 | 0.48446 | 0.40251 | 0.33637 | 0.28217 |
| 40% | 0.73640 | 0.66417 | 0.60340 | 0.55063 | 0.50398 |
| 45% | 0.87772 | 0.84440 | 0.81561 | 0.78979 | 0.76611 |
| 50% | 1 | 1 | 1 | 1 | 1 |

### 6.4.2 Како системот на блок-веригата се справува со проблемот на двојно трошење?

Следниот пример го објаснува процесот на справувањето со проблемот на двојно трошење:

– Нека А има еден биткоин и сака да го испрати на Б. Оваа трансакција (наречена Т1) оди во базенот на непотврдени трансакции и чека да биде потврдена.

– Во исто време А испраќа еден биткоин на В. Оваа трансакција (наречена Т2) исто така влегува во базенот и чека потврда.

– Нека првата трансакција Т1 се извлече од базенот на непотврдени трансакции. Пред трансакцијата да биде вклучена во блок-веригата, се проверува нејзината валидност. Оваа трансакција ќе биде валидна со оглед на тоа што А има еден биткоин и истиот е вметнат во блок-веригата. Сега, трансакцијата Т2 е извлечена од базенот, но не е валидна (бидејќи А нема повеќе биткоини) и нема да биде потврдена.

– Ако трансакциите Т1 и Т2 се потврдуваат истовремено, тогаш блок-веригата има две гранки и при тоа првата трансакција која ќе стигне до следниот блок на потврди ќе биде потврдена.

– Ако T1 и T2 го стигнат следниот блок истовремено, трката за следниот блок продолжува.

– Затоа се препорачува да се почекаат 6 потврди за завршување на потврдувањето на трансакцијата. Како што можеме да видиме во Табела 6.13, многу е неверојатно дека трансакциите ќе стигнат до следниот блок истовремено повеќе од 6 пати. Значи, на крајот само една трансакција ќе биде потврдена.

"Вилушка" (fork) во блок-веригата се создава кога оригиналниот код на блок-веригата се ажурира, но само некои од јазлите на блок-веригата го прифаќаат ажурирањето. Всушност, тоа е промена на протоколот на блок-верига-та, што го користи софтверот за да одлучи дали е валидна трансакцијата или не. На овој начин, оригиналната блок-верига останува иста и ажурираните јазли се разделуваат од оригиналната блок-верига и создаваат нова блок-верига. Алтернативниот историски напад е 100% веројатно да успее ако напаѓачот има контрола над повеќе од половина од мрежната хаш рата. Во тој случај, напаѓачот може да продолжи да се движи по неговата приватна "вилушка" сè до моментот кога гранката ќе стане подолга од "вилушката" изградена од "искрената" мрежа, затоа што тој сега може да генерира блокови побрзо од другите учесници на мрежата [32].

### 6.4.3 Скалирана блок-верига

Накамото воведе ограничување на големината на блокот од 1MB за секој блок во јавната блок-верига. Ова е безбедносна мерка, па така секој блок со големина над таа граница веднаш ќе биде одбиен од $P2P$ мрежата. Ова ограничување ги успорува трансакциите и не може да биде во чекор со брзината на плаќање со другите валути и кредитни картички.

Во Табела 6.3 даден е преглед на просечниот број на трансакции направени со криптовалути и со стандардна кредитна картичка [42]. Биткоинот има ограничување од 3 до 4 трансакции во секунда (иако теоретски извршува до 7, но тој број никогаш не се постигнува). Ова не е ситуацијата со приватната блок-верига. Таа може да достигне извршување на над 1000 трансакции во секунда, затоа што секој јазол на мрежата во приватната блок-верига користи високо квалитетен компјутер со силна интернет врска.

Постојат неколку решенија за подобрување на скалабилноста на блок-веригата кои што веќе се или ќе бидат имплементирани. Некои од најглавните решенија за скалабилноста анализирани во [75] се:

Табела 6.3: Биткоин и Етериум наспроти PayPal и Visa трансакции во секунда

| Валута | Трансакции во секунда |
|---|---|
| Bitcoin | 3 до 4 трансакции во секунда |
| Etherium | 20 трансакции во секунда |
| PayPal | 193 трансакции просечно |
| Visa | 1667 трансакции во секунда |

– “Одделен сведок” (Segwit)

– Зголемување на големината на блокот

– “Делење на парчиња” (Sharding)

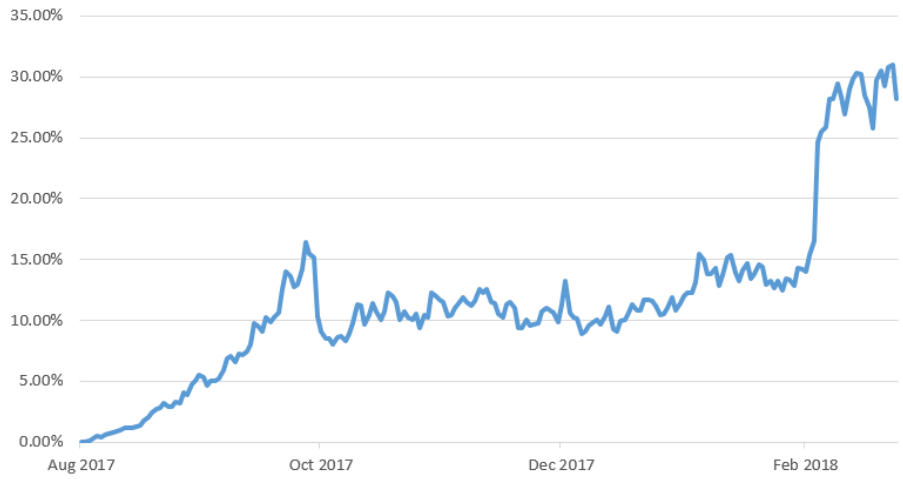
– “Доказ на удел” (Proof of Stake)

*“Одделен сведок” (Segwit)* е алтернативно решение за скалабилноста на блок-веригата, преку зголемување на бројот на трансакции во блок, без зголемување на големината на блокот. “Одделен сведок” помага да се зголеми просторот за нови трансакции со отстранување на податоците за потписите од биткоин трансакциите. Всушност, идејата во “одделен сведок” е да се отстранат дигиталните потписи и истите да се складираат надвор од основниот блок на трансакцијата. На овој начин “валидациониот“ дел е одвоен од “ефективниот“ дел на трансакцијата и повеќе трансакции можат да се вклопат во блокот без да се зголеми големината на блокот [37].

Дигиталниот потпис опфаќа 65 проценти од трансакцијата. Ако се отстрани, ќе се ослободи повеќе простор во биткоин блокот за повеќе трансакции. На овој начин се воведува и нова единица за големината на трансакциите. Со “одделен сведок”, трансакцијата е поделена на два дела: податоци кои што не се сведоци (и кои мора да бидат зачувани на блокот како порано) и податоци за сведоци (кои се преместуваат во продолжениот блок). Секој бајт во делот на податоците што не се сведоци се брои како 4 WU (weight units), додека бајтите во делот на податоци за сведоци се бројат само по 1 WU. Максималниот капацитет на блокот на овој начин изнесува 4 kWU, што би одговарало на старата максимална големина на блок од 1MB кога не би се користел “одделен сведок”. Надградбата со “одделен сведок” на биткоин

протоколот се појавува во август 2017 година. Сега за сега, новиот формат го користат околу 30 проценти од сите трансакции [37].

Согласно истражувањата на BitMEX процентот на трансакции кои го користат "одделен сведок" е даден на Слика 6.14.




Слика 6.14: Процент на трансакции кои користат "одделен сведок"

*Зголемување на големината на блокот*. Во биткоин системот големината на блокот е ограничена на максимум 1MB. Постојат неколку аргументи за и против зголемување на големината на блоковите. Главен аргумент против зголемувањето на големината на блокот е дека тоа ќе предизвика зголемена централизација.

Секој поврзан компјутер на биткоин мрежата се нарекува јазол. Постојат два вида јазли во биткоин блок-веригата: целосни јазли и делумни јазли. Целосните јазли го потврдуваат секој блок и секоја трансакција. Затоа, целосните јазли мора да складираат копија на комплетната главната книга на блок-веригата (повеќе од 165 GB). Делумните јазли не ја чуваат целата книга. Тие користат едноставен мод за верификација на плаќање (SPV), кој бара да се преземе само копијата на заглавијата на блоковите на најдолгиот ланец што се добива следејќи го доказот за работа (PoW) [83].

Од друга страна, рударот создава блокови во блок-веригата кои што ги чуваат јазлите. Биткоин мрежата се одржува со приближно 10.000 целосни јазли, додека бројот на рудари се проценува на околу 100.000. Зголемувањето на големината на блокот прави целосните јазли да бидат поскапи за ракува-

ње. Ова доведува до помалку хашери да работат со целосни јазли, со што централизираните ентитети би имале поголема моќ, што ја ослабува вредноста на биткоините. Рударите имаат корист од зголемувањето на големината на блокот, бидејќи зголемената големина на блок значи повеќе трансакции по блок. Но, ова ќе го зголеми износот на трошоците за трансакција што секој рудар може да ги направи од рударењето на блокот.

*"Делење на парчиња" (Sarding)* Еден од најголемите проблеми за криптовалутите е брзината на верификација на трансакциите. Секој целосен јазол во мрежата на блок-веригите ја чува целата главна книга на трансакции. Методот делење на парчиња има за цел да ја подели трансакцијата на делови, а потоа да ги рашири деловите меѓу мрежата така што јазлите да работаат на одделни делови од трансакцијата. На овој начин, вкупното време за потврдување на трансакцијата е намалено.

Нормален блок во блок-веригата има заглавие на блокот и тело што ги содржи сите трансакции во блокот. Коренот на Меркле (хаш на сите хашови од сите трансакции во блок) на сите трансакции се наоѓа во заглавието на блокот. Методот делење на парчиња го менува ова со помош на две нивоа на интеракција.

Првото ниво се состои од трансакциска група која е поделена на заглавие на трансакцискатa група и тело. Секој дел има своја група на трансакции. Хедерот на трансакцискатa група е поделен на лев и десен дел. Левиот дел содржи: ID на делот, корен пред состојба (состојба на коренот на делот, пред да се додадат трансакциите), корен после состојба (состојба на коренот на делот после додадените трансакции) и коренот за прием (корен за прием по додавањето на сите трансакции во делот). Десниот дел е полн со случајно избрани валидатори кои ги верификуваат трансакциите во делот. Телото на трансакцискатa група ги има ID на сите трансакции во делот.

Второто ниво е нормална блок-верига, но сега има два основни корени: корен на состојбата (кој ја претставува целата состојба која е поделена на делови) и коренот на трансакцискатa група (ги содржи сите групи на трансакции во тој блок). Всушност, ова ниво е како едноставна блок-верига, која прифаќа трансакциски групи наместо трансакции.

Со делењето на парчиња, многу паралелни трансакции можат да се случат во исто време и затоа на овој начин перформансите се подобруваат т.е. доаѓа до зголемување на спроводливоста (максималната стапка со која се процесираат трансакции) и намалување на латентноста (времето за потврдување дека трансакцијата е вклучена) во мрежата на блок-веригата [24].

Zilliqa, новата платформа на блок-вериги, базирана на технологијата на

делење на парчиња која го решава проблемот со скалабилноста со кои се соочуваат сегашните платформи за блок-вериги како што се Биткоин и Етериум, го објави јавно тестот на нејзината мрежа. Во Табела 6.4 е дадена споредбата на можностите за обработка на трансакциите во Етериум и Zilliqa.

Табела 6.4: Споредба на можностите на процесирање на трансакциите во Етериум и Zilliqa

| | Број на целосни јазли | Број на трансакции во секунда |
|---|---|---|
| Ethereum | 25000 | $15 - 20$ |
| Zilliqa | 1800 | 1218 |

Ако бројот на јазли во Zilliqa се зголеми двојно на 3600, спроводливоста ќе се зголеми на 2488 трансакции во секунда [45] [101] [31]. Од Табела 6.4 јасно е дека Zilliqa дури и со помал број јазли е во состојба да процесира многу повеќе трансакции во секунда од Етериум или Биткоин.

*"Доказ на удел"(Proof of stake)*. Повеќето криптовалути го следат протоколот на доказот за работа (proof of work), што значи дека рударите рударат криптовалути со решавање на крипто-загатки. Во протоколот на доказот на удел, постојат валидатори наместо рудари. Валидаторот треба да инвестира некои од неговите средства како удел. Потоа, валидаторот започнува да ги валидира блоковите на следниов начин: ако види блок за кој смета дека може да се додаде во блок-веригата, тој го потврдува блокот со тоа што ќе се обложи на него. Ако блокот се придружи на блок-веригата, валидаторот ќе добие награда пропорционална со вложениот удел. Ако валидаторот се обложи на злонамерен блок, уделот што тој го инвестирал ќе биде одземен од него. Етериум имплементира протокол за доказ на удел со користење на Каспер алгоритам за консензус [24]. Предноста на користењето на протоколот доказ на удел наместо доказ за работа е тоа што се користи значително помалку енергија и како резултат на тоа е поекономичен.

Во Табела 6.5 дадена е споредба на некои параметри помеѓу протоколите доказ за работа и доказ на удел.

Табела 6.5: Споредба на протоколите доказ за работа и доказ на удел

| | Доказ за работа | Доказ на удел |
|---|---|---|
| Потрошувачка на енергија | Голема | Мала |
| Потребни алатки | Опрема за рударење | Не е потребна опрема |
| Безбедност | Висока | Не тестирано |
| Децентрализираност vs. Централизираност | Тенденција кон централизираност | Корисниците можат да ги контролираат токените |

## 6.5 Безбедни масивни податоци и IoT во блок-веригите

Како што кажавме претходно блок-веригата претставува дистрибуирана книга што евидентира трансакции во блокови што формираат ланец. Овој ланец е безбеден, непроменлив и транспарентен. Затоа, решенијата што ги користи технологијата на блок-веригите може да бидат столб на системите за обработка и процесирање на податоците во рамките на организациите. Интегрирањето на блок-веригите со *масивните податоци* има многу предности бидејќи овозможува подобро управување со огромните количини и разновидност на информациите. Масивните податоци и блок-веригите се корелативни: масивните податоци имаат капацитети за обработка што можат да се справат со комплексноста на блок-веригата и нејзината голема експанзија и обратно [27]. Имплементацијата на блок-веригите во масивните податоци потврдува дека податоците се точни и безбедни и споделувањето на податоците ќе стане поедноставно.

### 6.5.1 Што се масивни податоци?

*Масивни податоци (Big Data)* е израз кој се користи за собирање на масивни и комплексни множества на податоци. На пример, Фејсбук од своето основање има собрано 300 петабајти (PB) лични податоци, како што се: зачувани записи, испратени пораки, објавени видеа, податоци за трансакции од онлајн трансакции и купување меѓу многу други извори на масивни податоци.

Тешко е да се обработуваат и анализираат масивните податоци со тради-

ционалните техники, бидејќи станува збор за голем обем на структурирани и неструктурирани податоци. Но, сепак денес скоро секоја компанија користи масивни податоци за да ја постигне конкурентната предност на пазарот. Имајќи го ова во предвид, алатките за обработка и анализа на масивните податоци се најповолниот избор на компаниите во однос на трошоците и другите придобивки. Различниот вид на податоци, IoT и неструктурирани извори можат да се чуваат и обработуваат со алатките како што се Hadoop, Plotly, Bokeh, итн. Студиите покажуваат дека околу 2.5 кватрилиони бајти податоци се генерираат секој ден.

И покрај ваквиот придонес за ефикасно управување со масивни количини на податоци, постои поголема загриженост за приватноста на корисниците и безбедноста на масивните податоци. Постојат многу примери кои го покажуваат тоа, како што се масивниот научен експеримент спроведен од Фејсбук, без да ги информира своите корисници експлицитно или владата на САД честопати беше напаѓана заради набљудување на нејзините граѓани без нивно директно одобрение. Денес бизнисите се внимателни во прифаќањето на масивните податоци, со цел да се обезбеди нивната тајност и приватност [38].

### 6.5.2 Придобивките од примената на блок-веригите во масивните податоци

Како што спомнавме, без оглед на разновидноста, брзината и обемот на податоците, алатките за обработка на масивните податоци можат да ги обработуваат податоците. Исто така кажавме дека блок-веригата овозможува транспарентност и едноставност во процесирањето во секоја област. Но, важноста на масивните податоци и развојот на технологијата на блок-веригите во последните години овозможува напуштање на старите структури за обработка на информации и обработка на бизнис трансакциите, т.е. се прави поврзување на овие две дисциплини. Анализа на придобивките од примената на блок-веригите во масивните податоци е направена во трудот [82].

Некои од придобивките од користењето на технологијата на блок-веригите во аналитиката на масивните податоци [49] се:

- намалување на трошоците (значително ги намалува трошоците за складирање);
- зголемување на следливоста (секој производ или документ има ”дигитална лозинка” што обезбедува следење на потеклото и патувањето на производот);

– подобрување на квалитетот на податоците (податоците се комплетни и структурирани);

– олеснување на пристапот до податоците (корисниците од различни одделі можат да пристапат до податоците за анализа на процесот и ова го скратува временскиот циклус на пристап и анализа на податоци);

– подобрување на безбедноста (системот е децентрализиран и транспарентен, така што ризикот од лажни активности се намалува).

**Анализа на неколку имплементации на технологијата на блок-веригите во масивните податоци**

– **Забрзување во индустријата за финансиски услуги**

Поврзувањето на блок-веригите и масивните податоци во финансиските институции ќе овозможи да се процени ризикот и да се идентификуваат сомнителните обрасци во реално време. Користењето на блок-веригите како средство за спроведување на трансакции ќе помогне да се заштитат банките и нивните клиенти од измама, да се забрза процесот на трансакции и да се намалат трошоците за трансфери на пари [33].

На пример, во последните неколку години, со цел да се поедностават трансферите на пари меѓу банкарските сметки со користење на технологијата на блок-веригите, организација на 47 јапонски банки се приклучи на блок-верига стартап наречен Ripple, а со цел е да се изврши трансфер во реално време со пониска цена. Традиционалните трансфери во реално време се скапи, како резултат на потенцијалните фактори на ризик, како што е двојното трошење кое што може да се избегне со технологијата на блок-веригите. Анализите на масивните податоци заедно со блок-веригите можат да ги идентификуваат ризичните трансакции побрзо од тековните. Ова ги намалува трошоците со трансакции во реално време [54].

– **Безбедност во индустријата надвор од банкарството**

Со цел да се справат со податоците и да го спречат хакерството и одливот на податоци, здравствената заштита, јавната администрација, претпријатијата започнаа со користење на блок-веригите.

На пример:

  - **Здравство**: Проект во лабораторијата на МИТ Медија, познат како Медрек, започнува да креира систем на блок-вериги кој им дава приоритет на агенцијата на пациенти и кој им дозволува на пациентите да ги споделат своите записи. Со цел да се олесни делењето на дозволите за пристап до податоците, Медрек го користи приватниот ланец Етериум [4].
  - **Осигурување**: Контролирана главна полиса од Велика Британија и три локални полиси од САД, Сингапур и Кенија запишани се во паметен договор кој овозможува заеднички увид во податоците и документацијата за полисите во реално време. Блок-веригата обезбедува јасност во покриеноста и плаќањето премија на локално и главно ниво. Исто така, овозможува автоматско предупредување до учесниците во мрежата после извршување на плаќањето [46].

- **Следење на снабдувачките синџири**

  Стоката вклучена во снабдувачкиот синџир се додава во блок-веригата и се користи мобилна апликација за следење на состојбата на стоките додека се испраќаат. Придобивките од користењето на блок-веригата во синџирот на снабдување се следните [46]:

  - намалување или елиминирање на измамите и грешките;
  - подобрување на управувањето со залихите;
  - минимизирање на трошоците за достава;
  - намалување на одложувањата поради документацијата;
  - побрзо идентификување на проблемите;
  - зголемена потрошувачка и партнерска доверба.

На пример, со цел да се подобри безбедноста на храната преку зголемување на набљудувањето на производите од местото на потекло до времето кога се продава на потрошувачот, Walmart користи технологија на блок-веригите. На овој начин, корисниците добиваат веродостоен увид за потеклото на храната. Добивањето непроменливи, веродостојни и следливи податоци е од големо значење за успехот во работењето на Walmart, затоа што обработува 40 петабајти (PB) податоци секој ден [38].

**Примери за подобрување на масивните податоци користејќи блок-верига**

Во овој дел ќе презентираме неколку примери за имплементација на блок-веригите во масивните податоци.

– Пример 1: GOLEM
GOLEM мрежата е децентрализирана мрежа на компјутери на која што секој може да пристапи, што ја користи компјутерската моќност на корисниците на мрежата за да го направи суперкомпјутерот функционален. Лаптопите, десктопите и центрите за податоци се само неколку примери на уреди кои придонесуваат за пресметување на моќта во мрежата. Мрежата Golem има за цел да создаде децентрализирана економија на споделување на компјутерската моќ, каде секој може да заработи пари "изнајмувајќи" од својата компјутерска моќ или развивање и продавање софтвер. Всушност, таа има за цел да ја направи компјутерската моќ многу поевтина отколку што е денес, а со тоа и намалување на трошоците за нејзино користење [23].

– Пример 2: PATH
Платформата PATH се стреми да ги обедини блок-веригите и масивните податоци, а со цел да им овозможува на корисниците да го "изнајмуваат" нивниот екстра опсег. На пример, една компанија сака да има увид како нејзината веб страна е презентирана во јавноста или пак колкава покриеност има нејзината мрежа во одреден период на денот и.т.н. На компјутерите на компаниите се инсталираат "јазли за рударење на Path" и тие работаат пасивно во позадина - заработуваат токени, а со цел да ги обезбедат работните увиди на компаниите, спомнати претходно. Ова е само една од многуте платформи на блок-веригите која им овозможува на компаниите да ја искористат огромната потреба на компаниите да ги подобрат множествата податоци кои што ги анализираат и да ги користат за своите производи и услуги. Всушност, вистинската вредност на податоците во блок-веригата е квалитетот на податоците и како тие се чуваат во главната книга. Така, оваа платформа заедно со другите платформи на блок-веригите им помагаат на компаниите да ги подобрат податоците и стануваат посредници меѓу компаниите и корисниците [52].

– Пример 3: ЕСТОНИЈА
ЕСТОНИЈА се стреми да биде најнапредното дигитално општество,

кое се заснова на блок-веригите. Таа ја користи инфраструктурата за потпишување без клуч (Keyless Signature Infrastructure) (КСИ) заснована на технологијата на блок-веригите. КСИ се користи за складирање на јавни податоци во блок-веригата, дизајнирана да обезбеди скалабилен дигитален потпис, со користење на криптографска хаш функција. Користејќи го ова, владата може да набљудува какви било промени во базата на податоци, со што се осигурава дека податоците се транспарентни. Од ова произлегуваат следните две придобивки: намалување на надворешните фалсификувани/изменети записи и отежнување на мешањето на неовластени владини службеници во информациите и податоците [38].

– Пример 4: SKRY
Финансиските институции, полицијата и биткоин компаниите можат да детектираат сомнителни активности користејќи го првиот производ на SKRY. Тој може да идентификува и легални и нелегални субјекти, што овозможува следење на регулативите и истрагите на интернет криминалот, како што се изнудување на откуп на софтвер, финансирање на тероризам или трговија со дрога на темна мрежа.

### 6.5.3 Интернет на нештата и блок-веригите

Примената на блок-веригите во интернет на нештата им овозможува на IoT уредите да учествуваат во трансакциите на блок-веригите и да пронаоѓаат нови стилови на дигитални интеракции. Оваа технологија ќе обезбеди едноставна инфраструктура за уредите директно и безбедно да пренесуваат податоци или пари користејќи паметни договори.

#### Предизвици за безбеден IoT модел

Тековниот IoT екосистем се заснова на централизиран - клиент/сервер модел, каде што сите уреди се идентификуваат, автентицираат и се поврзуваат преку облак сервери. Оваа структура е најмасивниот предизвик за безбедноста на IoT. Дури и ако уредите се на кратко растојание, врската помеѓу нив ќе мора да помине низ облакот. Овие модели продолжуваат да се користат во денешните IoT мрежи, иако тие не би можеле да се справат со растечките потреби на гигантскиот IoT екосистем во иднина. Трошоците за постојните IoT решенија, кои користат централизиран модел, се уште една голема пречка

за идниот раст на ИТ мрежата. Облаците, масивните фарми за сервери и мрежните уреди имаат поголеми трошоци за инфраструктурата и одржувањето. Исто така, за да се заштитат уредите и платформите на IoT од физичко нарушување, потребни се нови безбедносни технологии. Покрај тоа, многу уреди во IoT системот користат едноставни оперативни системи и процесори кои не поддржуваат напредни безбедносни системи [43].

Блок-веригите во IoT може да се користат за следење на поврзаните уреди, обработка на трансакциите и координација помеѓу уредите. Децентрализираната природа на технологијата на блок-веригите ќе произведе пофлексибилен систем за водење на уредите, а со криптографските алгоритми податоците ќе бидат повеќе приватни. Главната книга на блок-веригите не може да биде изманипулирана од злонамерни корисници бидејќи не постои на една единствена локација, што значи дека е непроменлива. Исто така, блок-веригата овозможува безбедно $P2P$ испраќање на пораки помеѓу IoT уредите. Можностите на блок-веригата, како што е децентрализацијата, автономијата и доверливоста, ја прават оваа технологија идеална за фундаментална компонента на IoT решенијата [43].

### 6.5.4 Пример на IoT и осигурување

Модели на осигурување засновано врз употреба (Usage-based insurance models-UBI) се некои од новите производи што ќе развијат точни актуарски модели кои се потребни за осигурителните компании. На пример, во авто осигурувањето, можеме да користиме шифрирани податоци за време на возење, растојанија, забрзување, сопирање и други информации кои се користат за идентификација на возачи со висок ризик и валидација на информациите вклучени во апликациите. На овој начин, може да им обезбедиме на потрошувачите поголема контрола над нивните премии. Прашањето е како да се управува со огромниот обем на податоци поради комуникацијата помеѓу милиони уреди. Масивните сложени мрежи може да бидат управувани од блок-веригата со уреди кои комуницираат меѓусебно засновано на $P2P$ мрежата, сигурно и прецизно, наместо да се гради скап центар за податоци. Овој тип на уреди за управување е поевтин од централниот модел на податоци [48].

# Глава 7

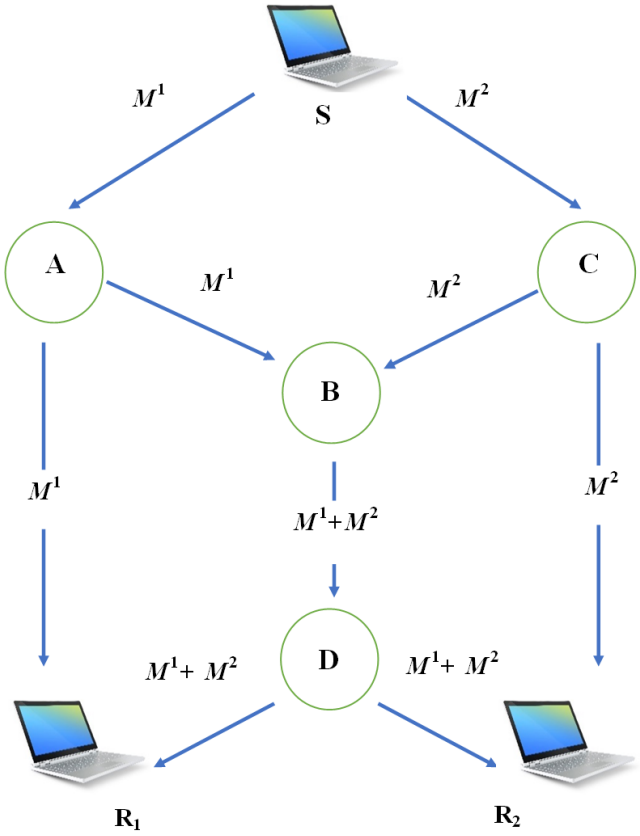
# Мрежно кодирање базирано на квазигрупи

Во првата глава ја објаснивме примената на квазигрупите и квазигрупните трансформации во дизајнирањето на криптографски примитиви, кодови за откривање и кодови за поправање на грешки. Ги наведовме и причините за тоа, како што се: структурата на квазигрупите, нивниот голем број, својствата на квазигрупските трансформации и други. Сега, се поставува прашање дали квазигрупите може да најдат примена во блок-веригите. Со цел да се обезбеди побрз пренос во една мрежа (па и во блок-веригите), се користи таканаречено мрежно кодирање. Во овој дел, ќе дадеме една примена на квазигрупните трансформации во овој вид на кодирање.

## 7.1 Мрежно кодирање

Мрежно кодирање е техника која ја подобрува мрежната моќност и обезбедува поголема сигурност. Пренесените податоци се кодираат со цел да се зголеми спроводливоста, да се намалат одложувањата и да се направи мрежата поробусна. Во мрежното кодирање, во јазолот се користат алгебарски алгоритми за кодирање на неговите примени пораки во излезни пораки. Во крајните јазли, добиените пораки се декодираат и се пренесуваат до нивните дестинации. На овој начин, потребни се помалку преноси за пренесување на сите податоци, но за да се реализира ова е потребна поголема обработка во посредничките и терминалните јазли. Најпопуларното мрежно кодирање е линеарното мрежно кодирање. Тоа претставува математичка техника која ја подобрува пропусната моќ, ефикасноста и скалабилноста на мрежата, како и

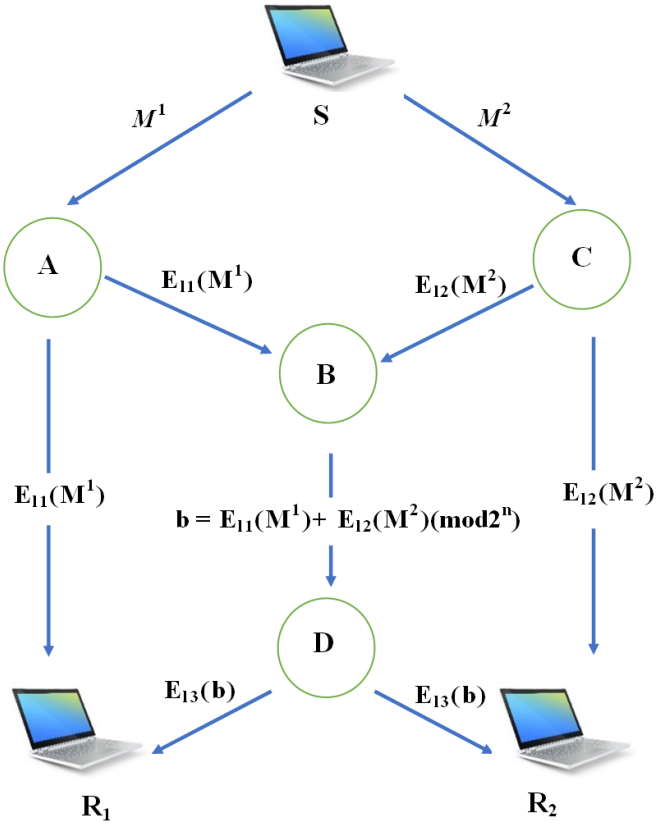
отпорноста од напади и прислушкување. Во линеарното мрежно кодирање наместо едноставен пренос на пакети со информации, јазлите генерираат нови пакети кои се линеарни комбинации на претходно примени пакети, множејќи ги со коефициенти избрани од конечно поле. Јазлите ги добиваат овие кодирани пораки и ги собираат во матрица. Оригиналните пораки можат да се добијат од матрицата со методот на Гаусова елиминација [5], [63], [58].




Слика 7.1: Мрежно кодирање во “Пеперутка”

Мрежата “пеперутка” е форма на мрежна топологија која може да се користи за поврзување на различни јазли во мултипроцесорски систем [20]. Оваа мрежа често се користи за да се илустрира како мрежното кодирање може да го заобиколи рутирањето (насочувањето). Двата информациски извори ги емитираат пораките $M^1$ и $M^2$ кои мора да бидат пренесени до два различни

јазли $R_1$ и $R_2$. Секој јазол може да носи само една порака. Централниот линк може да ги носи само $M^1$ или $M^2$, но не и двете. Ако претпоставиме дека ја испраќаме пораката $M^1$ низ центарот, тогаш $R_1$ би ја добил $M^1$ два пати и нема да ја добие $M^2$ воопшто. Испраќањето на $M^2$, повлекува сличен проблем за $R_2$. Значи, рутирањето е недоволно затоа што ниту една шема за рутирање не може да ги пренесе $M^1$ и $M^2$ истовремено до двете дестинации. На Слика 7.1 е прикажана мрежата "пеперутка" и како пораките $M^1$ и $M^2$ можат да се пренесат до крајните јазли $R_1$ и $R_2$ истовремено т.ш. ги кодираме пораките $M^1$ и $M^2$ користејќи $M^1 + M^2$.




Слика 7.2: Мрежно кодирање базирано на квазигрупи

Сложеноста на декодирањето со користење на Гаусов метод на елиминација, ни ја даде идејата за конструирање нов алгоритам за мрежно коди-

рање базиран на квазигрупи. Овој алгоритам е поедноставен и побрз отколку алгоритмите што се користат во линеарното мрежно кодирање. Идејата за мрежното кодирање базирано на квазигрупи е претставено на едноставна мрежа на Слика 7.2. Во овој алгоритам користиме:

– Квазигрупа $(Q, *)$ од ред $2^n$ заедно со нејзината парастрофа $\backslash$ .

– $E_l$ - трансформација со лидер $l$ за кодирање и шифрирање пораки и $D_l$ -трансформација со истиот лидер $l$ за декодирање и дешифрирање.

*Алгоритам за мрежно кодирање*

Во изворот $S$, конечен број на пораки се генерираат и се испраќаат до другите јазли на мрежата. Нека, $M^1$ и $M^2$ се оригинални пораки од $m$ симболи од $Q$.
**Чекор 1.** Јазолот $A$ ја прима пораката $M^1$ и на неа применува $E$ - трансформација со даден лидер $l_1$и ја испраќа кодираната порака $E_{l_1}(M^1)$ до средниот јазол $B$ и до крајниот јазол $R_1$. На исти начин јазолот $C$ ја прима пораката $M^2$ и на неа применува $E$ - трансформација со даден лидер $l_2$ и ја испраќа кодираната порака $E_{l_2}(M^2)$ до средниот јазол $B$ и до јазолот $R_2$.
**Чекор 2.** Во средниот јазол $B$ се прави собирање по модул $2^n$ на двете примени пораки $E_{l_1}(M^1)$ и $E_{l_2}(M^2)$ и се испраќа добиената порака $b$ до јазолот $D$.
**Чекор 3.** Јазолот $D$ ја испраќа кодираната порака $E_{l_3}(b)$ до крајните јазли $R_1$ и $R_2$. На тој начин, јазолот $R_1$ ја прима $E_{l_1}(M^1)$ и $E_{l_3}(b)$, а јазолот $R_2$ ја прима $E_{l_2}(M^2)$ и $E_{l_3}(b)$.

Крајните јазли $R_1$ и $R_2$, ги декодираат оригиналните пораки $M^1$ и $M^2$ на следниот начин:
*Алгоритам за декодирање*

Во јазолот $R_1$:
Примените пораки се $E_{l_1}(M^1)$ и $E_{l_3}(b))$.
**Чекор 1.** Пораката $b$ е добиена со примена на инверзна трансформација $D_{l_3}$ на $E_{l_3}(b))$, т.е. $b = D_{l_3}(E_{l_3}(b))$
**Чекор 2.** Пораката $E_{l_2}(M^2)$ се добива со одземање по модул $2^n$ на $b$ и $E_{l_1}(M^1)$.
**Чекор 3.** Оригиналните пораки $M^1$ и $M^2$ се добиваат со примена на инверзни трансформации $D_{l_1}$ и $D_{l_2}$ на $E_{l_1}(M^1)$ и $E_{l_2}(M^2)$ соодветно, т.е. $M^1 = D_{l_1}(E_{l_1}$

$(M^1))$ и $M^2 = D_{l_2}(E_{l_2}(M^2))$.

Во јазолот $R_2$:
Примените пораки се $E_{l_2(M^2}$ и $E_{l_3}(b))$.
**Чекор 1.** Пораката $b$ е добиена со примена на инверзна трансформација $D_{l_3}$ на $E_{l_3}(b)$, т.е. $b = D_{l_3}(E_{l_3}(b))$
**Чекор 2.** Пораката $E_{l_1}(M^2)$ се добива со одземање по модул $2^n$ на $b$ и $E_{l_2}(M^1)$.
**Чекор 3.** Оригиналните пораки $M^1$ и $M^2$ се добиваат со примена на инверзни трансформации $D_{l_1}$ и $D_{l_2}$ на $E_{l_1}(M^1)$ и $E_{l_2}(M^2)$ соодветно, т.е. $M^1 = D_{l_1}(E_{l_1}$ $(M^1))$ и $M^2 = D_{l_2}(E_{l_2}(M^2))$.

Процесот на кодирање и декодирање со овој алгоритам кој користи квазигрупа од ред $2^2$ опишан е преку следниот пример:

Пример: Нека $Q = \{0, 1, 2, 3\}$. Ја земаме следната квазигрупна операција $*$ и соодветната операција $\setminus$ на $Q$.

| $*$ | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0 | 0 | 2 | 1 | 3 |
| 1 | 1 | 3 | 2 | 0 |
| 2 | 2 | 0 | 3 | 1 |
| 3 | 3 | 1 | 0 | 2 |

(7.1)

| $\setminus$ | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0 | 0 | 2 | 1 | 3 |
| 3 | 0 | 1 | 2 | 0 |
| 2 | 3 | 0 | 1 | 1 |
| 1 | 2 | 3 | 0 | 2 |

(7.2)

– Кодирање
Ако $M^1 = 1233$, $M^2 = 2323$, $l_1 = 1$, $l_2 = 2$ и $l_3 = 3$, тогаш:

– $A$ ја испраќа $E_1(1233) = 2032$ до $B$ и $R_1$,

  - $C$ ја испраќа $E_2(2323) = 0310$ до $B$ и $R_2$
  - $B$ ја прима $E_1(1233) = 2032$, $E_2(2323) = 0310$ и ја испраќа пораката $b = (2032 + 0310) \bmod 4 = 2302$ до $D$.
  - Потоа $D$ ја испраќа $E_3(2302) = 1002$ до $R_1$ и $R_2$.

- Декодирање
  $R_1$ го добива следното:

  - $E_1(1233) = 2032$
  - $E_3(2302) = 1002$

  Изворните пораки се добиваат со следните операции:

  - $b = D_3(1002) = 2302$
  - $E_2(2323) = (2302 - E_1(1233)) \bmod 4 = (2302–2032) \bmod 4 = 0310$
  - $M_1 = D_1(2032) = 1233$
  - $M_2 = D_2(0310) = 2323$

  $R_2$ го добива следното:

  - $E_2(2323) = 0310$
  - $E_3(2302) = 1002$

  Изворните пораки се добиваат со следните операции:

  - $b = D_3(1002) = 2302$
  - $E_1(1233) = (2302 - E_2(2323)) \bmod 4 = (2302–0310) \bmod 4 = 2032$
  - $M_1 = D_1(2032) = 1233$
  - $M_2 = D_2(0310) = 2323$

Овој нов алгоритам има неколку предности во однос на линеарното мрежно кодирање. Овде, декодирањето користи многу поедноставни операции отколку во линеарното мрежно кодирање. Имено, наместо да решаваме систем линеарни равенки користејќи Гаусов метод на елиминација, применуваме само $D$-трансформација. Затоа, нашиот алгоритам обезбедува побрз процес на декодирање. Исто така, заради веќе докажаните криптографски својства на користените квазигрупни трансформации, овој алгоритам обезбедува тајност на пренесените податоци [81].

# Глава 8

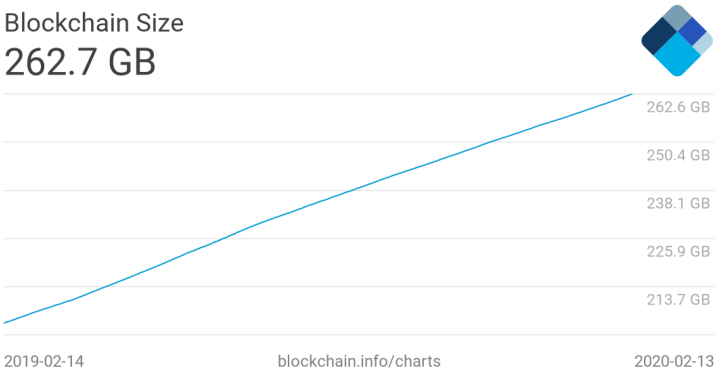
# Алгоритам за намалување на просторот во блок-веригите базиран на споделување на тајна

Како што спомнавме претходно, блок-веригата е дистрибуирана база на записи или јавна книга на сите временски запишани трансакции зачувани во сите компјутери во една $P2P$ мрежа. Всушност, блок-веригата е растечка листа на поврзани блокови [75]. Блоковите се состојат од валидни трансакции, временски печат и хаш покажувач како врска до претходниот блок во ланецот. Учесниците ја потврдуваат секоја трансакција во главната книга со консензус на мнозинството. Информациите што се внесуваат во главната книга никогаш не можат да се избришат. Главната книга на блок-веригата содржи запис за секоја трансакција некогаш направена [7].

Бидејќи секој јазол треба да ги чува сите трансакции, просторот за складирање брзо се зголемува и со тоа растат трошоците за складирање на целата мрежа на блок-веригата. Поради тоа, има потреба од промена на концептот на складирање во блок-веригата. На Слика 8.1 дадена е вкупната големина на сите заглавија на блокот и трансакциите во блок-веригата на биткоинот во текот на минатата година (2019).

Во [8] авторите предлагаат решавање на овој проблем со помош на мрежно кодирање. Тие го делат блокот со големина $G$ во $k$ пакети со еднаква должина. Овие $k$ пакети се кодираат во $n$ кодирани пакети користејќи линеарни комбинации на оригиналните пакети.

Концептот на *Дистрибуиран Простор за Складирање на блок-веригите* (Distributed Storage Blockchain или кратко DSB) е воведен во [94]. Во дистрибуираниот простор за складирање на блок-веригите трансакцискиот блок




Слика 8.1: Големина на Blockchain [21]

се шифрира со употреба на различни приватни клучеви и потоа се дистрибуира на подмножествата од јазли. Исто така, хаш вредностите и клучеви за шифрирање се чуваат меѓу јазлите од подмножеството во $P2P$ мрежата со помош на шемата за споделување на тајна. Дистрибуираниот простор за складирање на блок-веригите значително го намалува просторот за складирање во системот на блок-веригите. За да се подобри дистрибуираниот простор за складирање на блок-веригите, се предлага локална шема за споделување на тајна во [56].

Во [95] авторите ги подобруваат перформансите на блок-веригите и ги намалуваат трошоците за складирање со користење на техниките на теоретското кодирање.

Во овој дел, ќе предложиме на кој начин може да се намали просторот за складирање во блок-веригата со користење на шемата на Шамир за споделување на тајна. Прво ќе го објасниме концептот на работа на оваа шема.

## 8.1 Шема на Шамир за споделување на тајна

Во [100] Шамир го воведе концептот на *споделување на тајна* преку $(k, n)$ - шемата на прагот (threshold scheme). Неговиот модел се базира на полиноми. Според концептот на Шамир, шемата за споделување на тајна е да се дистрибуира тајната $S$ на група од $n$ учесници, така што секој учесник ќе добие еден дел од тајната. Познавањето на кој било $k$ или повеќе делови ја прави $S$ лесна за пресметување, но помалку од $k$ дела не ја реконструираат тајната порака

$S$. За да се сподели пораката $S$, случаен полином од степен $k$–1, $f(x) = a_0 + a_1x + a_2x^2 + \ldots + a_{k-1}x^k$–1 се избира на следниов начин:

- тајната порака $S$ е константниот член т.е. коефициентот $a_0$;
- $a_1, a_2, \ldots, a_{k-1}$ се случајно избрани коефициенти од конечно поле $F_p$, каде $p$ е прост број;
- $n$-те точки (делови) ($(i, f(i))$ за различни вредности на $i$ (на пример за $i = 1, 2, \ldots, n$) заедно со простиот број $p$, се дистрибуираат на $n$ учесници (јазли) [100] [9].
- секој член добива една точка $(i, f(i))$ и со кое било подмножество со $k$ од овие точки, коефициентите на полиномот можат да се најдат со користење на интерполација.

Шемата на Шамир за споделувње на тајна ($k$, $n$) има неколку предности. Некои од нив се:

- За фиксно $k$, можеме да направиме динамичко додавање или бришење на делчиња, без промена на другите делови;
- Можеме лесно да ја подобриме безбедноста без промена на тајната, конструирајќи само нови точки за учесниците, задржувајќи го истиот константен член на полиномот;
- Доколку е важна хиерархијата на учесниците, можеме да им дадеме на учесниците различен број парчиња според нивното ниво во хиерархијата.

## 8.2 Намалување на просторот за складирање

Во овој дел предлагаме едноставен алгоритам за намалување на просторот за складирање на блок-веригата врз основа на шемата на Шамир за споделување на тајна, опишано во претходниот дел. Алгоритамот е следен:

**Чекор 1**. Го делиме трансакциски блок $B$ на $k$ пакети со еднаква должина, означени со $b_0, b_1, \ldots, b_{k-1}$ , кои всушност се цели броеви.

(Сите следни аритметички операции се извршуваат во конечно поле $F_p$, каде што $p$ е прост број поголем од бројот на јазлите $n$ и $b_i$, за $1 \leqslant t \leqslant k$.)

**Чекор 2**. Конструираме полином $f(x)$ од степен $(k–1)$ со коефициенти $b_0, b_1,$
$\ldots, b_{k-1}$ т.е.

$$f(x) = b_0 + b_1 x + b_2 x^2 + \ldots + b_{k-1} x^{k-1}$$

**Чекор 3**. Се пресметуваат $n$-те делови $(i, f(i))$ за различни вредности на $i$ (на пример за $i = 1, 2, \ldots, n$) и се распределуваат до соодветните јазли на блок-веригата.

Секој јазол добива една точка $(i, f(i))$ и коефициентите на полиномот можат да се најдат знаејќи кој било $k$ од овие деловите, користејќи интерполација. Со ова, блокот ќе биде обновен бидејќи коефициентите на полиномот се делови од блокот. Значи, процесот на реконструкција се сведува на интерполација на полиномот.

Со овој алгоритам, секој јазол во системот на блок-веригата чува по еден дел, наместо цел блок, со што се намалува просторот за складирање, а истовремено и процесот на обновување е едноставен.

Со следниов пример ќе го прикажеме концептот на нашиот алгоритам.

**Пример 1.** *: Нека $B = 123416$ е еден блок. Го делиме на 3 парчиња со еднаква должина:*

$$b_0 = 12, b_1 = 34, b_2 = 16$$

*Земаме $p = 37$ и го формираме полиномот од втор степен*

$$f(x) = 12 + 34x + 16x^2$$

*Ако сакаме овие блокови да се споделат на 4 јазли, пресметуваме 4 делови $(i, f(i))$ за различни вредности на $i = 1, 2, 3, 4$. Ги добиваме точките $(1, 62)$, $(2, 144), (3, 258), (4, 404)$ и ги дистрибуираме овие точки до соодветните 4 јазли на блок-веригата.*

*Процес на реконструкција:*

За реконструкција на пораката $B$ доволни се 3 точки. На пример, ако ги знаеме точките: $(1, 62), (2, 144), (3, 258)$ ќе ги пресметаме Лагранжовите основни полиноми:

$$l_1 = \frac{x-2}{1-2} \cdot \frac{x-3}{1-3} = \frac{x-2}{-1} \cdot \frac{x-3}{-2} = \frac{x^2}{2} - \frac{5x}{2} + 3$$

$$l_2 = \frac{x-1}{2-1} \cdot \frac{x-3}{2-3} = \frac{x-2}{1} \cdot \frac{x-3}{-1} = -x^2 + 4x - 3$$

$$l_3 = \frac{x-1}{3-1} \cdot \frac{x-2}{3-2} = \frac{x-1}{2} \cdot \frac{x-2}{1} = \frac{x^2}{2} - \frac{3x}{2} + 1$$

Потоа, користејќи ги овие полиноми го добиваме оригиналниот полином:

$$f(x) = l_1 \cdot 62 + l_2 \cdot 144 + l_3 \cdot 258 =$$
$$= (\frac{x^2}{2} - \frac{5x}{2} + 3) \cdot 62 + (-x^2 + 4x - 3) \cdot 144 + (\frac{x^2}{2} - \frac{3x}{2} + 1) \cdot 258 =$$
$$= 12 + 34x + 16x^2$$

Од коефициентите на добиениот полином, ја реконструираме оригиналната порака $B = 123416$.

На овој начин, илустриравме нова и поедноставна идеја за намалување на трошоците за сместување во технологијата на блок-веригите заснована на шемата на Шамир за споделување на тајна. Резултатите од овој дел се објавени во [93].

# Заклучок

Во голем број истражувања, се покажа дека квазигрупите и квазигрупните трансформации може да најдат примена во теоријата на кодирање и криптографијата. Причини за тоа се: структурата на квазигрупите, нивниот голем број, својствата на квазигрупните трансформации и друго. Всушност, квазигрупните трансформации се пресликувања од конечни низи над конечна азбука и покажуваат својства на дискретни динамички системи. Ваквата примена на квазигрупите и квазигрупните трансформации е релативно нова област и претставува предизвик за нивното натамашно истражување и нивна примена во разни области.

Првиот дел од оваа дисертација е посветен на случајните кодови базирани на квазигрупи предложени за прв пат од Данило Глигороски, Смиле Марковски и Љупчо Коцарев. Овие кодови се комбинација на криптографски алгоритми и кодови за корекција на грешки и зависат од неколку параметри. Ако овие кодови се користени за кодирање на пораки, тогаш до оригиналните податоци може да се дојде само доколку се знае точно кои параметри се користени во процесот на кодирање, дури и ако комуникацискиот канал е без пречки. Затоа, ако информациите кои се пренесуваат, се од таен карактер, предложените случајни кодови се во голема предност пред останатите.

Влијанието на параметрите врз перформансите на овие кодови е проучено во [88]. Од резултатите добиени од тоа проучување, заклучено е дека избраната квазигрупа, должината на почетниот клуч и начинот на избор на редундантната информација (изборот на патернот) имаат големо влијание на перформансите на овие кодови. Од направените експерименти со RCBQ може да се заклучи дека брзината на декодирање е еден од најголемите проблеми кај овие кодови.

Со цел да се забрза процесот на декодирање, Александра Поповска-Митровиќ, Смиле Марковски и Верица Бакева, дефинираат нов алгоритам за кодирање/декодирање наречен алгоритам-за-декодирање-со-пресеци (Cut-Decoding или АДП алгоритам), а потоа истите автори прават модификација на

овој алгоритам, наречена алгоритам-за-декодирање-со-4-пресеци (АД4П алгоритам).

Декодирањето со овие алгоритми е декодирање со листа, па оттука брзината на декодирање зависи од големината на листата (помала листа значи побрзо декодирање). Во овие алгоритми, големината на листата зависи од $B_{max}$ (претпоставен број на бит грешки при пренос на еден блок). За помали вредности на $B_{max}$ се добива помала листа, но проблемот е тоа што никогаш не може однапред да се знае точниот број на настанати грешки при пренос на еден блок. Ако тој број е поголем од предвидениот $B_{max}$, грешката нема да биде поправена, но ако предвиденото $B_{max}$ е преголемо, тогаш имаме преголеми листи и премногу споро декодирање. За да се избегне ова, барем за канали со помал шум, се предлагаат нови алгоритми за да се обезбеди декодирање со помали листи наречени Брз-алгоритам-за-декодирање-со-пресеци (Fast-Cut-Decoding или БрзАДП) и Брз-алгоритам-за-декодирање-со-4-пресеци (Fast-4-Sets-Cut-Decoding или БрзАд4П).

Во експериментите со каналите со рафални грешки, не добиваме добри резултати со претходно дефинираните алгоритми. Поради тоа што интерливерот и деинтерливерот се користат за справување со рафални грешки во комуникациските системи предлагаме нови алгоритми за кодирање/декодирање наречени РафаленАДП (Burst-Cut-Decoding) и РафаленАД4П (Burst-4-Sets-Cut-Decoding), во кои воведуваме интерливер во алгоритамот за кодирање и соодветен деинтерливер во алгоритамот за декодирање. Анализирајќи ги добиените резултати заклучуваме дека веројатностите за бит грешка со новите рафални алгоритми се неколку (од 2 до 10) пати подобри од соодветните веројатности за бит грешка добиени со соодветните стари алгоритми за кодирање/декодирање. Истиот заклучок може да се извледе и за веројатностите за пакет грешка.

Во сите експерименти за пренос на слики и аудио датотеки, кај сликите и аудио датотеките беа воочливи оштетувања кои се резултат на некорегираните грешки. Со цел да се поправат овие оштетувања и да се добијат појасни слики, дефиниран е филтер за подобрување на квалитетот на сликите и аудио датотеките.

Една можна практична примена на овие криптокодовите е во безжичните канали со голем шум, каде кодот природно се вклопува во податочното ниво (data link layer), опфаќајќи ги и кодирањето и шифрирањето во една шема. Предложените кодови би можеле да најдат примена и за кодирање и шифрирање кај сателитско дигитално видео емитување (satellite digital video broadcasting (DVB-S)). Моментално, DVB-S и DVB-S2 користат две нивоа на шема

за кодирање, проследено со шема на шифрирање. Нашиот пристап за кодирање ги заменува овие слоеви на кодирање и шифрирање во единствена шема во која се имплементирани сите потребни карактеристики.

Развојот на технологијата на блок-веригите (Blockchain) во денешното современо живеење, ни ја дава идејата за уште една примена на квазигрупите - примена во блок-веригите за забрзување на преносот на трансакциите во веригата и за намалување на просторот за складирање. Всушност, блок-веригите претставува децентрилизиран систем (без централен авторитет на одлучување) за размена на вредности директно помеѓу физичките лица, на пример, трансфер на пари без централен авторитет (банка, финансиска институција, ...), т.е. децентрализирана мрежа за пренос на вредности (криптовалути) заснована на $P2P$ мрежата. На почетокот од вториот дел на оваа дисертација, направено е истражување за технологијата на блок-веригите. Разгледани се основните поими користени во оваа технологија, начинот на работа на блок-веригата, нејзините основни карактеристики, како и нејзината примена во масивните податоци (BigData) и Интернет на нештата (IoT).

Еден од главните проблеми во блок-веригите е брзината на преносот. Мрежното кодирање како една техника што ја подобрува моќноста на мрежата и обезбедува поголема безбедност е едно од решенијата на овој проблем. Со цел да се обезбеди побрз и поедноставен процес на декодирање во мрежното кодирање, дефиниран е нов алгоритам за мрежно кодирање базиран на квазигрупи за зголемување на спроводливоста во комуникациската мрежа "пеперутка".

Како што се зголемува бројот на трансакции во блок-веригата, така се зголемува и просторот потребен за складирање на истите, со што се ограничува скалабилноста на овој систем. Намалувањето на големината на трансакцијата која се чува во блоковите на блок-веригата е едно од решенијата за намалување на просторот потребен за складирање и намалување на трошоците за складирање на трансакцијата. Од таму и идејата за алгоритам за намалување на просторот за складирање во блок-веригата со користење на шемата на Шамир за споделување на тајна.

На крајот може да се заклучи дека анализите од истражувањата направени во оваа дисертација даваат нови резултати во насока на примената на квазигрупите како во областа на кодирањето и криптографијата, така и во технологијата на мрежното кодирање и намалувањето на просторот за складирање во блок-веригата. Темата е актуелна и добиените резултати отворија многу прашања за понатамошни истражувања. На пример, како квазигрупите и квазигрупните трансформации можат да се применат конкретно во технологи-

јата на блок-веригите за намалување на просторот за складирање, без да се зголеми комплексноста во мрежата и трошоците настанати при складирање на трансакциите.

Предложените алгоритми за зголемување на спроводливоста во $P2P$ мрежите, како и во намалување на трошоците за складирање ќе може да се применат во тековните системи со блок-вериги.

# Литература


[1] Bakeva V., Popovska-Mitrovikj A., Mechkaroska D., Dimitrova V., Jakimovski B., Ilievski V.: *Gaussian channel transmission of images and audio files using cryptcoding*, IET Communications, Vol. 13, Issue 11, (2019), p. 1625 – 1632. (**IF. 1.779**)

[2] Belousov V. D.: *n-arnie Kvazigruppi* (*n-aryQuasigroups*), Stiinca, Kisiniev, 1972.

[3] Boucher P. ,Figueiredo Do Nascimento S., Kritikos M.: *How Blockchain technology could change our lives*, European Parliament. PE 581.948, 2017.

[4] Casey M., Crane J., Gensler G., Johnson S., Narula N.: *The Impact of Blockchain Technology on Finance*: A Catalyst for Change, Geneva, International Center for Monetary and Banking Studies 2, 2018.

[5] Chou P. A., Wu Y., Jain K.: *Practical network coding*, Allerton Conference on Communication, Control, and Computing, October 2003.

[6] Cover T.M., Thomas A.J. : *Elements of Information Theory.*

[7] Crosby M., Nachiappan, Pattanayak P., Verma S. ,Kalyanaraman V., *Blockchain technology: Beyond bitcoin*, Applied Innovation Review, Berkeley, Issue No.2, June 2016.

[8] Dai M., Zhang S., Wang H., and Jin S., *A low storage room requirement framework for distributed ledger in Blockchain*, IEEE Access, vol. 6, 2018, pp. 22970–22975.

[9] Dawson E., Donovan D.:*The breadth of Shamir's secret-sharing scheme*, Computers & Security, Volume 13, Issue 1, 1994, ISSN 0167-4048, Pages 69-78.

[10] Delmolino K., Arnett M., Kosba A.E., Miller A., Shi E.: *Step by step towards creating a safe Smart contract: lessons and insights from a cryptocurrency lab.*, IACR Cryptol ePrint Arch 460, 2015.

[11] Denes J., Keedwell A. D: *Latin Squares and their Applications*, The English Universities Press Ltd., 1974.

[12] Dimitrova V., Bakeva V., Popovska-Mitrovikj A., Krapež A.: *Cryptographic Properties of Parastrophic Quasigroup Transformation*, Markovski S., Gusev M. (eds.) ICT-Innovations 2012, Springer (2012), pp. 235-243.

[13] Dimitrova, V., Markovski, J.: *On Quasigroup Pseudo Random Sequence Generators*, Proc. 1st Balkan Conference in Informatics, Thessaloniki, Greece, 2003, pp. 393–401

[14] Dimitrova V., Markovski S.: *Classification of quasigroups by image patterns*, Proc. of the Fifth International Conference for Informatics and Information Technology, Macedonia, (2007) pp. 152-160.

[15] Gligoroski D., Dimitrova V., Markovski S.: *Quasigroups as Boolean functions, their equation systems and Groebner bases*, Book: "Groebner Bases, Coding, and Cryptography", ISBN 978-3-540-93805-7, Springer 2009, pp. 415-420.

[16] Gligoroski D., Markovski S., Kocarev Lj.: *Error-correcting codes based on quasigroups*, Proc. 16th Intern. Confer. Computer Communications and Networks (2007), pp. 165-172.

[17] Gligoroski D., Markovski S., Kocarev Lj.: *Totally asynchronous stream ciphers + Redundancy = Cryptcoding*, S. Aissi, H.R. Arabnia (Eds.): Proc. Internat. Confer. Security and management, SAM 2007, Las Vegas, CSREA Press (2007) pp. 446-451.

[18] Godoy W., Periera D.: *A proposal of a cryptography algorithm with techniques of error correction*, Computer Communications 20 (1997), pp. 1374-1380.

[19] Gupta V. , *A brief history of Blockchain*, Harvard Business Review, Fevruary 28, 2017.

[20] Fragouli Ch.,Soljanin E.: *Network Coding Fundamentals*, Foundations and Trends in NetworkingVol. 2, No. 1 (2007) 1–133, 2007

[21] https://api.blockchain.info/charts/preview/s-t/blocks-size.png?lang=en&start=1550148820

[22] https://atozforex.com/news/top-countries-where-bitcoin-is-legal/

[23] https://golem.network/index.html

[24] https://blockgeeks.com/guides/blockchain-scalability/

[25] https://blockgeeks.com/guides/ethereum/

[26] https://blockgeeks.com/guides/smart-contracts/

[27] https://blog.dataone.io/big-data-and-blockchain-c84c6c02544c

[28] https://cointelegraph.com/news/russian-minister-we-will-never-consider-bitcoin-legal

[29] https://cointelegraph.com/tags/segwit

[30] https://cryptocurrencyhub.io/an-introduction-to-consensus-algorithms-proof-of-stake-and-proof-of-work-cd0e1e6baf52

[31] https://cryptodaily.co.uk/2018/05/zilliqa-zil-answer-blockchain-scalability/

[32] https://commodity.com/cryptocurrency/what-are-forks/

[33] https://jaxenter.com/blockcahin-big-data-146218.html

[34] http://improving-bpm-systems.blogspot.com/2016/01/entarch-view-on-blockchain.html

[35] https://gendal.me/2015/02/10/a-simple-model-for-smart-contracts/

[36] https://github.com/capiman/sha256-sat-bitcoin

[37] http://learnmeabitcoin.com/faq/segregated-witness

[38] https://medium.com/ethereum-dapp-builder/how-blockchain-will-improve-big-data-da1547dd268

[39] https://people.xiph.org/ greg/attack-success.html

[40] https://techcrunch.com/2016/10/08/how-Blockchain-can-change-the-music-industry/

[41] https://techcrunch.com/2016/11/24/Blockchain-has-the-potential-to-revolutionize-the-supply-chain/

[42] http://www.altcointoday.com/bitcoin-ethereum-vs-visa-paypal-transactions-per-second/

[43] https://www.bbvaopenmind.com/en/a-secure-model-ofiotwithblockchain/?utm_source=views &utm_medium=article06& utm_campaign=MITcompany&utm_content=banafajan07

[44] https://www.buybitcoinworldwide.com/confirmations/

[45] https://www.coinbureau.com/review/zilliqa-zil/

[46] https://www.datanami.com/2018/06/28/blockchain-solutions-usher-in-era-of-trusted-big-data/

[47] https://www.draglet.com/Blockchain-applications/smart-contracts/use-cases

[48] https://www.ey.com/Publication/vwLUAssets/EYblockhain-in-insurance/$FILE/EY-blockhain-in-insurance.pd

[49] https://www.forbes.com/sites/bernardmarr/2018/02/07/5- blockchain-opportunities-no-company-can-afford-to-miss/ 9893de51a83d

[50] http://www.iamwire.com/wp-content/uploads/2016/12/how-blockchain-works.png

[51] https://www.ibm.com/blockchain/industries/supply-chain

[52] https://www.inc.com/nicolas-cole/web-analytics-willget-a-massive-improvement-with-blockchain-technology.html

[53] http://www.loc.gov/law/help/bitcoin-survey/

[54] https://www.kdnuggets.com/2017/09/introduction-blockchain-big-data.html

[55] https://www.reuters.com/article/us-jpmorgan-cyber-bitcoin-idUSKCN11P2DE

[56] Kim Y., Raman R. K., Kim Y.S., Varshney L. R., Shanbhag N. R., *Efficient local secret sharing for distributed Blockchain systems*, IEEE Communications Letters, vol. 23, no. 2, pp. 282–285, 2018.

[57] Knag, J., Stark, W., Hero, A.: *Turbo codes for fading and burst channels*, IEEE Theory Mini Conference, pp. 40–45 (1998)

[58] Koetter R.,MédardM.: *An Algebraic Approach to Network Coding*, IEEE/ACM Transactions on Networking, Vol. 11, No. 5, pp. 782–795, October 2003.

[59] Kozlenkova I.,Hult G.T.M., Lund D.J., Mena J.A., Kekec P.: *The role of marketing channels in supply chain managemen*, Journal of Retailing, 91 (4), 2015, pp 586–609.

[60] Krapež A.: *An Application of Quasigroups in Cryptology*, Math. Maced. Vol. 8 (2010), pp. 47-52.

[61] Labiod, H.: *Performance of Reed Solomon error-correcting codes on fading channels*, In: IEEE International Conference on Personal Wireless Communications (Cat. No.99TH8366), Jaipur, India, pp. 259-263 (1999)

[62] Laywine C. F., Mullen G. L.: *Discrete Mathematics using Latin Squares*, John Wiley & Sons, Inc., 1998.

[63] Li S.Y. R., Yeung R. W., Cai N., *Linear Network Coding*, IEEE Transactions on Information Theory, Vol. 49, No. 2, pp. 371–381, February 2003.

[64] Markovski S.: *Quasigroup string processing and applications in cryptography*, Proceedings of the 1stMII 2003 Conference, Thessaloniki, April 14-16,2003,pp. 278-290.

[65] Gligoroski D., Knapskog S. J., Andova S.:*Cryptcoding - Encryption and Error-Correction Coding in a Single Step*, The 2006 International Conference on Security and Management. Las Vegas, Nevada, USA: CSREA Press 2006.

[66] Markovski S., Gligoroski D., Andova S.: *Using quasigroups for one-one secure encoding*, Proc.VIII Conf.Logic and Computer Science"LIRA '97", Novi Sad, 1997, pp. 157-162.

[67] Markovski S., Gligoroski D., Bakeva V.: *Quasigrouop string processing: Part* 1, Contributions, Sec. Math. Tech. Sci., MANU, Vol. XX 1-2 (1999) pp. 13-28.

[68] Markovski S., Gligoroski D., Bakeva V.: *Quasigroup and Hash Function*, Discrete Mathematics and Applications: Proceedings of Sixth International Conference, South-West University, Blagoevgrad, Bulgaria (2001), pp. 43-50.

[69] Markovski S., Gligoroski D., Kocarev L.: *Unbiased Random Sequences from Quasigroup String Transformations*, Proceededings of Fast Software Encryption 2005, LNCS 3557, Springer-Verlag, 2005, pp. 163-180.

[70] Markovski S., Gligoroski D., Markovski J.: *Classification of quasigroups by random walk on torus*, Journal of applied mathematics and computing, Vol. 19, No. 1-2, September 2005, pp. 57-75.

[71] Markovski S., Kusakatov V.: *Quasigroups string processing: Part 2*, Contributions, Sec. Math. Tech. Sci., MANU, XXI, 1-2, 2000, pp. 15-32.

[72] Markovski S., Kusakatov V.: *Quasigroup string processing: Part 3*, Contributions, Sec. Math. Tech. Sci., MANU, XXIII-XXIV, 1-2, 2002-2003, pp.7-27. 7

[73] Mathur C.N., Narayan K., Subbalakshmi K.P.: *High Diffusion Cipher: Encryption and Error Correction in a Single Cryptographic Primitive*, Applied Cryptography and Network Security Lecture Notes in Computer Science, Vol. 3989 (2006), pp. 309-324. 9

[74] Mechkaroska D., Dimitrova V., Popovska-Mitrovikj A.: *A survey on applications of Blockchain technology*, Proceedings of 15th International Conference on Informatics and Information Technologies, April 2018 Mavrovo, Macedoinia, pp. 66–69.

[75] Mechkaroska D., Dimitrova V., Popovska-Mitrovikj A.: *Analysis of the possibilities for improvement of Blockchain technology*, 2018 26th Telecommunications Forum (TELFOR), Bel-

grade, Serbia, Nov.2018, doi: 10.1109/TELFOR.2018.8612034 https://ieeexplore.ieee.org/document/8612034

[76] Mechkaroska D., Popovska-Mitrovikj A. and Bakeva V.: *Cryptcodes Based on Quasigroups in Gaussian channel*, Quasigroups and Related Systems 24 (2016) No.2, pp. 249-268.

[77] Mechkaroska D., Popovska-Mitrovikj A., Bakeva V.: *A filter for Images Decoded using Cryptocodes Based on Quasigroups*, Proceedings of the 14th International Conference on Informatics and Information, Mavrovo, april 2017

[78] Mechkaroska D., Popovska-Mitrovikj A., Bakeva Smiljkova V.: *Performances of Fast Algorithms for Random Codes Based on Quasigroups for Transmission of Audio Files in Gaussian Channel*, In: Kalajdziski, S., Ackovska, N. (eds): ICT Innovations 2018. Engineering and Life Sciences, Springer International Publishing, pp. 286–296.

[79] Mechkaroska D., Popovska-Mitrovikj A., Bakeva V.: *New Cryptcodes for Burst Channels*, In: Ćirić M., Droste M., Pin JÉ. (eds) Algebraic Informatics. CAI 2019. Lecture Notes in Computer Science, vol 11545. Springer, Cham, pp. 202–212.

[80] Mechkaroska D., Popovska-Mitrovikj A., Bakeva V.: *Cryptcoding of Images for Transmission Trough a Burst Channels*, Journal of Engineering Science and Technology Review (JESTR), pp 65—69, ISSN: 1791-2377, Proceedings of Fourth International Scientific Conference "Telecommunications, Informatics, Energy and Management", September 2019, Kavala, Greece

[81] Mechkaroska D., Popovska-Mitrovikj A., Bakeva V., Dimitrova V.: *Network Coding based on quasigroup*, Proceedings of the 10th International Conference on Information Society and Technology, March 2020, Kopaonik, Serbia, pp. 240-243

[82] Mechkaroska D., Popovska-Mitrovikj A., Dimitrova V.: *Secure Big Data and IoT with implementation of Blockchain*, International Scientific Journal Security & Future, vol. 2(4), 2018, pp. 183 – 185 ISSN (PRINT) 2535-0668, ISSN (Online) 2535-082X International Scientific Conference on Security „CONFSEC 2018", December 2018, Bulgaria

[83] Nakamoto S., *Bitcoin: A peer-to-peer electronic cash system*, 2008, http://bitcoin.org/bitcoin.pdf.

[84] Nomura Research Institute, *Survey on Blockchain technologies and related services*, March 2016.

[85] Popovska-Mitrovikj A., Bakeva V., Markovski S.: *On random error correcting codes based on quasigroups*, Quasigroups and Related Systems Vol. 19, (2011), pp. 301-316.

[86] Popovska-Mitrovikj A., Bakeva V., Mechkaroska D.: *New Decoding Algorithm for Cryptcodes Based on Quasigroups for Transmission Through a Low Noise Channel*, In: Trajanov, D., Bakeva, V. (eds.): Communications in Computer and Information Science Series (CCIS), Vol.778, ICT-Innovations 2017, Springer, pp. 196–204.

[87] Popovska-Mitrovikj A., Bakeva V., Mechkaroska D.: *Fast Decoding with Cryptcodes for Burst Errors*, In: Dimitrova V., Dimitrovski I. (eds.): Communications in Computer and Information Science Series (CCIS), ISTInnovations (accepted).

[88] Popovska-Mitrovikj A., Markovski S., Bakeva V.: *Performances of error-correcting codes based on quasigroups*, D.Davcev, J.M.Gomez (Eds.): ICT-Innovations 2009, Springer (2009), pp. 377-389. 15

[89] Popovska-Mitrovikj A., Markovski S., Bakeva V.: *Increasing the decoding speed of random codes based on quasigroups*, S. Markovski, M. Gusev (Eds.): ICT Innovations 2012, Web proceedings, ISSN 1857-7288, pp. 93-102.

[90] Popovska-Mitrovikj A., Markovski S., Bakeva V.: *On improving the decoding of random codes based on quasigroups*, Proceedings of the 9th Conference on Informatics and Information Technology with International Particpants, Faculty of Computer Science and Engineering, University "Ss.Cyril and Methodius" (Macedonia), Bitola, Macedonia, April 2012, pp. 214-217.

[91] Popovska-Mitrovikj A., Markovski S., Bakeva V.: *Some new results for random codes based on quasigroups*, Proceedings of the 10th Conference on Informatics and Information Technology with International Participants, Faculty of Computer Science and Engineering, University "Ss.Cyril and Methodius" (Macedonia), Bitola, Macedonia, April 2013

[92] Popovska-Mitrovikj A., Markovski S., Bakeva V.: *4-Sets-Cut-Decoding algorithms for random codes based on quasigroups*, International Journal of Electronics and Communications (AEU), Elsevier, Vol.69, Issue 10, 2015, pp. 1417–1428.

[93] Popovska-Mitrovikj A., Mechkaroska D., Dimitrova V., Bakeva V.: *Algorithm for Reducing Storage in Blockchain based on Secret Sharing*, 2nd International Conference on Electrical, Communication and Computer Engineering (ICECCE 2020), Istanbul, Turkey, 12-13 June 2020, pp. 567–569

[94] Raman R.K., Varshney L.R.: *Distributed storage meets secret sharing on the blockchain*, in Proceedings of Information Theory and Applications Workshop (ITA), San Diego, CA, USA, February 2018.

[95] Raman R.K., Varshney L.R. :*Dynamic distributed storage for blockchains*, in Proc. IEEE Int. Symp. Inf. Theory (ISIT), Jun. 2018, pp. 2619–2623. [Online]. Available: https://arxiv.org/pdf/1711.07617.pdf

[96] Rao T. R. N, Nam K.H.: *Private-Key Algebraic-Code Encryptions*, IEEE Transaction on information theory, Vol. 35, No 4, (1989) pp. 829- 833.

[97] Rosenfeld M.: *Analysis of hashrate-based double-spending*, arXiv:1402.2009.

[98] Sankar K.: *Bit Error Rate (BER) for BPSK modulation*, August 2007 http://www.dsplog.com/2007/08/05/bit-error-probability-for-bpsk-modulation/

[99] Satyavolu P., Sangamnerkar A.: *Blockchain's smart contracts: Driving the next wave of innovation across manufacturing value chains*. https://www.cognizant.com/whitepapers/BlockChains-smart-contracts-driving-the-next-wave-of-innovation-across-manufacturing-value-chains-codex2113.pdf

[100] Shamir: *How to share a secret*, Communications of the ACM, vol. 22, no. 11, pp. 612-613, 1979.

[101] The Zilliqa team: *The Zilliqa Techical Whitepaper*, https://docs.zilliqa.com/whitepaper.pdf, August 10, 2017.

[102] Tzonelih H., Rao T.R.N.: *Secret error-correcting codes*, Goldwasser, R. (Ed.): Advances in Cryptology - CRYPTO ‘88, LNCS 403, (1990), pp.540-563.

[103] Zivic N., Ruland C.: *Parallel Joint Channel Coding and Cryptography*, Inter. Jour. of Electrical and Electronics Rngineering 4:2 (2010), pp. 140-144.

[104] Yang M., Yang Y., Fellow: *Applying Network Coding to Peer-to-Peer File Sharing*, IEEE transactions on computers, Vol. 63, No. 8, August 2014

[105] Zhang P., Lin C., Senior Member, IEEE, Jiang Y., Fan Y., Xuemin (Sherman) Shen: *A Lightweight Encryption Scheme for Network-Coded Mobile Ad Hoc Networks*, IEEE Transactions on Parallel and Distributed Systems 25 (9), pp. 2211-2221.

[106] Мечкароска Д.: *Перформанси на Случајните кодови базирани на квазигрупи при пренос низ Гаусов канал*, магистерски труд, Универзитет ”Св. Кирил и Методиј”, Скопје, 2014.

[107] Поповска-Митровиќ А.: *Прилог кон примена на квазигрупите во теоријата на кодирање и криптографијата*, докторска дисертација, Универзитет ”Св. Кирил и Методиј”, Скопје, 2014

# Прилози

# Contents

# Chapter 1

# Quasigroups and quasigroups string transformation

In this chapter, we will give a brief overview of quasigroups. More details for quasigroups and their properties are given in [2], [11], [62]. Using quasigroups, some string transformations have been defined in [64], [66], [67], [68], [60], [12]. They are called quasigroup string transformations. These quasigroup transformationsfor are very useful for construction of cryptographic primitives and for designing error-detecting and correcting codes. The reason for this are: the structure of quasigroups, their large number and their properties.

## 1.1 Quasigroups and their properties useful in cryptography and coding theory

A quasigroup $(Q, *)$ is a groupoid, i.e., a set $Q$ with a binary operation $* : Q^2 \to Q$, satisfying the law:

$$(\forall u, v \in Q)(\exists! x, y \in Q) \ \ (x * u = v \ \& \ u * y = v) \tag{1.1}$$

In fact, (1.1) says that a groupoid $(Q, *)$ is a quasigroup if and only if the equations $x * u = v$ and $u * y = v$ have unique solutions $x$ and $y$ for each given $u, v \in Q$.

Further on, we will assume that the set $Q$ is final. Let the quasigroup $(Q, *)$ is given. Five new quasigroup operations, called parastrophies, can be performed from the operation $*$. We need only one of them denoted with "$\backslash$" which is defined as follows:

$$x * y = z \quad \Longleftrightarrow \quad y = x \setminus z.$$

The algebra $(Q, *, \backslash)$ satisfies identities:

$$x \backslash (x * y) = y, \quad x * (x \backslash y) = y \tag{1.2}$$

and $(Q, \backslash)$ is also a quasigroup.

Further on, we give some properties of quasigroups.

**Definition 1.1.** *A quasigroup $(Q, *)$ is commutative if it satisfies the identity*

$$x * y = y * x. \tag{1.3}$$

**Definition 1.2.** *A quasigroup $(Q, *)$ is associative if it satisfies the identity*

$$(x * y) * z = x * (y * z) \tag{1.4}$$

**Definition 1.3.** *A quasigroup $(Q, *)$ is a left loop if it has a left unit element $e \in Q$ such that*

$$(\forall\, x \in Q)\ (e * x = x) \tag{1.5}$$

**Definition 1.4.** *A quasigroup $(Q, *)$ is a right loop if it has a right unit $e \in Q$ such that*

$$(\forall\, x \in Q)\ (x * e = x) \tag{1.6}$$

**Definition 1.5.** *A quasigroup $(Q, *)$ is a loop if it has an unit $e \in Q$ such that*

$$(\forall\, x \in Q)\ (x * e = x = e * x) \tag{1.7}$$

Let note that $(Q, *)$ is a loop iff it is a left loop and right loop.

**Definition 1.6.** *A quasigroup $(Q, *)$ is idempotent if it satisfies the identity*

$$x * x = x \tag{1.8}$$

A quasigroup $(Q, *)$ of order $n$ is *shapeless* if and only if it is non-idempotent, non-commutative, non-associative, it does not have neither left nor right unit, it does not contain proper sub-quasigroups, and there is no $k < 2n$ for which identities of this kinds are satisfied:

$$\underbrace{x(\ldots * (x}_{k} * y)) = y,\ y = ((y * \underbrace{x) * \ldots) * x.}_{k} \tag{1.9}$$

The condition $k < 2n$ for identities (1.9) means that any left and right translation of quasigroup $(Q, *)$ should have the order $k \geqslant 2n + 1$. For cryptographic purposes, it is preferably to choose a shapeless quasigroup [17].

Table 1.1: Number of quasigroups of order $n \leqslant 11$

| $n$ | $Q_n$ |
|---|---|
| 1 | 1 |
| 2 | 2 |
| 3 | 12 |
| 4 | 576 |
| 5 | 161280 |
| 6 | 812851200 |
| 7 | 61479419904000 |
| 8 | 108776032459082956800 |
| 9 | 5524751496156892842531225600 |
| 10 | 9982437658213039871725064756920320000 |
| 11 | 776966836171770144107444346734230682311065600000 |

## 1.2 Quasigroups string transformation

Here, we will describe the quasigroup transformations that we will use in the following chapters for designing a class of error-correcting codes. For the given alphabet $Q$, the set of all finite strings of the elements of $Q$ will be denoted by $Q^+$, i.e., $Q^+ = \{a_1 a_2 \dots a_n | a_i \in Q, n \in \mathbb{N}\}$ or $Q^+ = Q \cup Q^2 \cup Q^3 \dots$. Furthermore, we will consider the quasigroup which is a degree of 2, i.e. $2^k$, for some $k$. In this way, any element of $Q$ can be presented as $k$-tuples of bits. If $k = 4$, the elements of $Q$ are nibbles.

Using the quasigroup operation $*$, two *quasigroup string transformations* are defined in [67]. They are mappings from $Q^+$ to $Q^+(\mid Q \mid \geqslant 2)$ called $e$-transformation and $d$ transformation.

- *$e$-transformation* is a function in $Q^+$, which is defined as follows.
  For fixed element $l \in Q$, called a *leader* and $a_i \in Q$, $i = 1, 2, \dots, n$,

$$e_{l,*}(a_1 \dots a_n) = b_1 \dots b_n \Leftrightarrow \begin{cases} b_1 &= l * a_1, \\ b_{i+1} &= b_i * a_{i+1}, \quad i = 1, 2, \dots n-1 \end{cases} \tag{1.10}$$

  The graphical presentation of $e_{l,*}$ is given on Fig. 1.1.

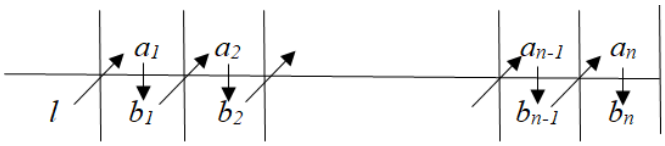



Figure 1.1: Graphical representation of the function $e_{l,*}$

- *d-transformation* is a function $d_{l,\backslash} : Q^+ \to Q^+$ which is defined as follows. For a fixed leader $l$ and $a_i \in Q$, $i = 1, 2, \ldots, n$,

$$d_{l,\backslash}(a_1 \ldots a_n) = c_1 \ldots c_n \Leftrightarrow \begin{cases} c_1 = l \setminus a_1, \\ c_{i+1} = a_i \setminus a_{i+1}, \quad i = 1, 2, \ldots n-1 \end{cases} \tag{1.11}$$

The graphical presentation of $d_{l,\backslash}$ is given on Fig. 1.2.

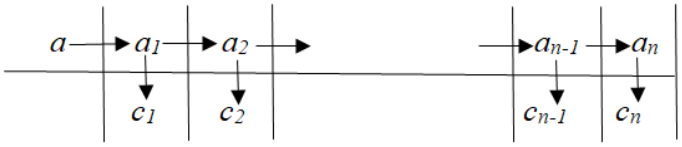



Figure 1.2: Graphical representation of the function $d_{l,\backslash}$

As a result of the equation (1.2) we have that the following property holds:

**Property 1.1.** ([67]) *The mappings $e$ and $d$ are bijections (permutations) and for each string $\alpha \in Q^+$, it is true*

$$d_{l,\backslash}(e_{l,*}(\alpha)) = \alpha = e_{l,*}(d_{l,\backslash}(\alpha))$$

*i.e., $d = e^{-1}$ is an inverse bijection of $e$ .*

Let $*_1, *_2, ..., *_n$ are quasigroup transformations defined over $Q$ and $\backslash_1, \backslash_2, ..., \backslash_n$ are corresponding parastophic transformations over $Q$. Let the transformations $e_{l_1,*_1}, e_{l_2,*_2}, ..., e_{l_n,*_n}$ are defined in the same way as in (1.10), a $d_{l_1,\backslash_1}, d_{l_2,\backslash_2}, ..., d_{l_n,\backslash_n}$ as in (1.11), by selecting fixed leaders $l_1, l_2,\ldots,l_n \in Q$.

We form the following compositions:

$$E = e_{l_n,*_n} \circ e_{l_{n-1},*_{n-1}} \circ, ..., \circ e_{l_1,*_1}$$
$$D = d_{l_1,\backslash_1} \circ d_{l_2,\backslash_2} \circ, ..., \circ d_{l_n,\backslash_n}$$

As a consequence of Property 1, we have that $E$ and $D$ are bijections that are mutually inverse.

## 1.3 Cryptographic properties of quasigroup transformations

Here we will consider some cryptographic properties of quasigroups string ($e$- and $d$-) transformations. The $E$-transformation can be used for designing an encrypting algorithm, and $D$-transformation - for decrypting algorithm. Let $M$ be an original message and $C = E(M)$ is corresponding encrypted message. Discovering the message $M$ from message $C$ is imposible without knowing the quasigroups used in transformation $E$. A brute force attack for questing these quasigroup is also impossible due to the large number of quasigroups (especially, for large order $n$).

– One of the most important cryptographic properties of quasigroup transformations is *uniform distribution* of the symbols (pairs, triples and so on) in the encrypted string which provides resistance against statistical kind of attack. This property of quasigroup $E-$transformation is addressed in [67] where the authors proved the following property.

  Let $(Q, *)$ be a given finite quasigroup and $\mathbf{p} = (p_1, p_2, ..., p_{|Q|})$ be the probability distribution of the symbols in $Q$, such that $p_i > 0$ for each $i$ and $\sum_i p_i = 1$. Then the following theorem holds.

  **Theorem 1.1.** ([67]) *Consider a random string $M = a_1 a_2 \dots a_n$ $(a_i \in Q)$ drawn i.i.d. according to the probability distribution* $\mathbf{p}$. *Let $C$ be obtained after $k$ applications of an $e-$transformation on $M$. If $n$ is a large enough integer then the distribution of substrings of $C$ of length $t$ is uniform for each $1 \leqslant t \leqslant k$.* (We note that for $t > k$ the distribution of substrings of $C$ of length $t$ is not uniform.)

– Another important cryptographic property of the encrypted string $C$ is its *period*. In [69] authors proved that the period of quasigroup processed strings grows at least linearly and the increasing of the period depends on the chosen

quasigroup. More about the period of quasigroup processed string is given in [13].

## 1.4 Classification of quasigroups by fractality and linearity

The results of existing investigations tell us that the good cryptographic properties of quasigroup processed string depend on the chosen quasigroups. Therefore, classifications of finite quasigroups are very important for successful application of quasigroups in cryptography and coding theory. It is a difficult problem, since the number of quasigroups (even of small order) is very large.

A classification of quasigroups by graphical presentation of quasigroup processed strings is given in [14]. In that paper, the authors classified the set of all quasigroups of given finite order $n$ into 2 disjoint classes, the class of so called *fractal* quasigroups (if the graphical presentation of quasigroup processed strings has structure), and the class of so called *non-fractal* quasigroups (if the graphical presentation of quasigroup processed strings has not structure). The class of fractal quasigroups is not recommended to be used for designing cryptographic primitives. The number of fractal quasigroups of order 4 is 192 and the number of non-fractal quasigroups is 384.

In [15], the authors give a representation of a quasigroup as vector valued Boolean functions. A quasigroup $(Q, *)$ of order $2^n$ can be represented by a vector valued Boolean function $f : \{0,1\}^{2n} \to \{0,1\}^n$ on the following way. Let represent an arbitrary $x \in Q$ as binary vector $x = (x_1, x_2, \ldots, x_n) \in \{0,1\}^n$. Then, for each $x, y \in Q$

$$x * y \equiv f(x_1, \ldots, x_{2n}) = (f_1(x_1, \ldots, x_{2n}), \ldots, f_n(x_1, \ldots, x_{2n})) \tag{1.12}$$

where we take

$$x = (x_1, x_2, \ldots, x_n),\ y = (x_{n+1}, x_{n+2}, \ldots, x_{2n})$$

and

$$f_i : \{0,1\}^{2n} \to \{0,1\}$$

are the corresponding components of $f$.

Using this Boolean representation, (in [15]), the quasigroups are divided into two classes: linear quasigroups and non-linear quasigroups by Boolean representation.

**Definition 1.7.** *The quasigroup $(Q, *)$ is linear if all functions $f_i$ (in 1.12) for $i = 1, 2, \ldots n$ are linear polynomials.*

**Definition 1.8.** *The quasigroup is non-linear if there exist function $f_i$ (in 1.12) for some $i = 1, 2, \ldots n$ which is not linear.*

Non-linear quasigroups are divided into two subclasses: partial (or weak) non-linear and pure non-linear quasigroups. Partial non-linear are quasigroups such that there exist at least one component that is linear Boolean function and at least one component that is non-linear Boolean function. A quasigroup is pure non-linear if all its components are non-linear Boolean functions.

# Chapter 2

# Random Codes based on quasigroup

The initial idea for applying quasigroups in random codes is given in [17] and [16], where Random Codes Based on Quasigroup (RCBQ) are proposed. RCBQs are cryptocodes, i.e., they are defined using a cryptographic algorithm during the encoding/decoding process. These codes with a single algorithm allow not only correction of certain amount of errors that occur when transmitting through a noisy channel, but also provide information security. The recipient of the information can only access the original data if he knows exactly what parameters are used in the coding process, even if the communication channel is noiseless. In this thesis we consider RCBQ as error correction codes and therefore their cryptographic properties will not be analyzed here.

These codes correct errors by using encryption/decryption algorithms during the encoding and decoding process. We will denoted these coding/decoding algorithms as Standard Algorithm for RCBQ. They include several parameters and their performances depend on the chosen parameters. The influence of the parameters on the performances of these codes has been studied in [88]. In [85] the authors compared performances of these codes with the performances of Reed-Miller (RMC) and Reed-Solomon codes (RSC). From the obtained results the authors concluded that, the RMC and the RSC have better decoding performances in a binary symmetrical channel with bit-error probability $p < 0.05$. In the opposite case, the RCBQ outperforms them significantly. Nevertheless, the time efficiency of the RMC and the RSC is much higher than that of RCBQ. So, the speed of decoding of RCBQ is its disadvantage and it is a challenge for further improvements. In order to improve the performances of these codes, the authors in [89], [90], [91] [107], defined new algorithms for coding/decoding, called Cut-decoding algorithm and 4-Sets-Cut-Decoding algorithm. In these papers, the performances

of Random Codes Based on Quasigroup for data transmission through a binary-symmetric channel are considered.

## 2.1 Description of Standard algorithm for RCBQ

In this section, we will present the Standard algorithm for Random Codes Based on Quasigroups proposed by Gligoroski, Markovski and Kocarev ([17] [16] [65]).

### 2.1.1 Totally asynchronous stream cipher

Random Codes Based on Quasigroup are based on a class of totally asynchronous stream cipher ($TASC$), which concept is defined in [17] as follows:

**Definition 2.1.** *A totally asynchronous stream cipher is one in which the keystream is generated as a function of the intermediate key and all previous plaintext letters. The encryption function of a totally asynchronous stream cipher can be described by the equations:*

$$k^{(i+1)} = f(k^{(i)}, m_i), c_i = h(k^{(i)}, m_i)$$

*where $k^{(0)}$ is the initial secret state of the key, $k^{(i)}$ are intermediate keys, $f$ is the key next-state function, and $h$ is the output function. The decryption function of a totally asynchronous stream cipher can be described by the equations:*

$$k^{(i+1)} = f'(k^{(i)}, c_i), m_i = h'(k^{(i)}, c_i).$$

From the definition, it is clear that the totally asynchronous stream cipher gives cipher text $c_i$ which depends on all previous plain text letters $m_0, m_1, ..., m_i$. The authors of RCBQ give one possible implementation of TASC by using quasigroup string transformations, called EdonZ.

### 2.1.2 Standard algorithm for coding process

Let $Q$ be a set of all $a$-bit symbols, i.e., $Q$ has $2^a$ letters, $(Q, *)$ is a given quasigroup, and $(Q, \backslash)$ is its parastrophe. Before encoding with Standard algorithm for RCBQ, the sequence of bits obtained from the information source is divided into blocks with a length of $N_{block}$ bits and let $M$ be one of these blocks. We will assume that $M = M_1 M_2 \dots M_l$, where $M_i \in Q$, i.e., $M_i$ are symbols of $a$-bits. Hence it is clear that $N_{block} = la$.

Then the coding process is performed in the following steps:

– The message $M$ received from the source is expanded by adding redundant information. In this way, each message with a length $N_{block}$ is mapped to a message with length $N > N_{block}$. This can be done in many different ways, but usually only zeros that are appropriately placed in the message are added, according to certain patterns and we get the expanded message:

$$L = L^{(1)}L^{(2)}...L^{(s)} = L_1L_2...L_m,$$

where $L_i \in Q$, and $L^{(i)}$ are sub-blocks of $r$ a-bit symbols. So, $N = ma$ and $m = rs$. In this way, we obtain $(N_{block}, N)$ code with a rate $R = N_{block}/N$.

– The extended message is encoded using quasigroup transformations. The encoding/encrypting algorithm (from TASC) is given in Figure 2.1.

– At the end of the coding algorithm, the code word

$$C = C_1C_2...C_m,$$

is obtained, where $C_i \in Q$.

### 2.1.3 Standard algorithm for decoding process

In fact, the decoding is a procedure in which the sent codewords should be returned (if it possible). Since errors occur during transmission, decoding is only possible if redundant information is used. In this code, decoding uses the fact that some letters in certain positions in the extended message are zero. During the decoding process block by block is decoded sequentially. This type of decoding allows part of the codeword to be decoded when it is impossible to decode the entire codeword. After transmission through a noisy channel, the codeword $C$ will be received as a message $D = D^{(1)}\ D^{(2)} \dots D^{(s)} = D_1D_2 \dots D_m$ where $D^{(i)}$ are blocks of $r$ symbols from $Q$ and $D_i \in Q$. The decoding process consists of four steps: $(i)$ procedure for generating the sets with predefined Hamming distance, $(ii)$ inverse coding algorithm, $(iii)$ procedure for generating decoding candidate sets and $(iv)$ decoding rule.

*Procedure for generating the sets with predefined Hamming distance-*
The probability that maximum $t$ bits in $D^{(i)}$ are not correctly transmitted is

$$P(p;t) = \sum_{k=0}^{t} \binom{ra}{k} p^k (1-p)^{ra-k},$$

| **Encryption** | **Decryption** |
|---|---|
| **Input**: Key $k = k_1k_2 \dots k_n$ and $L = L_1L_2 \dots L_m$<br>**Output**: codeword $C = C_1C_2...C_m$ | **Input**: The pair $(a_1a_2 \dots a_s, k_1k_2 \dots k_n)$<br>**Output**: The pair $(c_1c_2 \dots c_s, K_1K_2 \dots K_n)$ |
| For $j = 1$ to $m$<br>$X \leftarrow L_j$;<br>$T \leftarrow 0$;<br>For $i = 1$ to $n$<br>$X \leftarrow k_i * X$;<br>$T \leftarrow T \oplus X$;<br>$k_i \leftarrow X$;<br>$k_n \leftarrow T$<br>**Output**: $C_j \leftarrow X$ | For $i = 1$ to $n$<br>$K_i \leftarrow k_i$;<br>For $j = 0$ to $s - 1$<br>$X, T \leftarrow a_{j+1}$;<br>$temp \leftarrow K_n$;<br>For $i = n$ to 2<br>$X \leftarrow temp \setminus X$;<br>$T \leftarrow T \oplus X$;<br>$temp \leftarrow K_{i-1}$;<br>$K_{i-1} \leftarrow X$;<br>$X \leftarrow temp \setminus X$;<br>$K_n \leftarrow T$;<br>$c_{j+1} \leftarrow X$;<br>**Output**: $(c_1c_2 \dots c_s, K_1K_2 \dots K_n)$ |

Figure 2.1: Algorithms for encryption and decryption

where $p$ is the probability of bit-error in the channel. Let $B_{max}$ be a given integer which denotes the assumed maximum number of bit errors that occur in a block during transmission. We generate the sets $H_i = \{\alpha | \alpha \in Q^r, \quad H(D^{(i)}, \alpha) \leqslant B_{max}\}$, for $i = 1, 2, \dots, s$, where $H(D^{(i)}, \alpha)$ is a Hamming distance between $D^{(i)}$ and $\alpha$. The cardinality of the sets $H_i$ is

$$B_{checks} = 1 + \binom{ra}{1} + \binom{ra}{2} + ... + \binom{ra}{B_{max}}$$

and the number $B_{checks}$ determines the complexity of the decoding procedure: to find the element $C^{(i)}$ in the set $H_i$, less than or equal to $B_{checks}$ checks have to be made. Clearly, for efficient decoding the number of checks $B_{checks}$ has to be reduced as much as it is possible.

*Inverse coding algorithm* – The inverse coding algorithm (ICA) is the decrypting algorithm of TASC given in Figure 2.1.

*Generating decoding candidate sets* - The decoding candidate sets $S_0$, $S_1$, $S_2$,...,$S_s$ are defined iteratively. Let $S_0 = (k_1 \dots k_n; \lambda)$, where $\lambda$ is the empty sequence. Let $S_{i-1}$ be defined for $i \geqslant 1$. Then $S_i$ is the set of all pairs $(\delta, w_1 w_2 \dots w_{rai})$ obtained by using the sets $S_{i-1}$ and $H_i$ as follows ($w_j$ are bits). For each element $\alpha \in H_i$ and each $(\beta, w_1 w_2 \dots w_{ra(i-1)}) \in S_{i-1}$, we apply the inverse coding algorithm with input $(\alpha, \beta)$. If the output is the pair $(\gamma, \delta)$ and if both sequences $\gamma$ and $L^{(i)}$ have the redundant zeros in the same positions, then the pair $(\delta, w_1 w_2 \dots w_{ra(i-1)} c_1 c_2 \dots c_r) \equiv (\delta, w_1 w_2 \dots w_{rai})$ ($c_i \in Q$) is an element of $S_i$.

*Decoding rule* – The decoding of the received message $D$ is given by the following rule:

- If the set $S_s$ contains only one element $(d_1 \dots d_n, w_1 \dots w_{ras})$ then $L = w_1 \dots w_{ras}$ is the decoded (redundant) message. In this case, we say that we have a *successful decoding*.
- If the decoded message is not the correct one then we have an *undetected-error*.
- In the case when the set $S_s$ contains more than one element, we say that the decoding of $D$ is unsuccessful (of type *more-candidate-error*).
- In the case when $S_j = \emptyset$ for some $j \in \{1, \dots, s\}$, the process will be stopped (and then we say that an error of type *null-error* appears). We conclude that for some $i \leqslant j$, $D^{(i)}$ contains more than $B_{max}$ errors, resulting in $C_i \notin H_i$.

## 2.2 Cut-Decoding algorithm

A. Popovska-Mitrovikj, S. Markovski and V.Bakeva in their paper [88] investigated the influence of parametars of Random Codes Based on Quisigroup on codes performances. They conclude that all parameters used in the code (such as, the pattern, the key length, the quasigroup) affect on the performances of Random Codes Based on Quasigroup. But the biggest problem of these codes is the decoding speed. In order to improve the decoding speed, these authors (in the papers [89], [90], [91]) define a new encoding/decoding algorithm called Cut-Decoding algorithm and they study their performances when messages are transmitted through a binary symmetrical channel. Since decoding with RCBQ is actually a list decoding, decoding speed and the probability of accurate decoding depend on the size of the lists of potential candidates for the decoded message.Therefore, the

Cut-Decoding algorithm has been proposed to reduce the number of candidates for decoding in all iterations of the decoding process. In this algorithm, a message is coded twice using different parameters, and the candidates for the decoded message are obtained using the intersection of the corresponding sets $S$. On this way the decoding process for code (72,288) is 4.5 times faster than the Standard algorithm ([89], [107]).

### 2.2.1 Coding with Cut-Decoding algorithm

In Standard algorithm for coding described in the previous section, we used code $(N_{block}, N)$ with the rate $R = N_{block}/N$. In Cut-Decoding algorithm, two $(N_{block}, N/2)$ codes with rate $2R$ are used to encode/decode the same message with $N_{block}$ bits. The coding process consists of the following steps:

- The input message $M = M_1M_2...M_l$ is expanded by adding redundant $\nu$ zero symbols (in the same way as in the Standard coding algorithm) and thus we obtain redundant message $L = L^{(1)}L^{(2)}...L^{(s/2)} = L_1L_2...L_{m/2}$ of $N/2$ bits, where $L^{(i)}$ is a subblock of $r$ symbols from the alphabet $Q$, and $L_i \in Q$. Thereby, $N = am$, $m = rs$, and $m = l + \nu$.

- Then, for coding, we use twice the coding algorithm given in Figure 2.1, on the same redundant message $L$, using different parameters (different keys or quasigroups).

- In this way we obtain the codeword for the input message $M$ as a concatenation of two codewords of $N/2$ bits, i.e.,

$$C = C_1C_2...C_{m/2}C_{m/2+1}...C_m,$$

  where $C_i \in Q$.

### 2.2.2 Decoding with Cut-Decoding algorithm

The coded message is transmitted through a Gaussian channel and we obtain the outgoing message $D = D^{(1)}D^{(2)}...D^{(s)}$, where $D^{(i)}$ are sub-blocks of $r$ symbols. The message $D$ is divided into two messages $D_1 = D^{(1)}D^{(2)}...D^{(s/2)}$ and $D_2 = D^{(s/2+1)}\ D^{(s/2+2)}...D^{(s)}$ with equal length. Then, these two messages are decoded parallel with the corresponding parameters.The decoding process consists of the same 4 steps as Standard algorithm, with a modification in the procedure

| | |
|---|---|
| 1. The sets $H_i^{(1)}$ are formed. | 1. The sets $H_i^{(2)}$ are formed |
| 2. For each $\alpha_1 \in H_i^{(1)}$, we apply ICA $(\alpha_1, k_1)$ and let $(\beta_1, k_1')$=ICA$(\alpha_1, k_1)$. | 2. For each $\alpha_2 \in H_i^{(2)}$, we apply ICA $(\alpha_2, k_2)$ and let $(\beta_2, k_2')$=ICA$(\alpha_2, k_2)$. |
| 2.1. We check if $\beta_1$ and $L^{(i)}$ have the redundant symbols in the same positions. | 2.1. We check if $\beta_2$ and $L^{(i)}$ have the redundant symbols in the same positions. |
| If they do then $(k_1', \beta_1) \in S_i^{(1)}$ | If they do then $(k_2', \beta_2) \in S_i^{(2)}$ |

Table 2.1: Generating decoding candidate sets

for generating decoding candidate sets. For each block of the two messages, the steps in Table 2.1 are done in parallel.

– Let two decoding candidate sets $S_i^{(1)}$ and $S_i^{(2)}$ be obtained in the both decoding processes, in the same way as in Standard RCBQ.

– Let
$$V_j = \{w_1 w_2 \dots w_{ri} | (\delta, w_1 w_2 \dots w_{rai}) \in S_i^{(j)}\}, j = 1, 2$$
and $V = V_1 \cap V_2$.

– For each $(\delta, w_1 w_2 \dots w_{rai}) \in S_i^{(1)}$, if $w_1 w_2 \dots w_{rai} \notin V$, then
$$S_i^{(1)} \leftarrow S_i^{(1)} \setminus \{(\delta, w_1 w_2 \dots w_{rai})\}.$$
Also, for each $(\delta, w_1 w_2 \dots w_{rai}) \in S_i^{(2)}$, if $w_1 w_2 \dots w_{rai} \notin V$, then
$$S_i^{(2)} \leftarrow S_i^{(2)} \setminus \{(\delta, w_1 w_2 \dots w_{rai})\}.$$
Actually, we eliminate from $S_i^{(1)}$ all elements whose second part does not match with the second part of an element in the $S_i^{(2)}$, and vice versa. In the next iteration the both processes use the corresponding reduced sets $S_i^{(1)}$ and $S_i^{(2)}$.

– The procedure is repeated for each $i \leqslant s/2$.

The decoding rule in Cut-Decoding algorithm is defined in the following way.

- After the last iteration, if the reduced sets $S_{s/2}^{(1)}$ and $S_{s/2}^{(2)}$ have only one element with same second component $w_1 \dots w_{ras/2}$, then $L = w_1 \dots w_{ras/2}$ is the decoded redundant message. In this case, we say that we have a *successful decoding*.

- If the decoded message is not the correct one then we have an *undetected-error*.

- If we obtain $S_i^{(1)} = \emptyset,\ S_i^{(2)} \neq \emptyset$ or $S_i^{(2)} = \emptyset,\ S_i^{(1)} \neq \emptyset$ in some iteration then the decoding of the message continues only with the nonempty set $S_i^{(2)}$ or $S_i^{(1)}$, correspondingly, by using the standard RCBQ decoding algorithm.

- In the case when $S_i^{(1)} = S_i^{(2)} = \emptyset$ in some iteration, then the process will be stopped (*null-error* appears).

- If the reduced sets $S_{s/2}^{(1)}$ and $S_{s/2}^{(2)}$ have more than one element, after the last iteration, we have *more-candidate-error*.

- To resolve the problem of greater number of *more-candidate-errors* a heuristic is used in the decoding rule to eliminate this type of errors. Namely, from the experiments with RCBQ it can be seen that when the decoding ends with more elements in the reduced decoding candidates sets in the last iteration, almost always the correct message is in these sets. In this case, we can randomly select a message from the one of the reduced sets in the last iteration and it can be taken as the decoded message.

In the paper [91] the authors proposed methods for reducing the number of *null-error* and *more-candidate-error* in Cut-Decoding algorithm:

1. In order to decrease the number of unsuccessful decodings with *null-error*, method by backtracking in Cut-Decoding algorithm, is defined. Actually, if in the $i^{th}$ iteration of the decoding process, it is obtained that $S_i^{(1)} = S_i^{(2)} = \emptyset$, (i.e., the two reduced sets are empty), it means that in a previous step $j < i$ the correct block, which was to be processed, is lost. This is the case if the number of errors during the transmission of a block is greater than $B_{max}$. Some of these errors will be eliminated if several iterations of the decoding process are canceled and some of them are repeated with a larger value of $B_{max}$.

2. A similar modification by returning in Cut-Decoding algorithm is proposed in order to reduce the number of unsuccessful decodings with *more-candidate-error*. If the decoding process ends with *more-candidate-error*, then the number of candidates will be reduced to one, if some iterations are canceled and they are performed with a smaller value of $B_{max}$. Namely, in the case when the decoding ends with more than one element in the reduced decoding candidate sets in the last iteration, then several iterations are canceled and the first of the canceled iterations is performed with a smaller value of $B_{max}$. In the following iterations, the previous value of $B_{max}$ is used.

In order to reduce the number of unsuccessful decodings of *null-error* and *more-candidate-error* in Cut-Decoding algorithm, these two proposed methods have been combined and experiments with this algorithm have been performed in my master's thesis. The percentage of eliminated unsuccessful decodings ranges from 22% to 90% depending on the value of $SNR$ in Gaussian channel.

-

## 2.3 4-Sets-Cut-Decoding algorithm

Improving the decoding speed with Cut-Decoding algorithm gives the idea of using intersections of more sets $S_i$, in order to obtain a greater increasing in decoding speed. Thus, the authors of Cut-Decoding algorithm in [92] modified that algorithm and use four transformations of the redundant message. With this new algorithm, called the 4-Sets-Cut-Decoding algorithm, a better improvement in decoding speed was obtained. Also, in order to improve the probability of package error and bit error, several methods have been defined for generating reduced sets with decoding candidates.

### 2.3.1 Coding with 4-Sets-Cut-Decoding algorithm

In this modification of Cut-Decoding algorithm instead of $(N_{block}, N)$ code with rate $R$, we use four $(N_{block}, N/4)$ codes with rate $4R$, that encode/decode a same message of $N_{block}$ bits.

The input message $M = M_1M_2...M_l$ is expanded by adding redundant $\nu$ zero symbols (in the same way as in the Standard coding and Cut-Decoding algorithm) and thus we obtain redundant message $L = L^{(1)}L^{(2)}...L^{(s/4)} = L_1L_2...L_{m/4}$ of

$N/4$ bits, where $L^{(i)}$ is a subblock of $r$ symbols from the alphabet $Q$, and $L_i \in Q$. Thereby, $N = am$, $m = rs$ and $m = l + \nu$.

In the process of coding we apply the encryption algorithm, given in Figure 2.1, on the same redundant message $L$ four times using different parameters (different keys or quasigroups) and we obtain the codeword $C$ of the message as concatenation of the four codewords of $N/4$ bits.

### 2.3.2 Decoding with 4-Sets-Cut-Decoding algorithms

In [92], the authors suggest 4 different versions of 4-Sets-Cut-Decoding algorithms. The best results are obtained using the 4-Sets-Cut-Decoding algorithms #3. In our experiments we use only this version and here we will briefly explain it. After transmitting through a noisy channel, we divide the outgoing message $D = D^{(1)}D^{(2)}...D^{(s)}$ in four messages $D^1 = D^{(1)}D^{(2)}...D^{(s/4)}$, $D^2 = D^{(s/4+1)}D^{(s/4+2)}...D^{(s/2)}$, $D^3 = D^{(s/2+1)}D^{(s/2+2)}...\ D^{(3s/4)}$ and $D^4 = D^{(3s/4+1)}D^{(3s/4+2)}...D^{(s)}$ with equal lenghts and they are decoded parallelly with the corresponding parametars. Similarly, as in Cut-Decoding algorithm, in each iteration of the decoding process the decoding candidate sets obtained in the four decoding processes are reduced, in this way. Let $S_i^{(1)}$, $S_i^{(2)}$, $S_i^{(3)}$ and $S_i^{(4)}$ are sets of decoding candidates obtained in $i^{th}$ itteration of the four parallel decoding processes, $i = 1, \ldots, s/4$. Let $V_1 = \{w_1w_2 \ldots w_{rai} | (\delta, w_1w_2 \ldots w_{rai}) \in S_i^{(1)}\}$,..., $V_4 = \{w_1w_2 \ldots w_{rai} | (\delta, w_1w_2 \ldots w_{rai}) \in S_i^{(4)}\}$ and $V = V_1 \cap V_2 \cap V_3 \cap V_4$. If $V = \emptyset$, then $V = (V_1 \cap V_2 \cap V_3) \cup (V_1 \cap V_2 \cap V_4) \cup (V_1 \cap V_3 \cap V_4) \cup (V_2 \cap V_3 \cap V_4)$. Then, from $S_i^{(1)}$, $S_i^{(2)}$, $S_i^{(3)}$ и $S_i^{(4)}$ are eliminated all elements whose second part does not match with some element from $V$.

After the last iteration, if all reduced decoding candidate sets $S_{s/4}^{(1)}$, $S_{s/4}^{(2)}$, $S_{s/4}^{(3)}$, $S_{s/4}^{(4)}$ have only one element with same second component $w_1 \ldots w_{ras/4}$, then $L = w_1 \ldots\ w_{ras/4}$ is the decoded redundant message. In this case, we say that we have a *successful decoding*. If the decoded message is not the correct message, then we say that an *undetected error* has occurred. If the reduced sets have more than one element, after the last iteration, then we have *more-candidate-error*. If we obtain $S_i^{(1)} = S_i^{(2)} = S_i^{(3)} = S_i^{(4)} = \emptyset$ in some iteration of decoding process with 4-Sets-Cut-Decoding algorithm, then the process will be stoped (*null-error* appears). But, if we obtain at least one nonempty set, in some iteration, then the decoding continues with the nonempty sets (the reduced sets are obtained only by intersection the non empty sets). To solve the problem of most unsuccessful decoding with *more-candidate-error*, the same heuristic as in Cut-Decoding is used.

# Chapter 3

# Experimental results for codes based on quasigroup

The experimental results obtained with Standard algorithm and Cut-Decoding algorithm for transmission through a Gaussian channel were presented in my master thesis and published in [76]. There, the performances of these codes were investigated. We conclude that the performances depend on the code parametars: pattern, key and quasigroup. In this section, we give the experimental results obtained with 4-Sets-Cut-Decoding algorithm and we compare these results with the corresponding results obtained with Cut-Decoding. First, we will briefly explain Gaussian data transmission channel.

## 3.1 Gaussian channel

The most significant continuous data transmission channel is Gaussian channel presented in Figure 3.1. It is a discrete time channel where the output in time $i$ is $Y_i$ and that output is obtained as the sum of the input $X_i$ and the noise $Z_i$. The noise is drawn of independent and identicaly distributed random variables $Z_i$ from a Gaussian distribution with variance $N$. We call it *Additive White Gaussian Noise* (AWGN), and the channel is called Gaussian (AWGN) channel.

In fact,

$$Y_i = X_i + Z_i, \quad Z_i \sim \mathcal{N}(0, N).$$

We assume that the noise $Z_i$ is independent of the signal $X_i$. This is a good model that can describe several types of communication channels. If the noise variance is zero, then the receiver receives the transmitted signal correctly. Because

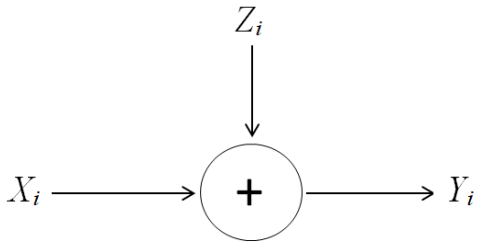



Figure 3.1: Gaussian channel

$X$ can receive any real value, the channel can transfer any real number without error.

If the noise variance is not zero and there is no input limit, we can choose an infinite subset of arbitrary inputs so that the resulting inputs will differ from each other with a low error probability. Such a scheme has infinite capacity [6].

The most common input limitation is power limitation. For any codeword $(x_1, x_2, ..., x_n)$ transmitted over the channel, we have:

$$\frac{1}{n}\sum x_i^2 \leqslant P$$

Suppose we want to send one bit through the channel. For the given power limit, the best we can do is send a signal whose strength is one of the two analog levels $+\sqrt{P}$ or $-\sqrt{P}$. The receiver sees the corresponding received signal $Y$ and tries to decide which of the two levels of the signal is sent to. Assuming that the two levels are equally likely (this is the case if we want to send exactly one bit of information), the optimal decoding rule is to decide that a signal with a power of $+\sqrt{P}$ is sent if $Y > 0$ or signal with strength $-\sqrt{P}$, if $Y < 0$.

Digital modulations are used to transfer data to mobile phones, scientific and geomagnetic instruments, etc. Any digital modulation scheme uses a finite number of different symbols to represent digital data. One of these modulations is PSK (Phase-shift keying). It uses a finite number of phases, each associated with a single pattern of binary digits. Usually each phase encodes an equal number of bits. Each bit model forms a symbol, which is represented by a certain phase. The demodulator, which is specially designed for the set of symbols used by the modulator, determines the phase of the received signal and replicates it back to the symbols that represent it, thus returning to the original data.

The simplest form of PSK modulator is BPSK (Binary Phase-shift keying) which uses only 2 phases. With BPSK, binary digits 1 and 0 can be represented by the analog levels $+\sqrt{E_b}$ and $-\sqrt{E_b}$, respectively. Here $P = E_b$ is a power limit.

This model is shown in Figure 3.2 ([98]).

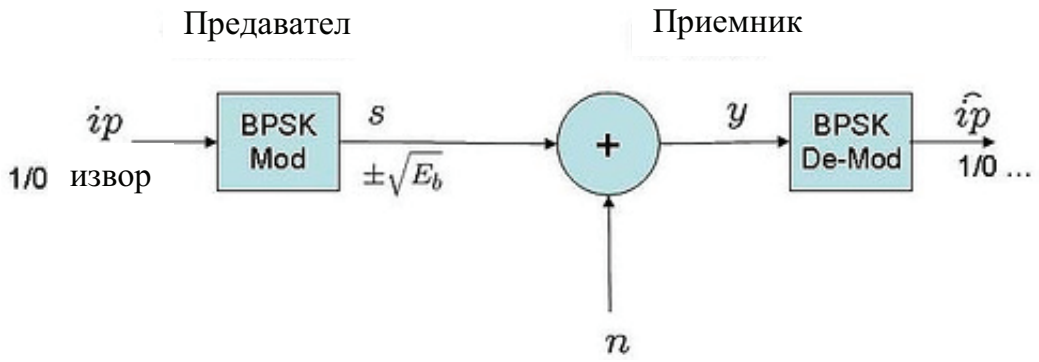



Figure 3.2: Simple block diagram with BPSK transmitter-receiver

The simulation of this modulation is performed through AWGN channel. However, the probability of bit error (Bit Error Rate, or short $BER$) is calculated by changing the value of the $SNR$ (Signal-to-noise-ratio) on the AWGN channel. So, the probability of bit-error $BER$ is a function of $SNR$.

Values for probability of bit-error $P_b$ for $SNR$ values in the range of $-3$ to $10$ decibels are shown in Table 3.1. From the table, it can be seen that the probability of bit-error $P_b$ decreases with an increase of $SNR$.

## 3.2 Choosing parameters for optimal RCBQ

Let briefly explain how the experiments in [76] were made and how we chose the parameters for optimal RCBQ obtained with Standard and Cut-Decoding algorithms. Experiments with RCBQ are performed as follows:

- First, the message received from the source is expanded by using a pattern to add redundant zero nibbles. Six different patterns were used.

| $SNR$ | $P_b$ |
|---|---|
| $-3$ | 0.1584 |
| $-2$ | 0.1306 |
| $-1$ | 0.1038 |
| 0 | 0.0786 |
| 1 | 0.0563 |
| 2 | 0.0375 |
| 3 | 0.0229 |
| 4 | 0.0125 |
| 5 | 0.0060 |
| 6 | 0.0024 |
| 7 | 0.000773 |
| 8 | 0.000191 |
| 9 | 0.0000336 |
| 10 | 0.00000387 |

Table 3.1: The probability of bit-error

- The extended message is coded using the encryption algorithm (Standard or Cut-Decoding algorithm).
- BPSK modulated is performed on the coded message , with $0 \rightarrow -1$ and $1 \rightarrow 1$.
- The signal is transmitted through the Gaussian channel in which there are noises and therefore the received signal at the output of the channel does not have to be the same as the input.
- Then, the output signal is demodulated as follows:
  1) if the received signal is greater than 0, then we assume that bit 1 is output at the input of the channel;
  2) if the received signal is less than 0, then we assume that bit 0 is output at the input of the channel;
- The demodulated message is decoded using the previously decoding rule.
- The received message is compared to the initial one and the probability of bit error $BER$ and the probability of packet error $PER$ are calculated.

The probability of packet error $PER$ on the communication channel is calculated as the ratio of the number of incorrectly transmitted blocks (packets) and the total number of blocks transmitted through the channel. Incorrectly transferred blocks appear in the following cases:

1. If there is only one element in the last set of candidates for decoding $S_s$, that element (decoded message) is compared with the input message. If they are equal, then we have an accurate decoding. If the decoded message differs in at least one bit from the input one, then an undetected error occurs and the packet is incorrectly transmitted, i.e., a packet error occurs.

2. Packet errors also occur in other types of unsuccessful decoding, i.e, *null-error and more-candidate error*.

The total number of incorrectly transmitted blocks is the sum of the incorrectly transmitted blocks in the previous two cases.

The bit-error probability $BER$ of the communication channel is calculated as the ratio of the number of incorrectly transmitted bits and the total number of bits transmitted through the channel. The total number of incorrectly transmitted bits is the sum of the incorrectly transmitted bits in the following two cases:

1. Upon successful decoding, the decoded and input message are compared and the number of bits in which they differ is determined.

2. In case of unsuccessful decoding, the following two cases are considered:

   2.1. When *null-error* appears, one of the sets $S_i = \emptyset$. In that case, all elements of the set $S_{i-1}$ are taken and the maximum common prefix of its elements is found. Let that prefix have length $k$. This common prefix is then compared to the first $k$ bits of the input message (which is $m$ bits long). If they differ in $t$ bits, then the number of incorrectly transmitted bits is $m - k + t$.

   2.2 If *more-candidate error* occurs, then we again find the maximum common prefix from the elements of the last set $S_s$ and the procedure for determining the number of incorrectly transmitted bits is performed in the same way as *null-error*.

In order to investigate the influence of the pattern on the code performance, in my master thesis, experiments were performed on several different patterns to add

redundant zero nibbles. Experiments are performed for different values of $SNR$ in the range of $-3$ to $10$ decibels. From the made experiments, we found the patterns which give the best results (the smallest BER and PER) and we use these patterns for experiments in this PhD thesis.

Also, in order to check the influence of the key on the performance of the code, experiments were performed with different key lengths. Analyzing the obtained results (in my master thesis) it is concluded that the key with length 10 gives the best results, so that key length is taken as a parameter in the construction of the optimal RCBQ.

To test whether quasigroup selection affects code performance, experiments were performed with a cyclic quasigroup of order 16 (for a key of length 10). Then, experiments were performed with a quasigroup of order 16 obtained as a direct product of quasigroups of order 2. Experimental results obtained with this quasigroup are worse than the results obtained with the cyclic quasigroup. The cyclic quasigroup and the quasigroup obtained by direct product of quasigroups of order 2 are examples of fractal quasigroups. On the other hand, the quasigroup given in Table 3.2 is an example of a nonfractal quasigroup and the results obtained with this quasigroup are quite satisfactory. Also, the given quasigroup is shapeless, which is good for cryptographic purposes. From the previous examples, it can be concluded that the choice of the quasigroup has a huge impact on the performance of the code [76].

## 3.3 Experimental results with 4-Sets-Cut-Decoding algorithm

We made many experiments for 4-Sets-Cut-Decoding algorithm and we found the optimal parametars at the same way which is explained in Section 3.2. The best results for code $(72, 576)$ with rate $R = 1/8$ are obtained with the following parameters:

- In Cut-Decoding - redundancy pattern: 1100 1100 1000 0000 1100 1000 1000 0000 1100 1100 1000 0000 1100 1000 1000 0000 0000 0000 for rate $1/4$ and two different keys of 10 nibbles.

- In 4-Sets-Cut-Decoding - redundancy pattern: 1100 1110 1100 1100 1110 1100 1100 1100 0000 for rate $1/2$ and four different keys of 10 nibbles.

- In all experments we used the same quasigroup on $Q$ given in Table 3.2.

Table 3.2: Quasigroup of order 16 used in the experiments

| $*$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | $a$ | $b$ | $c$ | $d$ | $e$ | $f$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 3 | $c$ | 2 | 5 | $f$ | 7 | 6 | 1 | 0 | $b$ | $d$ | $e$ | 8 | 4 | 9 | $a$ |
| 1 | 0 | 3 | 9 | $d$ | 8 | 1 | 7 | $b$ | 6 | 5 | 2 | $a$ | $c$ | $f$ | $e$ | 4 |
| 2 | 1 | 0 | $e$ | $c$ | 4 | 5 | $f$ | 9 | $d$ | 3 | 6 | 7 | $a$ | 8 | $b$ | 2 |
| 3 | 6 | $b$ | $f$ | 1 | 9 | 4 | $e$ | $a$ | 3 | 7 | 8 | 0 | 2 | $c$ | $d$ | 5 |
| 4 | 4 | 5 | 0 | 7 | 6 | $b$ | 9 | 3 | $f$ | 2 | $a$ | 8 | $d$ | $e$ | $c$ | 1 |
| 5 | $f$ | $a$ | 1 | 0 | $e$ | 2 | 4 | $c$ | 7 | $d$ | 3 | $b$ | 5 | 9 | 8 | 6 |
| 6 | 2 | $f$ | $a$ | 3 | $c$ | 8 | $d$ | 0 | $b$ | $e$ | 9 | 4 | 6 | 1 | 5 | 7 |
| 7 | $e$ | 9 | $c$ | $a$ | 1 | $d$ | 8 | 6 | 5 | $f$ | $b$ | 2 | 4 | 0 | 7 | 3 |
| 8 | $c$ | 7 | 6 | 2 | $a$ | $f$ | $b$ | 5 | 1 | 0 | 4 | 9 | $e$ | $d$ | 3 | 8 |
| 9 | $b$ | $e$ | 4 | 9 | $d$ | 3 | 1 | $f$ | 8 | $c$ | 5 | 6 | 7 | $a$ | 2 | 0 |
| $a$ | 9 | 4 | $d$ | 8 | 0 | 6 | 5 | 7 | $e$ | 1 | $f$ | 3 | $b$ | 2 | $a$ | $c$ |
| $b$ | 7 | 8 | 5 | $e$ | 2 | $a$ | 3 | 4 | $c$ | 6 | 0 | $d$ | $f$ | $b$ | 1 | 9 |
| $c$ | 5 | 2 | $b$ | 6 | 7 | 9 | 0 | $e$ | $a$ | 8 | $c$ | $f$ | 1 | 3 | 4 | $d$ |
| $d$ | $a$ | 6 | 8 | 4 | 3 | $e$ | $c$ | $d$ | 2 | 9 | 1 | 5 | 0 | 7 | $f$ | $b$ |
| $e$ | $d$ | 1 | 3 | $f$ | $b$ | 0 | 2 | 8 | 4 | $a$ | 7 | $c$ | 9 | 5 | 6 | $e$ |
| $f$ | 8 | $d$ | 7 | $b$ | 5 | $c$ | $a$ | 2 | 9 | 4 | $e$ | 1 | 3 | 6 | 0 | $f$ |

In the experiments with these algorithms we use $B_{max} = 5$ . In Table 3.3, we present experimental results for bit-error probabilities $BER_{cut}$ and $BER_{4sets}$, obtained with Cut-Decoding algorithm and 4-Sets-Cut-Decoding algorithm, respectively, and the corresponding packet-error probabilities $PER_{cut}$ and $PER_{4sets}$. For $SNR$ smaller than $-1$, using of Cut-Decoding algorithm does not have sense since the values of bit-error probabilities are larger than the bit-error probability in the channel.

Table 3.3: Experimental results with R=$1/8$

| $SNR$ | $BER_{cut}$ | $BER_{4sets}$ | $PER_{cut}$ | $PER_{4sets}$ |
|---|---|---|---|---|
| $-2$ | / | 0.06548 | / | 0.11283 |
| $-1$ | 0.05591 | 0.02225 | 0.10491 | 0.03831 |
| 0 | 0.01232 | 0.00449 | 0.02376 | 0.00835 |
| 1 | 0.00224 | 0.00066 | 0.00425 | 0.00136 |
| 2 | 0.00037 | 0.00008 | 0.00058 | 0.00014 |

Analyzing the results in Table 3.3, we can conclude that for all values of $SNR$, results for $BER_{4sets}$ are better than the corresponding results of $BER_{cut}$. The

same conclusions can be derived for comparison of $PER_{4sets}$ and $PER_{cut}$.

## 3.4 Experimental results for images

Here, we present the experimental results obtained for transmission of images through a Gaussian channel for different values of signal-to-noise ratio ($SNR$).

In all experiments (for different values of $SNR$ in the channel) we consider the differences between transmitted and decoded image. Also, we compare experimentally obtained values for bit-error probability ($BER$) and packet-error probability ($PER$) and the duration of the decoding processes with both algorithms.

In all decoding algorithms for RCBQ, when a *null-error* appears, the decoding process ends earlier and only a part of the message is decoded. Therefore in the experiments with images we use the following solution. In the cases of a *null-error* (all reduced sets are empty in some iteration), we take the strings without redundant symbols from all elements in the sets from the previous iteration and find their maximal common prefix substring. If this substring has $k$ symbols then in order to obtain decoded message of $l$ symbols we take these $k$ symbols and add $l - k$ zero symbols at the end of the message.

We present and compare the results obtained using both algorithms for RCBQ with modifications for reducing the number of unsuccessful decoding.

All experiments are for code $(72, 576)$ with rate $R = 1/8$, $B_{max} = 5$, different values of $SNR$, using the same parameters as in Section 3.3.

In the experiments for image transmission we use the image of "Lenna", given in Fig. 3.3 a). In the same figure, the second image is encrypted image of Lenna (using encryption algorithm given in Section 2.1.2) before transmitting through a Gaussian channel. It is evident that the algorithm crypts the images. After that, the image is transmitted through the channel (with different values of $SNR$) and the corresponding decoding algorithm is applied. In Fig. 3.4 – Fig. 3.8, we present images obtained for $SNR = -2$, $SNR = -1$, $SNR = 0$ and $SNR = 1$, correspondingly. In each figure, the first image is obtained after transmission through the channel without using any error-correcting code, the second image is obtained using Cut-Decoding algorithm and the third one – using 4-Sets-Cut-Decoding algorithm.

From the figures (first and third image in Fig. 3.9 – Fig. 3.12), we can see that Cut-Decoding and 4-Sets-Cut-Decoding algorithms correct many errors appeared during transmission. Also, we obtain less damages (lines) of the images with 4-Sets-Cut-Decoding algorithm than with Cut-Decoding algorithm.

a) b)

Figure 3.3: Original and encrypted image

Figure 3.4: $SNR = -3$

Figure 3.5: $SNR = -2$

The values of $BER$ and $PER$ obtained with both algorithms (presented in Table 3.4 and Table 3.5) confirm our conclusions from images. There,$BER_{cut}$ and $PER_{cut}$ denote the probabilities for Cut-Decoding algorithm and $BER_{4-sets}$ and $PER_{4-sets}$ – corresponding probabilities obtained by 4-Sets-Cut-Decoding

Figure 3.6: $SNR = -1$

Figure 3.7: $SNR = 0$

Figure 3.8: $SNR = 1$

algorithm. From these tables we can see that for all values of $SNR$, $BER_{4-sets}$ is more than 3 times smaller than $BER_{cut}$. Also, the packet-error probabilities obtained with 4-Sets-Cut-Decoding algorithm are more than 2 times smaller than the probabilities obtained with Cut-Decoding algorithm.

As we explain before when *null-error* appears we add $l - k$ zero symbols at the end of the message and this makes horizontal black lines on the image. The

Table 3.4: Experimental results for $BER$

| $SNR$ | $BER_{cut}$ | $BER_{4-sets}$ |
|---|---|---|
| $-3$ | 0.31351 | 0.10411 |
| $-2$ | 0.13257 | 0.03487 |
| $-1$ | 0.04029 | 0.01233 |
| 0 | 0.00990 | 0.00306 |
| 1 | 0.00161 | 0.00040 |

Table 3.5: Experimental results for $PER$

| $SNR$ | $PER_{cut}$ | $PER_{4-sets}$ |
|---|---|---|
| $-3$ | 0.52898 | 0.24045 |
| $-2$ | 0.24265 | 0.08019 |
| $-1$ | 0.07429 | 0.02622 |
| 0 | 0.01675 | 0.00645 |
| 1 | 0.00316 | 0.00082 |

horizontal white and gray lines are obtained in the case of *more-candidate-error* when the randomly selected message from the reduced sets in the last iteration differs from the original message.

On the other hand, the images obtained without using error-correcting codes do not have these lines, but the entire images have points that are incorrectly transmitted symbols.

In Table 3.6 and Table 3.7, probabilities of *null-error* ($PER_{null}$) and *more-candidate-error* ($PER_{more-candidate}$) are given. From these tables we can conclude that with Cut-Decoding algorithm we obtain a much greater number of unsuccessful decodings with *null-error* than with 4-Sets-Cut-Decoding algorithm, but the number of *more-candidate-errors* is smaller. Therefore, the images decoded with Cut-Decoding algorithm have more black lines than the images decoded with 4-Sets-Cut-Decoding algorithm. Also, the most of the lines in the third images (obtained with 4-Sets-Cut-Decoding algorithm) are white or gray.

Also, we analyzed the speed of two algorithms and concluded that 4-Sets-Cut-Decoding algorithm is faster than Cut-Decoding algorithm. The speed of the algorithms depends on the value of $SNR$ and it increases as $SNR$ increases. For

Table 3.6: Experimental results with Cut-Decoding

| $SNR$ | $PER_{null}$ | $PER_{more-candidate}$ |
|---|---|---|
| $-3$ | 0.52719 | 0.00096 |
| $-2$ | 0.24155 | 0.00041 |
| $-1$ | 0.07402 | 0.00027 |
| 0 | 0.01675 | 0 |
| 1 | 0.00316 | 0 |

Table 3.7: Experimental results with 4-Sets-Cut-Decoding

| $SNR$ | $PER_{null}$ | $PER_{more-candidate}$ |
|---|---|---|
| $-3$ | 0.07278 | 0.10588 |
| $-2$ | 0.02691 | 0.03131 |
| $-1$ | 0.01181 | 0.00796 |
| 0 | 0.00247 | 0.00288 |
| 1 | 0.00041 | 0.00027 |

example, for $SNR = 1$, 4-Sets-Cut-Decoding algorithm is two times faster than Cut-Decoding algorithm, and for $SNR = -2$, it is 16 times faster. In order to clear some of damages (horizontal lines) on the images, in the next subsection we propose a filter that visually enhance pixels damaged by *null-errors* and *more-candidate-errors*.

### 3.4.1 Filter for images decoded by cryptcodes based on quasigroups

For repairing of a damage, a filter has to locate where it appears. Locating of the *null-errors* is easy since we add zero symbols in the place of the undecoded part of the message. In order to locate *more-candidate-errors* we change the decoding rule for these kind of errors. Instead of random selection of a message from the reduced sets in the last iteration, we take a message of all zero symbols as a decoded message. Now, one pixel is considered as damaged if it belongs in a zero sub-block with at least four consecutive zero nibbles. The basic idea in the definition of this filter is to replace damaged pixel intensity value with a new value taken over a

neighborhood of fixed size. In this process, we use the median of the nonzero gray values of the surrounding pixels, so our filter is a median filter.

For each damaged pixel in the position $(i, j)$, the filter uses the following algorithm:

1. Take a 3 x 3 region centered around the pixel $(i, j)$;

2. Sort the nonzero intensity values of the pixels in the region into ascending order;

3. Select the middle value (the median) as the new value of the pixel $(i, j)$.

In Fig. 3.9 – Fig. 3.12, we present images obtained with Cut-Decoding and 4-Sets-Cut-Decoding algorithm before and after application of the proposed filter for $SNR = -2$, $SNR = -1$, $SNR = 0$ and $SNR = 1$, correspondingly. In each figure, first two images are for Cut-Decoding (without and with the filter, correspondingly) and last two images for 4-Sets-Cut-Decoding algorithm (without and with the filter).

Figure 3.9: $SNR = -2$

Figure 3.10: $SNR = -1$

Figure 3.11: $SNR = 0$

Figure 3.12: $SNR = 1$

From the presented images we can notice that the proposed filter provides a great improvement of the images for all considered values of $SNR$. Also, this filter gives better results for the images obtained with Cut-Decoding algorithm than with 4-Sets-Cut-Decoding algorithm. The reason for this is the larger number of *undetected-errors* obtained with 4-Sets-Cut-Decoding algorithm.

# Chapter 4

# Fast-Cut-Decoding and Fast-4-Sets-Cut-Decoding algorithms

In this chapter, we propose new coding/decoding algorithms for RCBQ called Fast-Cut-Decoding and Fast-4-Sets-Cut-Decoding algorithms. Our goal is increasing the decoding speed and decreasing the probability of bit-error. As we mention previously, the decoding with Cut-Decoding and 4-Sets-Cut-Decoding algorithms is actually list decoding. Therefore, the speed of decoding process depends on the list size (a shorter list gives faster decoding). In both algorithms, the list size depends on $B_{max}$ (the maximal assumed number of bit errors in a block). For smaller values of $B_{max}$, shorter lists are obtained. But, we do not know in advance how many errors appear during transmission of a block. If this number of errors is larger than assumed number of bit errors $B_{max}$ in a block, the errors will not be corrected. On the other side, if $B_{max}$ is too large, we have long lists and the process of decoding is too slow. Also, larger value of $B_{max}$ can lead to ending of the decoding process with *more-candidate-error* (the correct message will be in the list of the last iteration, if there are no more than $B_{max}$ errors during transmission). Therefore, with all decoding algorithms for RCBQ, *more-candidate-errors* can be obtained, although the bit-error probability of the channel is too small and the number of bit errors in a block is not greater than $B_{max}$ (or no errors during transmission). In order to solve this problem, we propose new modifications of Cut-Decoding and 4-Sets-Cut-Decoding algorithms, called Fast-Cut-Decoding and Fast-4-Sets-Cut-Decoding algorithms.

Here, instead of fixed value $B_{max}$ in the both algorithms, we start with $B_{max} = 1$. If we have successful decoding, the procedure is done. If not, we increase the value of $B_{max}$ with $1$ and repeat the decoding process with new value of $B_{max}$, etc. The decoding finishes with $B_{max} = 4$ (for rate $1/4$) or with $B_{max} = 5$ (for

rate $1/8$).

The new algorithms try to decode the message using the shorter lists and in the case of successful decoding with small value of $B_{max}$ ($B_{max} < 4$), we avoid long lists and slower decoding. Also, we decrease the number of *more-candidate-errors*.

## 4.1 Experimental results

In this section, we will present the experimental results for new fast algorithms and compare them with the existing ones.

## 4.2 Experimental results for Fast-Cut-Decoding and Fast-4-Sets-Cut-Decoding algorithms

In this section, we give the experimental results obtained with new Fast-Cut-Decoding and Fast-4-Sets-Cut-Decoding algorithms for rate $R = 1/4$ and $R = 1/8$, for different values of $SNR$ in Gaussian channel. We compare these results with corresponding results obtained with Cut-Decoding and 4-Sets-Cut-Decoding algorithms. In order to show the efficiency of new algorithms we present percentages of messages which decoding finished with $B_{max} = 1, 2, 3, 4$ or $5$.

First we present results for code $(72, 288)$ with rate $R = 1/4$. For this code, we made experiments with Cut-Decoding algorithm and Fast-Cut-Decoding algorithm using the following parametars:

- Redundancy pattern: 1100 1100 1000 0000 1100 1000 1000 0000 1100 1100 1000 0000 1100 1000 1000 0000 0000 0000 for rate $1/2$
- Two different keys of 10 nibbles, and
- Quasigroup on $(Q, *)$ given in Table 3.2 and the corresponding parastrophe.

In experiments with Cut-Decoding algorithm we use $B_{max} = 4$ and for Fast-Cut-Decoding algorithm the maximum value of $B_{max}$ is 4.

In Table 4.1, we give experimental results for bit-error probabilities $BER_{cut}$ and packet-error probabilities $PER_{cut}$ (obtained with Cut-Decoding algorithm) and the corresponding probabilities $BER_{f-cut}$ and $PER_{f-cut}$ (obtained with Fast-Cut-Decoding algorithm). The results for $BER_{cut}$ and $PER_{cut}$ for Cut-Decoding algorithm are given in Chapter 3. In this part, the authors concluded that for values

of $SNR$ smaller than 0, the coding does not have sense since the bit-error probability obtained with Cut-Decoding algorithm is larger than the bit-error probability in the channel (without coding).

Table 4.1: Experimental results with $R = 1/4$

| $SNR$ | $BER_{cut}$ | $BER_{f-cut}$ | $PER_{cut}$ | $PER_{f-cut}$ |
|---|---|---|---|---|
| 0 | 0.07153 | 0.06122 | 0.10001 | 0.08993 |
| 1 | 0.01830 | 0.01493 | 0.02722 | 0.02275 |
| 2 | 0.00249 | 0.00155 | 0.00410 | 0.00274 |
| 3 | 0.00073 | 0.00006 | 0.00230 | 0.00014 |
| 4 | 0.00052 | 0.00001 | 0.00252 | 0.00007 |

Analyzing the results in Table 3.3, we can conclude that for all values of $SNR$, results for $BER_{f-cut}$ are better than the corresponding results of $BER_{cut}$. For $SNR = 4$, $BER_{f-cut}$ is even 50 times smaller than $BER_{cut}$. The same conclusions can be derived by comparison of $PER_{f-cut}$ and $PER_{cut}$.

In Table 4.2, we present the percentage of messages which decoding ended with $B_{max} = 1$, $B_{max} = 2$, $B_{max} = 3$ or $B_{max} = 4$. From the results given there, we can see that for smaller values of $SNR$ (0 or 1), we have larger percentage of messages which decoding needed $B_{max} = 3$ or 4. On the other side, for $SNR = 4$, the decoding of more than 90% of the messages successfully finished with $B_{max} = 1$. From these results, we can conclude that for larger values of $SNR$ (low noise in the channel) decoding with the new proposed algorithm is much faster than the old one.

Table 4.2: Percentage of messages decoded with different values of $B_{max}$

| $SNR$ | $B_{max} = 1$ | $B_{max} = 2$ | $B_{max} = 3$ | $B_{max} = 4$ |
|---|---|---|---|---|
| 0 | 2.74% | 26.35% | 37.28% | 33.63% |
| 1 | 14.66% | 45.83% | 28.98% | 10.53% |
| 2 | 44.62% | 41.90% | 11.41% | 2.07% |
| 3 | 76.45% | 20.50% | 2.79% | 0.26% |
| 4 | 93.68% | 5.89% | 0.42% | 0.01% |

Further on, we present the results for code $(72, 576)$ with rate $R = 1/8$. We compare experimental results obtain with Cut-Decoding, Fast-Cut-Decoding, 4-Sets-Cut-Decoding, Fast-4-Sets-Cut-Decoding algorithms. In the experiments we used the following parameters:

- In Cut-Decoding, Fast-Cut-Decoding algorithms - redundancy pattern: 1100 1100 1000 0000 1100 1000 1000 0000 1100 1100 1000 0000 1100 1000 1000 0000 0000 0000 for rate $1/4$ and four different keys 10 nibbles.

- In 4-Sets-Cut-Decoding, Fast-4-Sets-Cut-Decoding algorithms - redundancy pattern: 1100 1110 1100 1100 1110 1100 1100 1100 0000 for rate $1/2$ and four different keys 10 nibbles.

- In all experments we used the same quasigroup on $Q$ given in Table 3.2.

Here, in the experiments with old algorithms we use $B_{max} = 5$ and for Fast-Cut-Decoding and Fast-4-Sets-Cut-Decoding algorithms the maximum value of $B_{max}$ is 5. In Table 4.3, we present experimental results for bit-error probabilities $BER_{cut}$, $BER_{f-cut}$, $BER_{4sets}$, $BER_{f-4sets}$ obtained with Cut-Decoding algorithm, Fast-Cut-Decoding algorithm, 4-Sets-Cut-Decoding algorithm and Fast-4-Sets-Cut-Decoding algorithm, respectively, and the corresponding packet-error probabilities $PER_{cut}$, $PER_{f-cut}$, $PER_{4sets}$, $PER_{f-4sets}$. For $SNR$ smaller than $-1$, using of Cut-Decoding algorithm does not have sense since the values of bit-error probabilities are larger than the bit-error probability in the channel.

In Table 4.4, we give the percentage of messages which decoding ended with $B_{max} = 1$, $B_{max} = 2$, $B_{max} = 3$, $B_{max} = 4$ or $B_{max} = 5$ for both new algorithms.

From the results given in Table 4.3 and Table 4.4, we can derive similar conclusions for rate $1/8$ as for rate $1/4$. Namely, we can conclude that for all values of $SNR$, the results for $BER$ and $PER$ obtained with the new algorithms are better than the corresponding results obtained with the old versions of these algorithms. Again, this improvement is more significant for the larger value of $SNR$, i.e., for low noise in the channel. Even more, for $SNR \geqslant 2$, the values for $BER$ and $PER$ obtained with the new algorithms are equal to 0.

From Table 4.4, we can see that if the values of $SNR$ are smaller then with new Fast-4-Sets-Cut-Decoding algorithm, we have larger percentage of messages which decoding needed $B_{max} = 4$ or 5. On the other side, for $SNR = 2$, decoding of more than 80% of the messages successfully finished with $B_{max} = 1$ or $B_{max} = 2$. Therefore, we can conclude that for larger values of $SNR$ (low noise in the channel) decoding with the new proposed algorithms is much faster than

Table 4.3: Experimental results with R=$1/8$

| $SNR$ | $BER_{cut}$ | $BER_{f-cut}$ | $BER_{4sets}$ | $BER_{f-4sets}$ |
|---|---|---|---|---|
| $-2$ | / | / | 0.06548 | 0.04905 |
| $-1$ | 0.05591 | 0.03920 | 0.02225 | 0.00795 |
| 0 | 0.01232 | 0.00872 | 0.00449 | 0.00074 |
| 1 | 0.00224 | 0.00069 | 0.00066 | 0.00013 |
| 2 | 0.00037 | 0 | 0.00008 | 0 |
| $SNR$ | $PER_{cut}$ | $PER_{f-cut}$ | $PER_{4sets}$ | $PER_{f-4sets}$ |
| $-2$ | / | / | 0.11283 | 0.09086 |
| $-1$ | 0.10491 | 0.07171 | 0.03831 | 0.01656 |
| 0 | 0.02376 | 0.01598 | 0.00835 | 0.00151 |
| 1 | 0.00425 | 0.00187 | 0.00136 | 0.00021 |
| 2 | 0.00058 | 0 | 0.00014 | 0 |

with the old one. The results from this part are published in [86].

## 4.3 Experimental results for images

In this section, we investigate the performances of fast algorithms for transmission of images. For this goal, we made many experiments for transmission in Gaussian channel and RCBQ as an error-correcting code. We compare the results obtained using four algorithms for RCBQ: Cut-Decoding, Fast-Cut-Decoding, 4-Sets-Cut-Decoding and Fast-4-Sets-Cut-Decoding with modifications for reducing the number of unsuccessful decodings. All experiments are made for code $(72, 576)$ with rate $R = 1/8$ and different values of $SNR$ in Gaussian channel.

In the experiments we used the same parametars (given in previou section) as for ordinary messages.

Here, in the experiments with old algorithms we use $B_{max} = 5$ and for Fast-Cut-Decoding and Fast-4-Sets-Cut-Decoding algorithms the maximum value of $B_{max}$ is 5. In the case of *null-error* or *more-candidate-error*, we apply the same heuristic as in Section 3.3.

As previous, in the experiments for image transmission we use the image of ”Lenna”, given in Fig. 3.3 a). The image is transmitted through the channel (with

Table 4.4: Percentage of messages decoded with different values of $B_{max}$

| | Fast-Cut-Decoding | | | | |
|---|---|---|---|---|---|
| $SNR$ \ $B_{max}$ | 1 | 2 | 3 | 4 | 5 |
| $-2$ | / | / | / | / | / |
| $-1$ | 0% | 1.57% | 25.32% | 44.46% | 28.64% |
| 0 | 0.07% | 13.80% | 49.38% | 27.88% | 8.86% |
| 1 | 1.71% | 48.12% | 39.16% | 9.43% | 1.58% |
| 2 | 18.84% | 65.41% | 13.88% | 1.68% | 0.18% |
| | Fast-4-Sets-Cut-Decoding | | | | |
| $SNR$ \ $B_{max}$ | 1 | 2 | 3 | 4 | 5 |
| $-2$ | 0% | 1.14% | 7.68% | 51.39% | 39.79% |
| $-1$ | 0% | 6.99% | 32.13% | 49.54% | 11.02% |
| 0 | 3.39% | 28.74% | 48.76% | 17.62% | 1.48% |
| 1 | 20.24% | 51.47% | 25.88% | 2.33% | 0.09% |
| 2 | 57.86% | 37.67% | 4.37% | 0.10% | 0.01% |

different values of $SNR$) and the corresponding decoding algorithm is applied. In Fig. 4.1 – Fig. 4.4, we present images obtained for $SNR = -2$, $SNR = -1$, $SNR = 0$ and $SNR = 1$, correspondingly. In each figure, the first image is obtained using Cut-Decoding, the second image is obtained using Fast-Cut-Decoding algorithm, the third image using 4-Sets-Cut-Decoding algorithm and the fourth one using Fast-4-Sets-Cut-Decoding algorithm .

Figure 4.1: $SNR = -2$

Figure 4.2: $SNR = -1$

Figure 4.3: $SNR = 0$

Figure 4.4: $SNR = 1$

If we compare the images, we can conclude that the images obtained with Fast-Cut-Decoding algorithm and Fast-4-Sets-Cut-Decoding algorithm are clearer than the corresponding images obtained with Cut-Decoding algorithm and 4-Sets-Cut-Decoding algorithm. Also, the images obtained with Fast-4-Sets-Cut-Decoding algorithm are clearer than the corresponding images obtained with Fast-Sets-Cut-Decoding algorithm for all values of $SNR$.

In Table 4.5, we present experimental results for bit-error probabilities $BER_{cut}$, $BER_{f-cut}$, $BER_{4sets}$, $BER_{f-4sets}$ obtained with Cut-Decoding algorithm, Fast-

Cut-Decoding algorithm, 4-Sets-Cut-Decoding algorithm and Fast-4-Sets-Cut-Decoding algorithm, respectively, and the corresponding packet-error probabilities $PER_{cut}$, $PER_{f-cut}$, $PER_{4sets}$, $PER_{f-4sets}$.

Table 4.5: Experimental results with $R = 1/8$

| $SNR$ | $BER_{cut}$ | $BER_{f-cut}$ | $BER_{4sets}$ | $BER_{f-4sets}$ |
|---|---|---|---|---|
| $-2$ | 0.13257 | 0.14189 | 0.03487 | 0.04886 |
| $-1$ | 0.04029 | 0.03947 | 0.01233 | 0.00823 |
| 0 | 0.00990 | 0.00594 | 0.00306 | 0.00062 |
| 1 | 0.00161 | 0.00140 | 0.00040 | 0 |
| $SNR$ | $PER_{cut}$ | $PER_{f-cut}$ | $PER_{4-sets}$ | $PER_{f-4sets}$ |
| $-2$ | 0.24265 | 0.24952 | 0.08019 | 0.09063 |
| $-1$ | 0.07429 | 0.07127 | 0.02622 | 0.01634 |
| $-0$ | 0.01675 | 0.01332 | 0.00645 | 0.00151 |
| 1 | 0.00316 | 0.00307 | 0.00082 | 0 |

In Table 4.6, we present the percentage of messages which decoding ended with $B_{max} = 1$, $B_{max} = 2$, $B_{max} = 3$, $B_{max} = 4$ or $B_{max} = 5$ with Fast-Cut-Decoding algorithm. The conclusions are very similar as for ordinary messages. From the results given there, we can see that for greater values of $SNR$ (1 or 2), we have larger percentage of messages which decoding needed $B_{max} = 2$. From these results, we can conclude that for greater values of $SNR$ (low noise in the channel) decoding with the new proposed algorithm is much faster than the old one.

Table 4.6: Percentage of messages decoded with different values of $B_{max}$

| $SNR$ | $B_{max} = 1$ | $B_{max} = 2$ | $B_{max} = 3$ | $B_{max} = 4$ | $B_{max} = 5$ |
|---|---|---|---|---|---|
| $-2$ | 0% | 0.12% | 6.04% | 33.92% | 59.91% |
| $-1$ | 0% | 1.61% | 25.89% | 43.27% | 29.24% |
| 0 | 0.08% | 14.16% | 48.15% | 28.93% | 8.68% |
| 1 | 1.99% | 47.68% | 39.29% | 9.37% | 1.68% |
| 2 | 18.76% | 66.03% | 13.43% | 1.68% | 0.07% |

In Table 4.7, we present the percentage of messages which decoding ended with $B_{max} = 1$, $B_{max} = 2$, $B_{max} = 3$, $B_{max} = 4$ $B_{max} = 5$ with Fast-4-Sets-Cut-Decoding algorithms. From the results given there, we can see that for greater values of $SNR$ (1 or 2), we have larger percentage of messages which decoding needed $B_{max} = 1$ or 2. This means that for greater values of $SNR$ (low noise in the channel) decoding with the new proposed algorithm is much faster than the old one.

Table 4.7: Percentage of messages decoded with different values of $B_{max}$

| $SNR$ | $B_{max} = 1$ | $B_{max} = 2$ | $B_{max} = 3$ | $B_{max} = 4$ | $B_{max} = 5$ |
|---|---|---|---|---|---|
| $-2$ | 0.03% | 1.15% | 7.84% | 51.11% | 39.87% |
| $-1$ | 0.36% | 7.28% | 32.74% | 48.81% | 10.82% |
| 0 | 3.53% | 28.00% | 49.77% | 17.19% | 1.51% |
| 1 | 20.54% | 51.37% | 25.84% | 2.14% | 0.10% |
| 2 | 57.79% | 37.70% | 4.49% | 0.03% | 0% |

With these new algorithms, Fast-Cut-Decoding and Fast-4-Sets-Cut-Decoding algorithms, we obtained better results for packet-errors and bit-error probabilities than with the previously defined (Cut-Decoding and 4-Sets-Cut-Decoding) algorithms of RCBQ, for transmission through a Gaussian channels. Also, from the presented percentages of messages which decoding ends with different values of $B_{max}$ we can conclude that for larger value of $SNR$ Fast-Cut-Decoding and Fast-4-Sets-Cut-Decoding algorithms provide faster decoding.

## 4.4 Experimental results for audio files

In this section, we investigate the performances of fast algorithms for transmission of audio files. In our experiments we use Gaussian channel and RCBQ as an error-correcting code. Similarly, as for images, we compare the results for code $(72, 576)$ with rate $R = 1/8$ obtained using four algorithms for RCBQ: Cut-Decoding, Fast-Cut-Decoding, 4-Sets-Cut-Decoding and Fast-4-Sets-Cut-Decoding and different values of $SNR$ in Gaussian channel. In these experiments we use the audio that is consisted of one 16-bit channel with a sampling rate of 44100 $Hz$ and it is a part of Beethoven's "Ode to joy" with a total length of approximately 4.3 seconds.

In the experiments we used the same parametars as for ordinary messages.

Similary as for images, in the experiments with Cut-Decoding and 4-Sets-Cut-Decoding algorithms we use $B_{max} = 5$ and for Fast algorithms the maximum value of $B_{max}$ is 5.

In all decoding algorithms for RCBQ, when a null-error appears, the decoding process ends earlier and only a part of the message is decoded. Therefore in the experiments with audio files we use the following solution. In the cases when a null-error appears, i.e., all reduced sets are empty in some iteration, we take the strings without redundant symbols from all elements in the sets from the previous iteration and we find their maximal common prefix substring. If this substring has $k$ symbols then in order to obtain decoded message of $l$ symbols we take these $k$ symbols and add $l - k$ zero symbols at the end of the message.

In Table 4.8, we present experimental results for bit-error probabilities $BER_{cut}$, $BER_{f-cut}$, $BER_{4sets}$, $BER_{f-4sets}$ obtained with Cut-Decoding algorithm, Fast-Cut-Decoding algorithm, 4-Sets-Cut-Decoding algorithm and Fast-4-Sets-Cut-Decoding algorithm, respectively and the corresponding packet-error probabilities $PER_{cut}$, $PER_{f-cut}$, $PER_{4sets}$, $PER_{f-4sets}$. For $SNR$ smaller than $-1$, the usage of Cut-Decoding algorithm does not have sense since the values of bit-error probabilities are larger than the bit-error probability in the channel.

Table 4.8: Experimental results for $BER$ and $PER$

| $SNR$ | $BER_{cut}$ | $BER_{f-cut}$ | $BER_{4sets}$ | $BER_{f-4sets}$ |
|---|---|---|---|---|
| $-2$ | / | / | 0.03658 | 0.04782 |
| $-1$ | 0.04741 | 0.04019 | 0.01122 | 0.00825 |
| 0 | 0.01114 | 0.00713 | 0.00281 | 0.00081 |
| 1 | 0.00175 | 0.00086 | 0.00052 | 0.00003 |
| $SNR$ | $PER_{cut}$ | $PER_{f-cut}$ | $PER_{4sets}$ | $PER_{f-4sets}$ |
| $-2$ | / | / | 0.08451 | 0.08917 |
| $-1$ | 0.08623 | 0.07589 | 0.02499 | 0.01644 |
| 0 | 0.02208 | 0.01418 | 0.00632 | 0.00165 |
| 1 | 0.00383 | 0.00177 | 0.00113 | 0.00012 |

From the experimental results given in Table 4.8 we can conclude that for all values of $SNR$, the results for $PER$ and $BER$ obtained with Fast algorithms are better than the results obtained with Cut-Decoding and 4-Sets-Cut-Decoding algorithms, especially for larger values of $SNR$ (low noise). Even more, for $SNR \geqslant 2$

bit-error and packet-error probabilities obtained with Fast algorithms are $0$. Also, decoding with these new algorithms is much faster than with the old one.

For all experiments, we also consider the differences between the sample values of the original and decoded signal. We present these analysis on graphs where the sample number in the sequence of samples consisting the audio signal is on the $x$-axes and the value of the sample is on the $y$-axis. In Fig. 4.5 the original audio samples are presented. Graphs for decoded audio files for considered values of $SNR$ are given in Fig. 4.6 - Fig. 4.9. In Fig. 4.6 (for $SNR = -2$) we present only two graphs, the first one is for 4-Sets-Cut-Decoding algorithm and the second is for Fast-4-Sets-Cut-Decoding algorithm, since the coding/decoding with Cut-Decoding algorithm does not have sense. Namely, the obtained value od $BER$ is larger than the bit-error in the channel. For the other values of $SNR$, in Fig. 4.7 - Fig. 4.9, we present 4 graphs (for decoded audio files using all 4 mentioned algorithms) in the following order: Cut -Decoding, Fast-Cut-Decoding, 4-Sets-Cut-Decoding and Fast-4-Sets-Cut-Decoding.

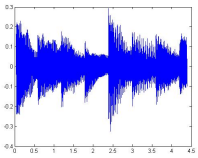


Figure 4.5: Original audio samples

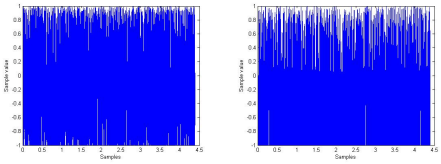


Figure 4.6: Results for $SNR = -2$

From these figures we can derive the same conclusions as from the tables for $PER$ and $BER$. From the graphs it is evident that for all values of $SNR$, the results obtained using 4-Sets-Cut-Decoding algorithm are better than the results obtained with Cut-Decoding algorithm and the results obtained with the new Fast algorithms are better than corresponding results obtained with the old versions of these algorithms.

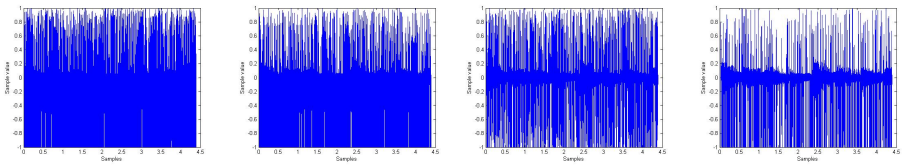


Figure 4.7: Results for $SNR = -1$

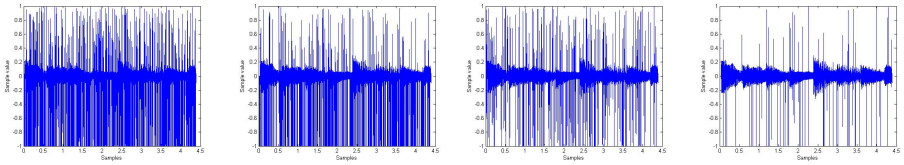


Figure 4.8: Results for $SNR = 0$

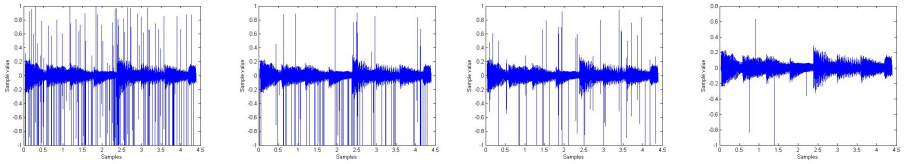


Figure 4.9: Results for $SNR = 1$

All audio files obtained in our experiments for transmission through a Gaussian channel with different $SNR$ can be found on the following link: `https://www.dropbox.com/sh/5zq5ly6qtiho8d6/AACTQBgUDopFq9psdbaMb8BKa?dl=0`.

If someone listened to these audio files, he/she would notice that as $SNR$ decreases, the noise increases, but the original melody can be listened completely in background. This is also evident on the graphs. If we compare the graphs for files decoded with all 4 algorithms with the graph for original audio samples we can see that the samples of the original audio are contained on all graphs. This is the reason why we can still hear the original melody in the background intermixed with the noisy sounds. In order to clear some of these noisy sounds, in the next subsection we propose a filter that can repair some of damages done by *null-errors* and *more-candidate-errors*.

### 4.4.1 Filter for enhancing the quality of audio decoded by crypt-codes based on quasigroups

For repairing of a damage, a filter has to locate where it appears. Locating of the *null-errors* is easy since we add zero symbols in the place of the undecoded part of the message. In order to locate *more-candidate-errors* we change the decoding rule for these kind of errors. Instead of random selection of a message from the reduced sets in the last iteration, we take a message of all zero symbols as a decoded message.

The basic idea in the definition of this filter is to replace damaged (erroneously decoded) nibbles with a new value derived from the values of several previous nibbles. So, we take all decoded messages as one list of nibbles and one nibble is considered as an erroneously decoded symbol if it belongs in a zero sub-list with at least four consecutive zero nibbles. Then each erroneously decoded symbol (nibble) we replace with the median of the previous $2k + 1$ nibbles in the list. If the erroneous nibble is at the beginning of the list and there are no $2k + 1$ previous ones, then a median of all previous nibbles (to the beginning of the list) is taken. For repairing of a nibble, we use only previous $2k+1$ nibbles since the next nibbles are zeros (erroneously decoded symbol belongs in a zero sub-list with at least four consecutive zero nibbles) and probably they are erroneously decoded. We made experiments for $2k + 1$ equal to $3$, $5$, $7$ and $9$ and the results were similar, but they are a little bit better for $2k+1$ equal to 7 or 9. Further on, we present results obtain with median of 7 previous nibbles. Notice that we take an odd number of previous nibbles since a median of these nibbles is computed and if this number is even the median can be a number which is not in $Q$.

In Fig. 4.10 – Fig. 4.13, we present graphs of samples of audio files obtained with Fast-Cut-Decoding and Fast-4-Sets-Cut-Decoding algorithm before and after using of the proposed filter for $SNR = -2$, $SNR = -1$, $SNR = 0$ and $SNR = 1$, correspondingly. In each figure, first two graphs are for Fast-Cut-Decoding (without and with the filter, correspondingly) and last two images for Fast-4-Sets-Cut-Decoding algorithm (without and with the filter). Also, audio files obtained after application of the proposed filter can be found on the following link: `https://www.dropbox.com/sh/5zq5ly6qtiho8d6/AACTQBgUDopFq9psdbaMb8BKa?dl=0`.

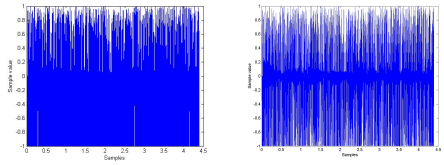


Figure 4.10: $SNR = -2$

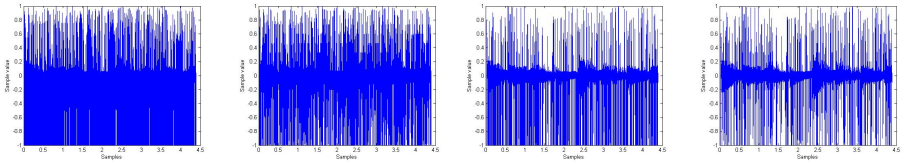


Figure 4.11: $SNR = -1$

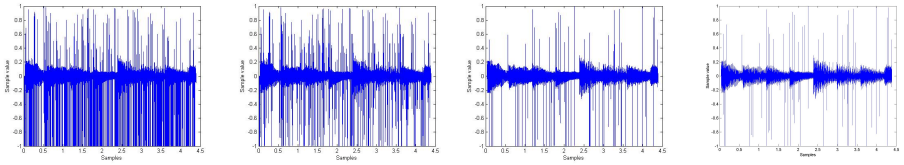


Figure 4.12: $SNR = 0$

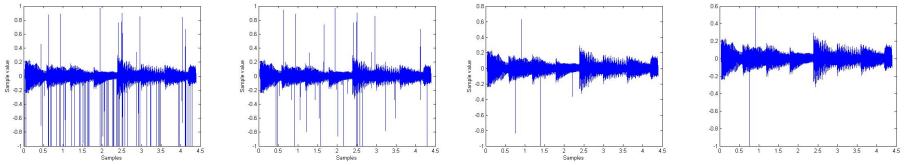


Figure 4.13: $SNR = 1$

From the given graphs and audio files we can notice that the proposed filter provides a great improvement of the audio for all considered values of $SNR$.

The results of this section are published in [78].

# Chapter 5

# Gilbert-Elliot channel

## 5.1 Gilbert-Elliott Burst channel

Gilbert-Elliott Burst channel is a channel model introduced by Edgar Gilbert and E. O. Elliott. This model is based on a Markov chain with two states: G (good or gap) and B (bad or burst). In the good state the probability for incorrect transmission of a bit is small, and in the bad state this probability is large. This model is widely used for describing burst error patterns in transmission channels, that enables simulations of the digital error performance of communications links. This model is shown in Fig. 5.1, where G represents the good state and B represents the bad state. The probability of moving from bad to good state is $P_{BG}$ and this probability from good to bad state is $P_{GB}$ ([57], [61]).

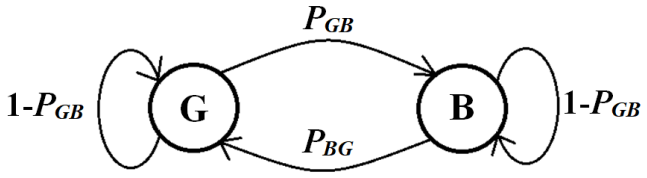



Figure 5.1: Gilbert-Elliott Burst Model

We made experiments with two kinds of Gilbert-Eliott channel. In the first one, in each state the channel is a binary symmetric with bit error probabilities $P_e(G)$ in the good state and $P_e(B)$ in the bad state. In the second one, the channels are

Gaussian where $SNR$ is high in the good state and low in the bad state.

## 5.2 New cryptcodes for burst channels

In experiments with burst channel, previously explained algorithms do not give good results and therefore we define a new algorithms for coding/decoding called Burst-Cut-Decoding and Burst-4-Sets-Cut-Decoding algorithms [79]. It is known that interleaver and deinterleaver are useful for reducing errors caused by burst errors in a communication system. Therefore, the new algorithms introduce an interliver in the encoding algorithm and an appropriate deinterliver in the decoding algorithm.

The encoding process in the new algorithms consists of the following steps:

– The input message $M = M_1M_2...M_l$ is expanded by adding redundant $\nu$ zero symbols (in the same way as in ADP and AD4P) and thus we obtain redundant message $L = L^{(1)}L^{(2)}...L^{(s/2)} = L_1L_2...L_{m/2}$ of $N/2$ bits (in ADP), i.e., $L = L^{(1)}L^{(2)}...L^{(s/4)} = L_1L_2...L_{m/4}$, of $N/4$ bits (in AD4P), where $L^{(i)}$ is a subblock of $r$ symbols from the alphabet $Q$, and $L_i \in Q$. Thereby, $N = am$, $m = rs$ and $m = l + \nu$.

– Then, for coding, we use twice (four times) the coding algorithm given in Figure 2.1, on the same redundant message $L$, using different parameters, thus we obtain two (four) codewords.

– We apply the interliver to each codeword individually. The interliver rearranges (in rows) the $m$-nibbles of the codeword into a $k \times (k/m)$ matrix. The output of the interliver is a mixed message obtained by reading this matrix in columns.

– Then, we concatenate two (four) outputs of the interliver and thus obtain the codeword for the input message $M$, i.e., $C = C_1C_{\frac{m}{k}+1}C_{\frac{2m}{k}+1}...C_2C_{\frac{m}{k}+2}$ $...C_{\frac{(k-1)m}{k}}C_m$ where $C_i \in Q$.

The decoding process consists of the following steps:

– The coded message is transmitted through a burst channel and we obtain the outgoing message $D = D^{(1)}D^{(2)}...D^{(s)}$, where $D^{(i)}$ are sub-blocks of $r$ symbols. The message $D$ is divided into two (four) messages with equal length.

– Before parallel decoding, the deinterliver is applied to each message individually. Specifically, each part of the outgoing message is rearranged (in rows) into $(k/m) \times k$, matrix and then the decoded messages are obtained by reading columns of the resulting matrices.

– These mixed messages are decoded with the appropriate RCBQ (ADP or AD4P) decoding algorithm.

The encoding/decoding process of the new schematic algorithms is presented in Figure 5.2.

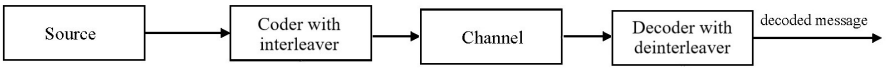



Figure 5.2: Coding/decoding with new algorithms

## 5.3 Experimental results for Gilbert-Elliott model

In this section we present the experimental results obtained with RCBQ for transmission through a burst channel. For simulation of the channel we use the Gilbert-Elliott model. We compare the values of packet-error probability ($PER$) and bit-error probability ($BER$) obtained with Cut-Decoding, 4-Sets-Cut-Decoding, Burst-Cut-Decoding and Burst-4-Sets-Cut-Decoding algorithms. We consider codes $(72, 288)$ with rate $1/4$ using Cut-Decoding algorithm and Burst-Cut-Decoding algorithm, and also $(72, 576)$ with rate $1/8$ using all algorithms (Cut-Decoding, 4-Sets-Cut-Decoding, Burst-Cut-Decoding and Burst-4-Sets-Cut-Decoding). In the experiments we use the following code parameters :

– for code $(72, 288)$ in Cut-Decoding and Burst-Cut-Decoding algorithm, the parameters are:

  – redundancy pattern: 1100 1110 1100 1100 1110 1100 1100 1100 0000 for rate $1/2$ and two different keys of 10 nibbles.

– for code $(72, 576)$ the code parameters are:

  – in Cut-Decoding/Burst-Cut-Decoding - redundancy pattern: 1100 1100 1000 00001100 1000 1000 0000 1100 1100 1000 0000 1100 1000 1000 0000 0000 0000, for rate $1/4$ and two different keys of 10 nibbles.

  – in 4-Sets-Cut-Decoding/Burst-4-Sets-Cut-Decoding - redundancy pattern: 1100 1110 1100 1100 1110 1100 1100 1100 0000 for rate $1/2$ and four different keys of 10 nibbles.

For all experiments we use the same quasigroup on $Q$ given in Table 3.2.

For new burst algorithms, we made experiments for different values of $k$, i.e., for all divisors of 36. Namely, a number of nibbles of two (or four) concatenated codewords in Cut-Decoding/Burst-Cut-Decoding (or 4-Sets-Cut-Decoding/Burst-4-Sets-Cut-Decoding) algorithms is 36 (or 72). Best results are obtained for $k = 9$. Further on, we will present only results for this value of $k$.

In Subsection 5.3.1. we present experimental results for Gilbert-Elliott model with binary symmetric channels for different values of bit-error probability and different transition probabilities. The experimental results for different values of $SNR$ and different transition probabilities in Gilbert-Elliott model with Gaussian channels, are given in Subsection 5.3.2.

### 5.3.1 Experiments for Gilbert-Elliott with BSC channels

In all experiments for Gilbert-Elliott model with binary symmetric channels we use $P_e(G) = 0.01$ for bit-error probabilities in the good state and some different values of bit error probabilities in the bad state $P_e(B) \in \{0.2; 0.16; 0.13; 0.1\}$.

In Table 5.1, we give experimental results for bit-error probabilities $BER_{cut}$ and packet-error probabilities $PER_{cut}$ (obtained with Cut-Decoding algorithm) and the corresponding probabilities $BER_{b-cut}$ and $PER_{b-cut}$ (obtained with Burst-Cut-Decoding algorithm) for code with rate $1/4$, and following combinations of transition probabilities from good to good state $P_{GG}$ and from bad to bad state $P_{BB}$:

– $P_{GG} = 0.8$ and $P_{BB} = 0.8$

– $P_{GG} = 0.5$ and $P_{BB} = 0.5$

– $P_{GG} = 0.2$ and $P_{BB} = 0.8$

– $P_{GG} = 0.8$ and $P_{BB} = 0.2$

Table 5.1: Experimental results for $R = 1/4$

| $P_e(B)$ | $PER_{cut}$ | $PER_{b-cut}$ | $BER_{cut}$ | $BER_{b-cut}$ |
|---|---|---|---|---|
| | $P_{GG} = 0.8$, | | $P_{BB} = 0.8$ | |
| 0.1 | 0.14069 | 0.06818 | 0.10453 | 0.05030 |
| 0.13 | 0.34014 | 0.18173 | 0.26055 | 0.13535 |
| 0.16 | 0.58078 | 0.32466 | 0.45424 | 0.24698 |
| 0.2 | 0.81271 | 0.51087 | 0.67081 | 0.40820 |
| | $P_{GG} = 0.5$ | | $P_{BB} = 0.5$ | |
| 0.1 | 0.13464 | 0.04665 | 0.09799 | 0.03376 |
| 0.13 | 0.32711 | 0.12283 | 0.24417 | 0.08970 |
| 0.16 | 0.56365 | 0.23394 | 0.43088 | 0.17073 |
| 0.2 | 0.82358 | 0.43581 | 0.65848 | 0.33149 |
| | $P_{GG} = 0.2$ | | $P_{BB} = 0.8$ | |
| 0.1 | 0.20470 | 0.14177 | 0.14853 | 0.10266 |
| 0.13 | 0.46781 | 0.34619 | 0.35206 | 0.25621 |
| 0.16 | 0.73048 | 0.59497 | 0.57155 | 0.45788 |
| 0.2 | 0.942468 | 0.84951 | 0.79485 | 0.68579 |
| | $P_{GG} = 0.8$ | | $P_{BB} = 0.2$ | |
| 0.1 | 0.05709 | 0.00921 | 0.04201 | 0.00560 |
| 0.13 | 0.14292 | 0.01555 | 0.10386 | 0.01066 |
| 0.16 | 0.27728 | 0.03333 | 0.20678 | 0.02259 |
| 0.2 | 0.48898 | 0.07250 | 0.37152 | 0.05260 |

Analyzing the results in Table 5.1, we can conclude that for all values of $P_e(B)$ and for all values of $P_{GG}$ and $P_{BB}$ the results for $BER_{b-cut}$ are better than the corresponding results of $BER_{cut}$. The same conclusions can be derived for comparison of $PER_{b-cut}$ and $PER_{cut}$.

– for $P_{GG} = 0.8$ and $P_{BB} = 0.8$, $BER_{b-cut}$ is from 1.6 to 2.5 times better than $BER_{cut}$;

– for $P_{GG} = 0.5$ and $P_{BB} = 0.5$, $BER_{b-cut}$ is from 1.9 to 2.8 times better than $BER_{cut}$;

– for $P_{GG} = 0.2$ and $P_{BB} = 0.8$, $BER_{b-cut}$ is about 1.2 times better than $BER_{cut}$;

– for $P_{GG} = 0.8$ and $P_{BB} = 0.2$, $BER_{b-cut}$ is from 7.5 to 10 times better than $BER_{cut}$.

In Table 5.2, we give experimental results for code with rate $1/8$, with same combinations of transition probabilities from good to good state $P_{GG}$ and from bad to bad state $P_{BB}$ as for the code with rate $1/4$. There, $BER_{cut}$, $BER_{b-cut}$, $BER_{4sets}$ and $BER_{b-4sets}$ are bit-error probabilities obtained with Cut-Decoding, Burst-Cut-Decoding, 4-Sets-Cut-Decoding and Burst-4-Sets-Cut-Decoding algorithm, correspondingly. Also, $PER_{cut}$, $PER_{b-cut}$, $PER_{4sets}$ and $PER_{b-4sets}$ are corresponding packet-error probabilities.

Table 5.2: Experimental results for $R = 1/8$

| | $P_{GG} = 0.8, \quad P_{BB} = 0.8$ | | | |
|---|---|---|---|---|
| $P_e(B)$ | $BER_{cut}$ | $BER_{b-cut}$ | $BER_{4sets}$ | $BER_{b-4sets}$ |
| 0.1 | 0.09183 | 0.04224 | 0.01796 | 0.00601 |
| 0.13 | 0.23069 | 0.11561 | 0.08060 | 0.02616 |
| 0.16 | 0.41820 | 0.22276 | 0.20798 | 0.07405 |
| 0.2 | 0.64522 | 0.39519 | 0.45947 | 0.20217 |
| $P_e(B)$ | $PER_{cut}$ | $PER_{b-cut}$ | $PER_{4sets}$ | $PER_{b-4sets}$ |
| 0.1 | 0.15790 | 0.07445 | 0.03578 | 0.01252 |
| 0.13 | 0.37334 | 0.19549 | 0.16957 | 0.05429 |
| 0.16 | 0.62600 | 0.35476 | 0.41101 | 0.15207 |
| 0.2 | 0.85671 | 0.56624 | 0.75302 | 0.34288 |
| | $P_{GG} = 0.5 \quad P_{BB} = 0.5$ | | | |
| $P_e(B)$ | $BER_{cut}$ | $BER_{b-cut}$ | $BER_{4sets}$ | $BER_{b-4sets}$ |
| 0.1 | 0.08336 | 0.02759 | 0.01619 | 0.00412 |
| 0.13 | 0.21774 | 0.07662 | 0.07504 | 0.01353 |
| 0.16 | 0.20663 | 0.15975 | 0.20663 | 0.04268 |
| 0.2 | 0.63793 | 0.31081 | 0.47000 | 0.01298 |
| $P_e(B)$ | $PER_{cut}$ | $PER_{b-cut}$ | $PER_{4sets}$ | $PER_{b-4sets}$ |
| 0.1 | 0.22753 | 0.16028 | 0.07099 | 0.03729 |
| 0.13 | 0.52584 | 0.39026 | 0.29586 | 0.17713 |
| 0.16 | 0.80105 | 0.66229 | 0.66993 | 0.45636 |
| 0.2 | 0.97227 | 0.89840 | 0.95542 | 0.82632 |
| | $P_{GG} = 0.8 \quad P_{BB} = 0.2$ | | | |
| $P_e(B)$ | $BER_{cut}$ | $BER_{b-cut}$ | $BER_{4sets}$ | $BER_{b-4sets}$ |
| 0.1 | 0.03391 | 0.00320 | 0.00576 | 0.00044 |
| 0.13 | 0.0926 | 0.00866 | 0.02041 | 0.00124 |
| 0.16 | 0.18494 | 0.02014 | 0.06019 | 0.00256 |
| 0.2 | 0.34766 | 0.04408 | 0.16918 | 0.00803 |
| $P_e(B)$ | $PER_{cut}$ | $PER_{b-cut}$ | $PER_{4sets}$ | $PER_{b-4sets}$ |
| 0.1 | 0.05889 | 0.00583 | 0.01094 | 0.00079 |
| 0.13 | 0.16503 | 0.01684 | 0.04133 | 0.00230 |
| 0.16 | 0.31617 | 0.03578 | 0.12802 | 0.00518 |
| 0.2 | 0.55465 | 0.07913 | 0.34569 | 0.01569 |

From the results in Table 5.2, we can conclude that for all values of $P_e(B)$ and for all values of $P_{GG}$ and $P_{BB}$, the results for $BER_{b-cut}$ are better than the corresponding results of $BER_{cut}$ and the results for $BER_{b-4sets}$ are better than the

corresponding results of $BER_{4sets}$. Also, if we compare the results for burst algorithms, we can conclude that Burst-4-Sets-Cut-Decoding algorithm gives from 2 to 7 times better results than Burst-Cut-Decoding algorithm depending of the channel parameters. The same conclusions can be derived for packet-error probabilities.

### 5.3.2 Experiments for Gilbert-Elliott with Gaussian channels

In this subsection, we presented the experimental results for Gilbert-Elliott model with Gaussian channels with $SNR_G = 4$ and for different values of $SNR_B \in \{-3, -2, -1\}$ and the same transition probabilities as in the experiments with binary symmetric channels. First, in Table 5.3 we give experimental results for code with rate $1/4$, where we use the same notations as previously.

Table 5.3: Experimental results for $R = 1/4$

| $P_e(B)$ | $PER_{cut}$ | $PER_{b-cut}$ | $BER_{cut}$ | $BER_{b-cut}$ |
|---|---|---|---|---|
| | $P_{GG} = 0.8$, | $P_{BB} = 0.8$ | | |
| $-3$ | 0.56322 | 0.32366 | 0.44058 | 0.24605 |
| $-2$ | 0.46867 | 0.35419 | 0.35223 | 0.26129 |
| $-1$ | 0.16467 | 0.08179 | 0.12212 | 0.05911 |
| | $P_{GG} = 0.5$ | $P_{BB} = 0.5$ | | |
| $-3$ | 0.56322 | 0.32366 | 0.44058 | 0.24605 |
| $-2$ | 0.32754 | 0.12348 | 0.24303 | 0.08867 |
| $-1$ | 0.15243 | 0.05609 | 0.10993 | 0.03993 |
| | $P_{GG} = 0.2$ | $P_{BB} = 0.8$ | | |
| $-3$ | 0.73127 | 0.57783 | 0.57146 | 0.4417 |
| $-2$ | 0.46867 | 0.35419 | 0.35223 | 0.26129 |
| $-1$ | 0.22775 | 0.16561 | 0.16732 | 0.12115 |
| | $P_{GG} = 0.8$ | $P_{BB} = 0.2$ | | |
| $-3$ | 0.27282 | 0.03513 | 0.20121 | 0.02449 |
| $-2$ | 0.15056 | 0.01980 | 0.10891 | 0.01393 |
| $-1$ | 0.06732 | 0.00986 | 0.04852 | 0.00589 |

From Table 5.3, we can see that the results obtained with the new Burst-Cut-Decoding algorithm are from 2 to 8 times better than the corresponding results obtained with the old Cut-decoding algorithm.

In Table 5.4, we give experimental results for bit-error probabilities and packet-error probabilities for codes with rate $1/8$. The notations are previously given. From this table, we can make similar conclusions for rate $1/8$ as for rate $1/4$. Namely, we can conclude that for all values of $SNR$, results for $BER$ and $PER$ obtained with the new algorithms are better than the corresponding results obtained with the old versions of the algorithms. Also, Burst-4-Sets-Cut-decoding

algorithm gives from 2 to 8 times better results than Burst-Cut-decoding algorithm. The results of this part are published in [79].

Table 5.4: Експериментални резултати за $R = 1/8$

| $P_e(B)$ | $BER_{cut}$ | $BER_{b-cut}$ | $BER_{4sets}$ | $BER_{b-4sets}$ |
|---|---|---|---|---|
| | $P_{GG} = 0.8,$ | | $P_{BB} = 0.8$ | |
| $-3$ | 0.40468 | 0.21957 | 0.202399 | 0.07334 |
| $-2$ | 0.23806 | 0.11587 | 0.08246 | 0.02539 |
| $-1$ | 0.10796 | 0.04945 | 0.02266 | 0.00871 |
| | $P_{GG} = 0.5$ | | $P_{BB} = 0.5$ | |
| $-3$ | 0.38576 | 0.15412 | 0.20372 | 0.043289 |
| $-2$ | 0.21698 | 0.07825 | 0.07806 | 0.01490 |
| $-1$ | 0.09497 | 0.03024 | 0.02002 | 0.00385 |
| | $P_{GG} = 0.2$ | | $P_{BB} = 0.8$ | |
| $-3$ | 0.53986 | 0.42327 | 0.35535 | 0.22462 |
| $-2$ | 0.33082 | 0.24201 | 0.15007 | 0.08345 |
| $-1$ | 0.15119 | 0.10676 | 0.03686 | 0.02265 |
| | $P_{GG} = 0.8$ | | $P_{BB} = 0.2$ | |
| $-3$ | 0.17904 | 0.02085 | 0.05540 | 0.00328 |
| $-2$ | 0.09442 | 0.01037 | 0.02074 | 0.00134 |
| $-1$ | 0.04009 | 0.00503 | 0.00703 | 0.00060 |
| $P_e(B)$ | $PER_{cut}$ | $PER_{b-cut}$ | $PER_{4sets}$ | $PER_{b-4sets}$ |
| | $P_{GG} = 0.8,$ | | $P_{BB} = 0.8$ | |
| $-3$ | 0.61031 | 0.34835 | 0.40048 | 0.15092 |
| $-2$ | 0.38256 | 0.19693 | 0.17461 | 0.05457 |
| $-1$ | 0.17821 | 0.08640 | 0.048603 | 0.01771 |
| | $P_{GG} = 0.5$ | | $P_{BB} = 0.5$ | |
| $-3$ | 0.60865 | 0.26116 | 0.40710 | 0.09454 |
| $-2$ | 0.37028 | 0.13637 | 0.16748 | 0.03204 |
| $-1$ | 0.16402 | 0.05501 | 0.04169 | 0.00835 |
| | $P_{GG} = 0.2$ | | $P_{BB} = 0.8$ | |
| $-3$ | 0.78650 | 0.65300 | 0.65797 | 0.44758 |
| $-2$ | 0.53269 | 0.40408 | 0.31820 | 0.18130 |
| $-1$ | 0.26101 | 0.18613 | 0.08208 | 0.04910 |
| | $P_{GG} = 0.8$ | | $P_{BB} = 0.2$ | |
| $-3$ | 0.30645 | 0.03809 | 0.11880 | 0.00604 |
| $-2$ | 0.16345 | 0.01886 | 0.04428 | 0.00252 |
| $-1$ | 0.07063 | 0.00907 | 0.01404 | 0.00115 |

## 5.4 Experimental results for images for transmission through burst channels

In this section we present experimental results obtained with $RCBQ$ for transmission of images through a burst channel. Namely, we investigate performances of Burst-Cut-Decoding and Burst-4-Sets-Cut-Decoding algorithms for transmission of images through a Gilbert-Elliot channel. We made experiments with two kinds of Gilbert-Elliott channels (binary symmetric channels and Gaussian channels).

All experiments, presented in this subsection, are made for code (72, 576) with rate R=1/8, $B_{max} = 5$ and the same parametars as for ordinary messages fiven in previous section.

First, in Fig. 5.3 we present some images for $P_{GG} = 0.2$, $P_{BB} = 0.8$ obtained after transmission through the channel without using any error-correcting code. The first one is obtained after transmission through a Gilbert-Elliott with BSCs with a bit-error probability in the good state $P_e(G) = 0.01$ and a bit-error probability in the bad state $P_e(G) = 0.01$ and the second one − with Gaussian channels with $SNR_G = 4$ when the channel is in the good state and $SNR_B = -3$ when the channel is in the bad state.

Figure 5.3: Images obtained after transmission through the channel without using any error-correcting code

### 5.4.1 Experiments for Gilbert-Elliott with BSC channels

In all experiments for Gilbert-Elliott model with binary symmetric channels, we use bit-error probability in the good state $P_e(G) = 0.01$ and a few different values of bit-error probabilities in the bad state $P_e(B) \in \{0.16; 0.13; 0.1\}$ and the following combinations of transition probabilities from good to good state $P_{GG}$ and from bad to bad state $P_{BB}$:

- $P_{GG} = 0.2$ and $P_{BB} = 0.8$
- $P_{GG} = 0.8$ and $P_{BB} = 0.2$

In Fig. 5.4 we present the images obtained with Burst-Cut-Decoding algorithm for $P_{GG} = 0.2$, $P_{BB} = 0.8$ and all considered $P_e(B)$ obtained with Burst-Cut-Decoding algorithm. The corresponding images obtained after aplication of the filter (proposed in Section 3.3.1.) are presented in Fig. 5.5. Images obtained for

$P_{GG} = 0.8$ and $P_{BB} = 0.2$ are given in Fig. 5.6 (without the filter) and Fig. 5.7 (with the filter).

Figure 5.4: Images obtained with Burst-Cut-Decoding algorithm without the filter for $P_{GG} = 0.2$, $P_{BB} = 0.8$

Figure 5.5: Images obtained with Burst-Cut-Decoding algorithm with the filter for $P_{GG} = 0.2$, $P_{BB} = 0.8$

Images obtained with Burst-4-Sets-Cut-Decoding algorithm for $P_{GG} = 0.2$, $P_{BB} = 0.8$ are presented in Fig. 5.8 (without the filter) and Fig. 5.9 (with the filter), while images for $P_{GG} = 0.8$, $P_{BB} = 0.2$ are presented in Fig. 5.10 (without the filter) and Fig. 5.11 (with the filter).

From the images in Fig. 5.4 and Fig. 5.8 (for $P_{GG} = 0.2$, $P_{BB} = 0.8$) we can conclude that Burst-4-Sets-Cut-Decoding algorithm gives better results than Burst-Cut-Decoding algorithm for all considered values of $P_e(B)$. Comparing the images before and after applying the filter we can conclude that the filter significantly enhances the quality of images decoded with both algorithms. But, it is visible that the filter gives better results for the images obtained with Burst-Cut-Decoding algorithm than with Burst-4-Sets-Cut-Decoding algorithm. The reason for this is

Figure 5.6: Images obtained with Burst-Cut-Decoding algorithm without the filter for $P_{GG} = 0.8$, $P_{BB} = 0.2$

Figure 5.7: Images obtained with Burst-Cut-Decoding algorithm with the filter for $P_{GG} = 0.8$, $P_{BB} = 0.2$

Figure 5.8: Images obtained with Burst-4-Sets-Cut-Dcoding algorithm without the filter for $P_{GG} = 0.2$, $P_{BB} = 0.8$

the larger number of undetected-errors in the experiments with Burst-4-Sets-Cut-Decoding algorithm. Namely, the filter cannot detect this kind of errors. The images for $P_{GG} = 0.8$, $P_{BB} = 0.2$ (in Fig. 5.6, Fig. 5.7, Fig. 5.10, Fig. 5.11) are

Figure 5.9: Images obtained with Burst-4-Sets-Cut-Dcoding algorithm with the filter for $P_{GG} = 0.2$, $P_{BB} = 0.8$

Figure 5.10: Images obtained with Burst-4-Sets-Cut-Dcoding algorithm without the filter for $P_{GG} = 0.8$, $P_{BB} = 0.2$

Figure 5.11: Images obtained with Burst-4-Sets-Cut-Dcoding algorithm with the filter for $P_{GG} = 0.8$, $P_{BB} = 0.2$

clearer due to the smaller number of errors in the channels with these transition probabilities. Therefore, in these images there is no great difference between the images decoded with both algorithms and after applying the filter.

### 5.4.2 Experiments for Gilbert-Elliott with Gaussian channels

In this subsection, we present the experimental results for Gilbert-Elliott model with Gaussian channels for $SNR_G = 4$ and different values of $SNR_B \in \{-3, -2, -1\}$. For these channels we made experiments with the same transition probabilities as in the experiments with binary symmetric channels.

Images obtained with Burst-Cut-Decoding algorithm for $P_{GG} = 0.2, P_{BB} = 0.8$ and all considered $SNR_B$ are given in Fig. 5.12. The images in Fig. 5.13 are obtained from the images given in Fig. 5.12 after applying the filter. Images for $P_{GG} = 0.8$, $P_{BB} = 0.2$ are given in Fig. 5.14 (without the filter) and Fig. 5.15 (with the filter).

Figure 5.12: Images obtained with Burst-Cut-Decoding algorithm without the filter for $P_{GG} = 0.2$, $P_{BB} = 0.8$

Figure 5.13: Images obtained with Burst-Cut-Decoding algorithm with the filter for $P_{GG} = 0.2$, $P_{BB} = 0.8$

In Fig. 5.16 (without the filter) and in Fig. 5.17 (with the filter) we present the images obtained with Burst-4-Sets-Cut-Decoding algorithm for $P_{GG} = 0.2$,

Figure 5.14: Images obtained with Burst-Cut-Decoding algorithm without the filter for $P_{GG} = 0.8$, $P_{BB} = 0.2$

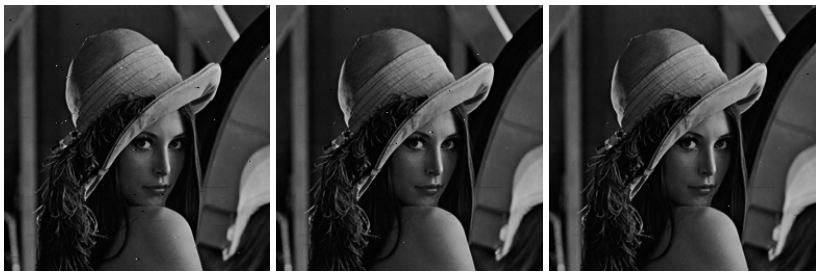


Figure 5.15: Images obtained with Burst-Cut-Dcoding algorithm with the filter for $P_{GG} = 0.8$, $P_{BB} = 0.2$

$P_{BB} = 0.8$. Images obtained with this algorithm for $P_{GG} = 0.8$, $P_{BB} = 0.2$ are given in Fig. 5.18 (without the filter) and in Fig. 5.19 (with the filter).

Figure 5.16: Images obtained with Burst-4-Sets-Cut-Decoding algorithm without the filter for $P_{GG} = 0.2$, $P_{BB} = 0.8$

Figure 5.17: Images obtained with Burst-4-Sets-Cut-Decoding algorithm with the filter for $P_{GG} = 0.2$, $P_{BB} = 0.8$

Figure 5.18: Images obtained with Burst-4-Sets-Cut-Decoding algorithm without the filter for $P_{GG} = 0.8$, $P_{BB} = 0.2$

Figure 5.19: Images obtained with Burst-4-Sets-Cut-Decoding algorithm with the filter for $P_{GG} = 0.8$, $P_{BB} = 0.2$

Analyzing the images given in this subsection (transmitted through a Gilbert-Elliott with Gaussian channels) we can derive the same conclusions as for images transmitted through a Gilbert-Elliott with BSCs. The results of this section are

published in [80].

## 5.5 Fast Decoding with Cryptcodes for Burst Errors

In this section, we want to adopt fast algorithms of RCBQ given in Chapter 4. for transmission through a burst channel. So, we combine the previous two ideas in two new algorithms for coding/decoding called FastB-Cut-Decoding and FastB-4-Sets-Cut-Decoding algorithms. Namely, at first we apply one of the coding algorithm (Fast-Cut-Decoding or Fast-4-Sets-Cut-Decoding algorithms) on the original message. After that we apply interleaver on the obtained codewords (before concatenation). Interleaved message is transmitted through a noisy burst channel. On the received messages on the output of the channel, after dividing in two (or four) parts we apply deinterleaver of each part and then all parts are decoded using the appropriate decoding algorithm.

### 5.5.1 Experimental Results

Here, we present experimental results obtained with FastB-Cut-Decoding and Fast-B-4-Sets-Cut-Decoding algorithms for rate $R = 1/4$ and $R = 1/8$, for transmission through a burst channel. In our experiments we use Gilbert-Elliott model where in both states the channel is a Gaussian and $SNR_G$ in a good state is high and $SNR_B$ in a bad state is low.

We compare results obtained with FastB-Cut-Decoding and FastB-4-Sets-Cut-Decoding algorithms with the corresponding results for Burst-Cut-Decoding and Burst-4-Sets-Cut-Decoding algorithms. Also, in order to show the efficiency of Fast algorithms we present percentages of messages which decoding finished with $B_{max} = 1, 2, 3, 4$ or $5$.

In the experiments we use the following code parameters.

- For code $(72, 288)$ in Burst-Cut-Decoding and FastB-Cut-Decoding algorithm, the parameters are:
  - redundancy pattern: 1100 1110 1100 1100 1110 1100 1100 1100 0000 for rate $1/2$ and two different keys of 10 nibbles.

- For code $(72, 576)$, the code parameters are:

– in Burst-Cut-Decoding and FastB-Cut-Decoding - redundancy pattern: 1100 1100 1000 00001100 1000 1000 0000 1100 1100 1000 0000 1100 1000 1000 0000 0000 0000, for rate $1/4$ and two different keys of 10 nibbles,

– in Burst-4-Sets-Cut-Decoding and FastB-4-Sets-Cut-Decoding - redundancy pattern: 1100 1110 1100 1100 1110 1100 1100 1100 0000 for rate $1/2$ and four different keys of 10 nibbles.

In all experiments we use the same quasigroup on $Q$ given in Table 3.2.

In the experiments with Burst-Cut-Decoding for the code $(72, 288)$ we use $B_{max} = 4$, and in the experiments with FastB-Cut-Decoding algorithm the maximum value of $B_{max}$ is 4. For the code $(72, 576)$ in Burst-Cut-Decoding and Burst-4-Sets-Cut-Decoding algorithms we use $B_{max} = 5$, and in FastB-Cut-Decoding and FastB-4-Sets-Cut-Decoding algorithms the maximum value of $B_{max}$ is 5.

In all experiments the value of $SNR$ in the good state is $SNR_G = 4$. We made experiments for different values of $SNR$ in the bad state $SNR_B \in \{-3, -2, -1\}$ and for the following combinations of transition probabilities from good to good state $P_{GG}$ and from bad to bad state $P_{BB}$ in the Gilbert-Elliott model:

– $P_{GG} = 0.8$ and $P_{BB} = 0.8$

– $P_{GG} = 0.5$ and $P_{BB} = 0.5$

– $P_{GG} = 0.2$ and $P_{BB} = 0.8$

– $P_{GG} = 0.8$ and $P_{BB} = 0.2$

In Table 5.5, we give experimental results for bit-error probabilities $BER$ and packet-error probabilities $PER$ for the code $(72, 288)$. With $BER_{b-cut}$ and $PER_{b-cut}$ we denote probabilities obtained with Burst-Cut-Decoding algorithm, and with $BER_{fb-cut}$ and $PER_{fb-cut}$ probabilities obtained with FastB-Cut-Decoding algorithm. The results for $BER_{b-cut}$ and $PER_{b-cut}$ for Burst-Cut-Decoding algorithm are presented in Section 5.3.2.

Table 5.5: Experimental results for code $(72, 288)$

| $SNR_B$ | $PER_{fb-cut}$ | $PER_{b-cut}$ | $BER_{fb-cut}$ | $BER_{b-cut}$ |
|---|---|---|---|---|
| | | $P_{GG} = 0.8$ | $P_{BB} = 0.8$ | |
| $-3$ | 0.27657 | 0.32366 | 0.20524 | 0.24605 |
| $-2$ | 0.15431 | 0.17727 | 0.11143 | 0.13127 |
| $-1$ | 0.06588 | 0.08179 | 0.04531 | 0.05911 |
| | | $P_{GG} = 0.5$ | $P_{BB} = 0.5$ | |
| $-3$ | 0.20838 | 0.23135 | 0.14832 | 0.16945 |
| $-2$ | 0.10023 | 0.12348 | 0.06993 | 0.08867 |
| $-1$ | 0.04097 | 0.05609 | 0.02839 | 0.03993 |
| | | $P_{GG} = 0.2$ | $P_{BB} = 0.8$ | |
| $-3$ | 0.57402 | 0.57783 | 0.43486 | 0.44170 |
| $-2$ | 0.35088 | 0.35419 | 0.25447 | 0.26129 |
| $-1$ | 0.15315 | 0.16561 | 0.10795 | 0.12115 |
| | | $P_{GG} = 0.8$ | $P_{BB} = 0.2$ | |
| $-3$ | 0.02254 | 0.03513 | 0.01529 | 0.02449 |
| $-2$ | 0.00994 | 0.01980 | 0.00669 | 0.01393 |
| $-1$ | 0.00382 | 0.00986 | 0.00253 | 0.00589 |

Analyzing the results in Table 5.5, we can conclude that for all values of $SNR_B$ and all combinations of transition probabilities, results for $BER_{fb-cut}$ and $PER_{fb-cut}$ are slightly better than the corresponding results of $BER_{b-cut}$ and $PER_{b-cut}$.

When decodings in FastB-Cut-Decoding algorithm end with $B_{max} = 1$ or $B_{max} = 2$, decoding with this algorithm is much faster than with Burst-Cut-Decoding. Therefore, in Table 5.6 we give the percentage of messages for which decoding ended with $B_{max} = 1$, $B_{max} = 2$, $B_{max} = 3$ or $B_{max} = 4$ in the experiments with FastB-Cut-Decoding algorithm. From the results given there, we can see that the percentages depend not only on the value of $SNR_B$ but also on the transition probabilities for bad-to-bad and good-to-good state. For $P_{GG} = 0.8, P_{BB} = 0.2$ and all values of $SNR_B$ for more than $75\%$ of the messages decoding successfully finished with $B_{max} = 1$ or $B_{max} = 2$. For $P_{GG} = 0.8, P_{BB} = 0.8$ or $P_{GG} = 0.5, P_{BB} = 0.5$ and $SNR_B = -1$ more than $50\%$ of the decoding successfully finished with $B_{max} = 1$ or $B_{max} = 2$. In other experiments, where the probability the channel to be in a bad state is greater we have a smaller percentages

for successful decoding with $B_{max} = 1$ or $B_{max} = 2$. So, we can conclude that the new algorithm improves decoding speed of RCBQs for transmission through a Gilbert-Elliott channel with smaller probability for bad state.

Table 5.6: Percentage of messages decoded with different values of $B_{max}$

| $SNR_B$ | $B_{max} = 1$ | $B_{max} = 2$ | $B_{max} = 3$ | $B_{max} = 4$ |
|---|---|---|---|---|
| | | $P_{GG} = 0.8$ | $P_{BB} = 0.8$ | |
| $-3$ | 15.79% | 18.23% | 17.71% | 48.26% |
| $-2$ | 20.06% | 22.56% | 22.34% | 35.04% |
| $-1$ | 24.95% | 29.59% | 24.25% | 21.21% |
| | | $P_{GG} = 0.5$ | $P_{BB} = 0.5$ | |
| $-3$ | 5.11% | 20.39% | 26.30% | 48.19% |
| $-2$ | 8.92% | 29.10% | 30.34% | 31.64% |
| $-1$ | 18.17% | 37.82% | 27.70% | 16.31% |
| | | $P_{GG} = 0.2$ | $P_{BB} = 0.8$ | |
| $-3$ | 0.17% | 3.28% | 10.95% | 85.60% |
| $-2$ | 0.42% | 8.55% | 21.45% | 69.58% |
| $-1$ | 2.17% | 20.99% | 32.60% | 44.24% |
| | | $P_{GG} = 0.8$ | $P_{BB} = 0.2$ | |
| $-3$ | 42.42% | 32.80% | 16.32% | 8.46% |
| $-2$ | 50.97% | 32.00% | 12.66% | 4.37% |
| $-1$ | 60.10% | 29.56% | 8.13% | 2.22% |

In Table 5.7 and Table 5.8, we present the experimental results for code $(72, 576)$ with rate $R = 1/8$. We compare bit-error ($BER$) and packet-error ($PER$) probabilities obtain with Burst-Cut-Decoding, FastB-Cut-Decoding, Burst-4-Sets-Cut-Decoding and FastB-4-Sets-Cut-Decoding algorithms. With $BER_{b-cut}$, $PER_{b-cut}$, $BER_{b-4sets}$, $PER_{b-4sets}$ we denote probabilities for packet- and bit-error obtained with Burst algorithms (Burst-Cut-Decoding and Burst-4-Sets-Cut-Decoding algorithms) and with $BER_{fb-cut}$, $PER_{fb-cut}$, $BER_{fb-4sets}$ and $PER_{fb-4sets}$ the corresponding probabilities obtained with FastB algorithms (FastB-Cut-Decoding and FastB-4-Sets-Cut-Decoding).

Table 5.7: Experimental results for $BER$ for code $(72, 576)$

| $SNR_B$ | $BER_{fb-cut}$ | $BER_{b-cut}$ | $BER_{fb-4sets}$ | $BER_{b-4sets}$ |
|---|---|---|---|---|
| | $P_{GG} = 0.8,$ | | $P_{BB} = 0.8$ | |
| $-3$ | 0.06381 | 0.09182 | 0.01421 | 0.03798 |
| $-2$ | 0.02148 | 0.03935 | 0.00322 | 0.01631 |
| $-1$ | 0.00624 | 0.01347 | 0.00044 | 0.00518 |
| | $P_{GG} = 0.5$ | | $P_{BB} = 0.5$ | |
| $-3$ | 0.03417 | 0.05539 | 0.00588 | 0.02263 |
| $-2$ | 0.01257 | 0.02198 | 0.00138 | 0.00853 |
| $-1$ | 0.00250 | 0.00707 | 0.00022 | 0.00251 |
| | $P_{GG} = 0.2$ | | $P_{BB} = 0.8$ | |
| $-3$ | 0.16994 | 0.19376 | 0.06586 | 0.08592 |
| $-2$ | 0.06408 | 0.08687 | 0.01560 | 0.03244 |
| $-1$ | 0.01697 | 0.02736 | 0.00258 | 0.01085 |
| | $P_{GG} = 0.8$ | | $P_{BB} = 0.2$ | |
| $-3$ | 0.00200 | 0.00477 | 0 | 0.00225 |
| $-2$ | 0.00063 | 0.00174 | 0 | 0.00091 |
| $-1$ | 0.00013 | 0.00075 | 0 | 0.00026 |

From the results given in Table 5.7 and Table 5.8, we can conclude that for all combinations of transition probabilities and all values of $SNR_B$, results for $BER$ and $PER$ obtained with new FastB algorithms are better than the corresponding results obtained with Burst algorithms. Comparing the results of all algorithms, we can see that the best results (for all channel parameters) are obtained with FastB-4-Sets-Cut-Decoding algorithm. For $P_{GG} = 0.8, P_{BB} = 0.2$ and all considered values of $SNR_B$, the values of $BER$ and $PER$ obtained with this algorithm are equal to 0. Also, from the duration of the experiments we concluded that FastB-4-Sets-Cut-Decoding algorithm is much faster than the other three considered algorithms.

In Table 5.9, we give the percentage of messages which decoding with FastB-Cut-Decoding and FastB-4-Sets-Cut-Decoding algorithms ended with $B_{max} = 1$, $B_{max} = 2$, $B_{max} = 3$, $B_{max} = 4$ or $B_{max} = 5$.

Table 5.8: Experimental results for $PER$ for code $(72, 576)$

| $SNR_B$ | $PER_{fb-cut}$ | $PER_{b-cut}$ | $PER_{fb-4sets}$ | $PER_{b-4sets}$ |
|---|---|---|---|---|
| | | $P_{GG} = 0.8,$ | $P_{BB} = 0.8$ | |
| $-3$ | 0.11470 | 0.16734 | 0.02657 | 0.06552 |
| $-2$ | 0.04155 | 0.07273 | 0.00662 | 0.02931 |
| $-1$ | 0.01202 | 0.02556 | 0.00079 | 0.00900 |
| | | $P_{GG} = 0.5$ | $P_{BB} = 0.5$ | |
| $-3$ | 0.06509 | 0.10765 | 0.01202 | 0.03910 |
| $-2$ | 0.02304 | 0.04198 | 0.00324 | 0.01419 |
| $-1$ | 0.00540 | 0.01375 | 0.00043 | 0.00446 |
| | | $P_{GG} = 0.2$ | $P_{BB} = 0.8$ | |
| $-3$ | 0.29551 | 0.35628 | 0.12025 | 0.14653 |
| $-2$ | 0.11874 | 0.16381 | 0.02988 | 0.05688 |
| $-1$ | 0.03283 | 0.05357 | 0.00490 | 0.01879 |
| | | $P_{GG} = 0.8$ | $P_{BB} = 0.2$ | |
| $-3$ | 0.00374 | 0.00886 | 0 | 0.00410 |
| $-2$ | 0.00115 | 0.00382 | 0 | 0.00158 |
| $-1$ | 0.00029 | 0.00158 | 0 | 0.00043 |

From Table 5.9, we can see that in all experiments we obtained better percentages (larger percentages for smaller $B_{max}$ and smaller percentages for larger $B_{max}$) with FastB-4-Sets-Cut-Decoding algorithm than with FastB-Cut-Decoding algorithm. In almost all cases, the percentage of messages which decoding needed $B_{max} = 5$ with these algorithm is below $8\%$ and for more than $60\%$ of messages the decoding ended with $B_{max} \leqslant 3$. Exception of this are only the cases when $P_{GG} = 0.2, P_{BB} = 0.8, SNR_B \leqslant -2$ and $P_{GG} = 0.8, P_{BB} = 0.8, SNR_B = -3$, Also, for $P_{GG} = 0.8, P_{BB} = 0.2$ more than half of decodings ended with $B_{max} = 1$. This means that for these channel parameters FastB-4-Sets-Cut-Decoding algorithm is faster than Burst-4-Sets-Cut-Decoding algorithm.

Table 5.9: Percentage of messages decoded with different values of $B_{max}$

| | FastB-Cut-Decoding | | | | |
|---|---|---|---|---|---|
| $SNR_B$ \ $B_{max}$ | 1 | 2 | 3 | 4 | 5 |
| | $P_{GG} = 0.8 \quad P_{BB} = 0.8$ | | | | |
| $-3$ | 3.49% | 14.38% | 24.80% | 28.57% | 28.76% |
| $-2$ | 5.16% | 22.52% | 32.39% | 24.98% | 14.96% |
| $-1$ | 8.10% | 34.92% | 34.77% | 16.15% | 6.06% |
| | $P_{GG} = 0.5 \quad P_{BB} = 0.5$ | | | | |
| $-3$ | 0.22% | 9.41% | 33.84% | 33.83% | 22.70% |
| $-2$ | 0.78% | 20.39% | 43.13% | 25.25% | 10.45% |
| $-1$ | 2.62% | 39.57% | 40.81% | 13.28% | 3.72% |
| | $P_{GG} = 0.2 \quad P_{BB} = 0.8$ | | | | |
| $-3$ | 0% | 0.18% | 6.57% | 29.51% | 63.75% |
| $-2$ | 0% | 1.56% | 20.51% | 41.17% | 36.76% |
| $-1$ | 0.04% | 7.75% | 40.79% | 35.68% | 15.74% |
| | $P_{GG} = 0.8 \quad P_{BB} = 0.2$ | | | | |
| $-3$ | 17.09% | 49.87% | 24.72% | 6.55% | 1.78% |
| $-2$ | 25.58% | 52.37% | 17.68% | 3.64% | 0.73% |
| $-1$ | 36.12% | 51.60% | 10.25% | 1.68% | 0.35% |
| | FastB-4-Sets-Cut-Decoding | | | | |
| $SNR_B$ \ $B_{max}$ | 1 | 2 | 3 | 4 | 5 |
| | $P_{GG} = 0.8 \quad P_{BB} = 0.8$ | | | | |
| $-3$ | 23.22% | 18.02% | 22.20% | 25.42% | 11.13% |
| $-2$ | 27.61% | 25.09% | 26.56% | 17.04% | 3.70% |
| $-1$ | 35.64% | 32.21% | 24.14% | 7.33% | 0.68% |
| | $P_{GG} = 0.5 \quad P_{BB} = 0.5$ | | | | |
| $-3$ | 7.13% | 20.07% | 35.07% | 30.54% | 7.19% |
| $-2$ | 12.36% | 31.98% | 38.33% | 15.50% | 1.84% |
| $-1$ | 23.95% | 42.21% | 28.65% | 4.85% | 0.34% |
| | $P_{GG} = 0.2 \quad P_{BB} = 0.8$ | | | | |
| $-3$ | 0.17% | 2.12% | 8.19% | 45.62% | 43.90% |
| $-2$ | 0.68% | 6.96% | 25.23% | 50.45% | 16.68% |
| $-1$ | 3.18% | 19.99% | 44.45% | 28.76% | 3.62% |
| | $P_{GG} = 0.8 \quad P_{BB} = 0.2$ | | | | |
| $-3$ | 54.50% | 31.98% | 11.89% | 1.55% | 0.09% |
| $-2$ | 63.89% | 29.17% | 6.44% | 0.48% | 0.01% |
| $-1$ | 73.37% | 23.83% | 2.68% | 0.12% | 0% |

With these new algorithms, FastB-Cut-Decoding and FastB-4-Sets-Cut-Decoding algorithms, we obtained better results for packet-error and bit-error probabilities than with the previously defined (Burst-Cut-Decoding and Burst-4-Sets-Cut-Decoding) algorithms of RCBQ for transmission through a burst channels. Also, from the presented percentages of messages which decoding ends with different values of $B_{max}$ we can conclude that for some channel parameters FastB-Cut-Decoding and FastB-4-Sets-Cut-Decoding algorithms provide faster decoding.

# Chapter 6

# Blockchain Technology

## 6.1 Blockchain Technology - notion and basics

*Blockchain* is a decentralized ledger of transactions, distributed on all computers in one peer-to-peer network ($P2P$) where all details of transactions are visible to everyone connected to the network. Namely, this is a growing list of linked blocks. The blocks consist of valid transactions, a timestamp and a hash pointer as a link to the previous block in the chain and ”nonce” (a number that is generated only once and which participates in the formation of the hash value of the block) shown in Fig. 6.1.

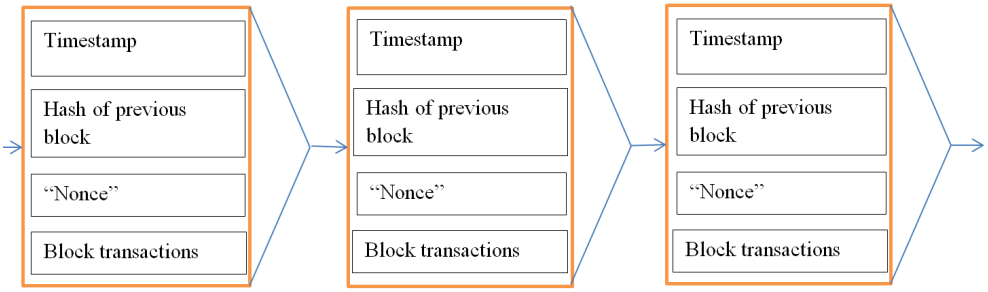



Figure 6.1: A simple Blockchain

When someone requests a transaction or two parties exchange data, this could be money, contract or any other asset that can be digitally described, the requested

transaction is broadcast to peer-to-peer network consisting of computers called nodes. This network of nodes validates the transaction and the status of the user using known algorithms. Depending on the network's parameters, the transaction is either verified immediately or transcribed into a secured record and placed in a list of pending transactions. Each block is identified by a hash and contains a header (a reference to the hash of the previous block) and a group of transactions given in Fig. 6.2.

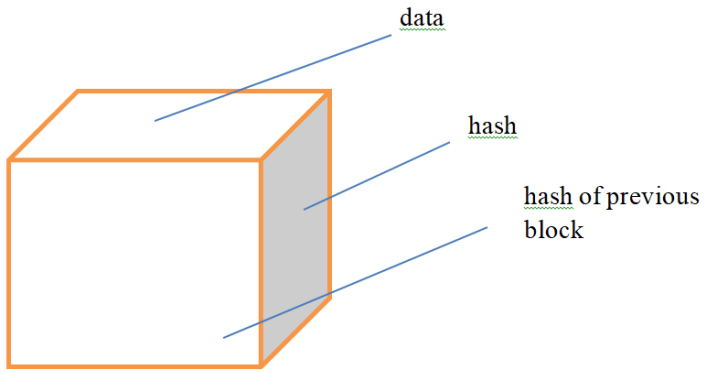



Figure 6.2: One block in the Blockchain

The sequence of linked blocks creates a secure, interdependent chain. Blocks must first be validated to be added to the Blockchain. When a block is verified, it is distributed through the network (added to the existing chain) and each node adds the block to the majority Blockchain. Then the transaction is completed. A schematic representation of Blockchain technology is given in Fig. 6.3.

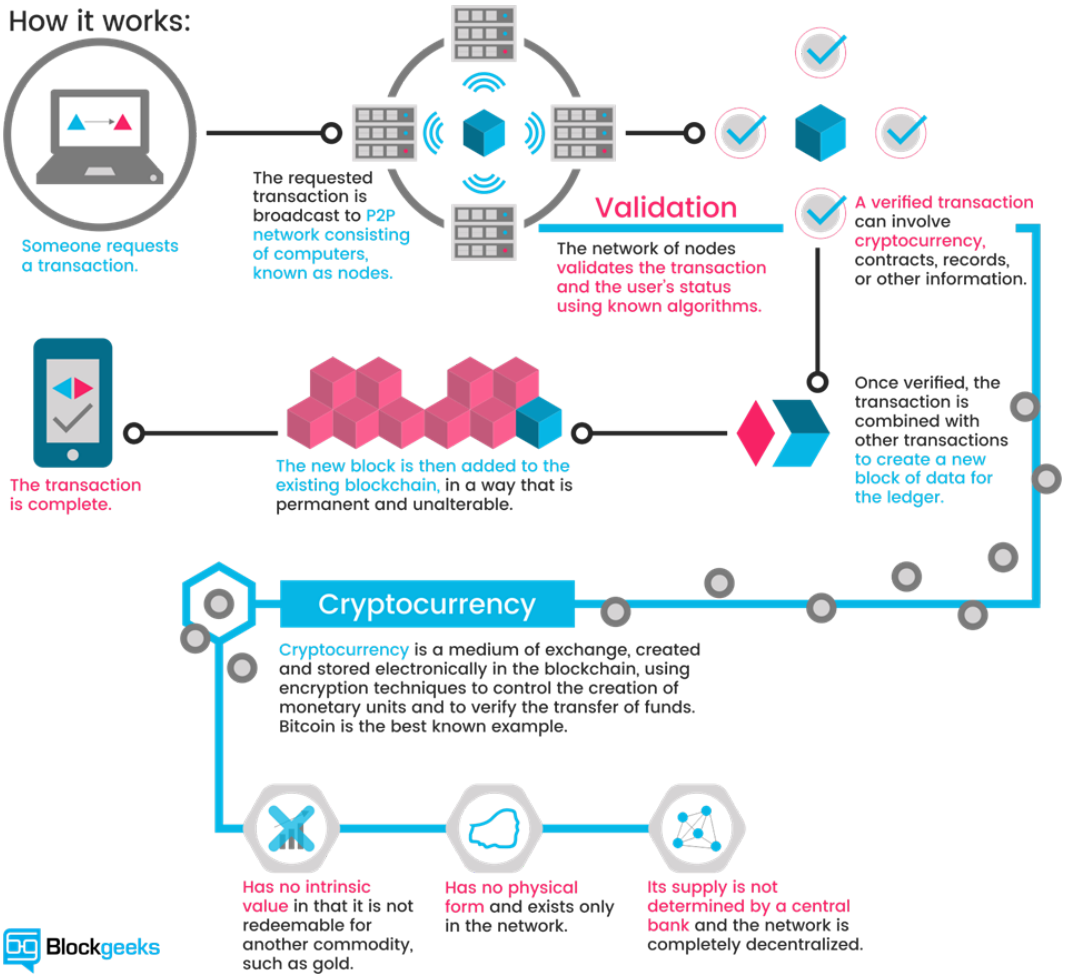



Figure 6.3: Blockchain technology [50]

There are four types of Blockchains:

– *Public Blockchain* - where there are no access restrictions. All transactions that take place in the public Blockchain are completely transparent, i.e., everyone has access to every transaction ever made. Anyone with an internet connection can send transactions and also be a validator (i.e., participate in the execution of the consensus protocol). The most popular public Blockchain innovation are *bitcoin* and *ethereum*, digital currencies [19].

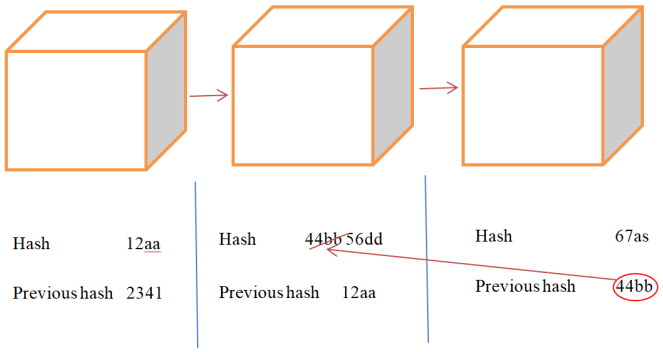



Figure 6.4: Modified block in Blockchain

– *Private Blockchain* - users in this Blockchain need permission to join the chain. Transactions are private and available only to users who have been granted access to the chain. The private Blockchain is more centralized and the chain management entities have significant control over its participants.

– *Permitted Blockchain* - Unlike a private Blockchain where an entity controls the entire Blockchain, the permitted Blockchain is managed by a group of entities, i.e., this type can be considered as a subcategory of the private Blockchain. The advantage of this Blockchain is that it can unite a group of businesses that work together, but also compete with each other, thus enabling them to be more efficient, both individual and collective, collaborating in some aspects of their businesses.

– *Hybrid Blockchain* - combining the privacy advantage of a private or permitted Blockchain with the security and transparency benefits of a public Blockchain. This contributes to significant flexibility in the selection of entities, which data to make public and transparent, and which private when forming a Blockchain.

## 6.2 Bitcoin

In 2008, an individual or group writing under the name of Satoshi Nakamoto, published a paper entitled “Bitcoin: A Peer-To-Peer Electronic Cash System” [83]. *Bitcoin*, which is the first major application of Blockchain, is a peer-to-peer ver-

sion of the electronic cash that allows direct online payments from one party to another without the need of trusted third party. This cryptocurrency uses cryptography to secure transactions. Everybody who installs the open source program that implements this new protocol can become part of the bitcoin peer-to-peer network [7].

The main characteristics of bitcoin are:

- *Decentralized* - means non-existence of a central bank. Emitters of this currency are the users or holders of ”mining” computers who verify the transactions.
- The security provided by cryptography with public key gives *the confidence* of this currency.
- *Authentication* of the transactions for one to another peer in Blockchain network is made with digital signature.
- Digital signature of the message also provides *integrity* through the transfer.

Every transaction consists of:

- *input* (that shows transactions with which the sender received Bitcoins)
- *quantity* (the amount of Bitcoins that sender sends to the recipient)
- *output* (Bitcoin address of the recipient)

### 6.2.1 How does bitcoin work?

If Alice wants to send money (5 bitcoins) to Bob, she broadcasts a message with her account and the amount of 5 bitcoins. Each node that receives the message updates the copy of the ledger of transactions and then sends the message for the transaction, shown in Fig. 6.5.

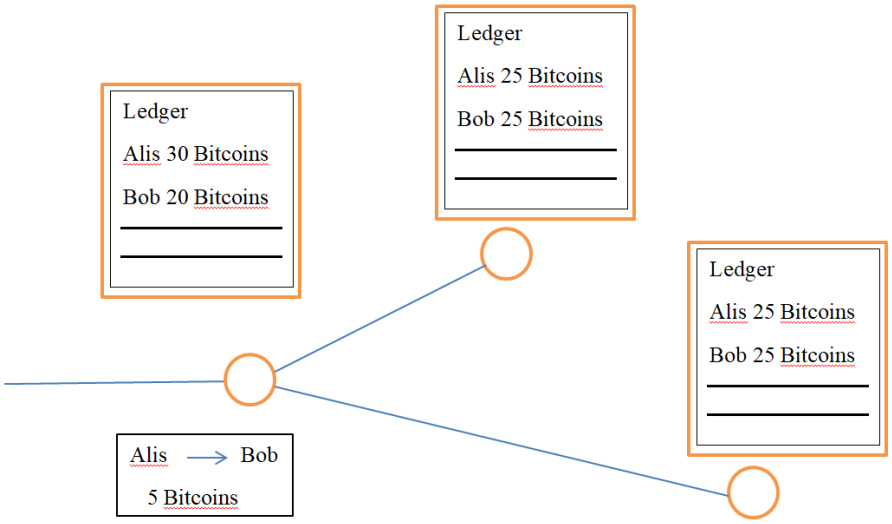



Figure 6.5: Sending message

The bitcoin network user requires a special password to be able to receive the sent money. This type of security is provided by the *digital signature*. Its help proves the authenticity of the message through a mathematical algorithm that protects against copying and fraud in the digital sphere.

The digital signature uses two connected keys:

– *Private key* used to create the signature.

– *Public key* used to check a message's affiliation.

The sender generates a private key and public key. After that, they sign the message with the signature and send their public key, the signature and the message to the network. Then the node checks that the message has been signed by the sender using the verification algorithm. That can be done by the holder of the private key to the public key that is sent. The process of generating and verifying the signature is schematically shown in Fig. 6.6.

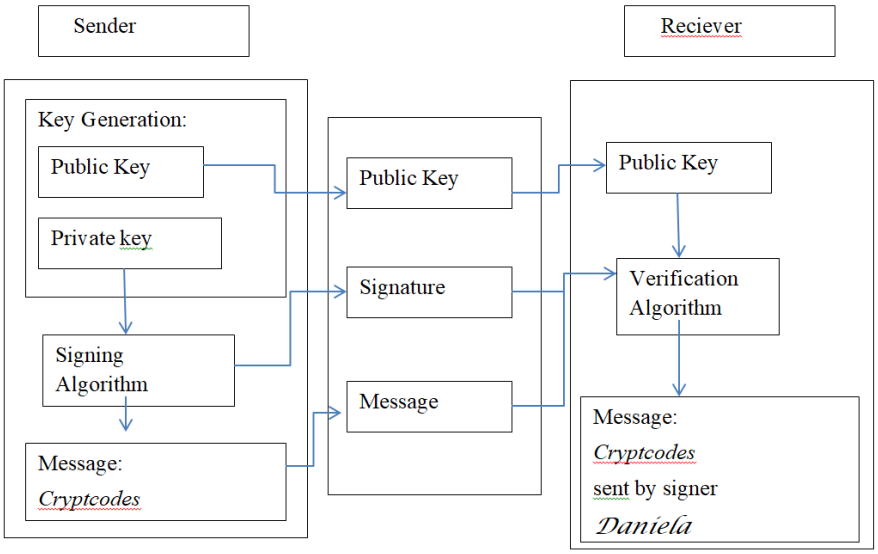



Figure 6.6: Digital signature

Since the signature depends on the message, it will be different for each transaction (Fig. 6.7). This dependency means that no one can change the message as it passes through the network, as any change in the message will destroy the signature. In this way, the integrity of the transmission is ensured.

| | | Digital signature |
|---|---|---|
| Alis → Bob | 5 bitcoins | 546373728... |
| Alis → Bil | 10 bitcoins | 535372728... |
| Alis → Kim | 100 bitcoins | 662772827... |
| | | different each time |

Figure 6.7: Transaction messages

### 6.2.2 Transaction and ledger of transactions

If Alice wants to send 5 bitcoins to Bob, she must refer to those transactions where Alice gets bitcoins. These are *input transactions*. Other nodes that verify this transaction check all of Alice's input transactions. In the bitcoin network, the rule is that all incoming transactions must be used. If the output is less than the input transactions then the difference between the input and the output as an input transaction is returned to the sender. The transaction chain is shown in Fig. 6.8.

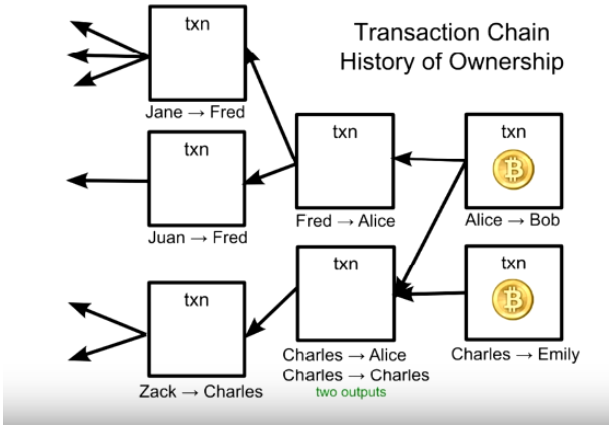



Figure 6.8: Transaction chain [34]

Bitcoin *miners* verify the transactions, put them into blocks of transactions and decide which block is the next one, i.e., *bitcoin system* groups the transactions into blocks and connects them in Blockchain. Since, many people can create blocks at the same time, which block will enter first in Blockchain is decided by using a hash function. The factor that determines the probability that the *mining* computer will verify the block of transactions is known as *difficulty* factor.

### 6.2.3 Hashing in Bitcoin system

A hash function is a mathematical process that takes as input a string of arbitrary length, performs an operation on it, and returns output data of a fixed length.
Bitcoin uses both SHA-256 and RIPEMD-160 hashes. Usually, when a hash is used in Bitcoin system, it is computed twice. Most often SHA-256 hashes are used, while RIPEMD-160 is used for creating a shorter hash for a bitcoin address.

On Table 6.1 we give an example of double hashing of string ”bitcoin”.

Table 6.1: Example of double hashing of string “bitcoin”

| $SHA256$ | |
|---|---|
| *first round of SHA-256* | 6b88c087247aa2f07ee1c5956b8e1a9f<br>4c7f892a70e324f1bb3d161e05ca107b |
| *second round of SHA-256* | a23b7f87e4250b3a64b737f349c06422<br>f752f419cbb25ae9169a6cf1e23f4462 |
| *SHA-256 & RIPEMD-160* | |
| *first round with SHA-256* | 6b88c087247aa2f07ee1c5956b8e1a9f<br>4c7f892a70e324f1bb3d161e05ca107b |
| *second round*<br>*with RIPEMD-160* | b67f99610e811d5eba9e<br>337877a8f55f766d7401f |

### 6.2.4 Difficulty factor in Bitcoin system

The output of the hash function of a block has to be under specific value called *target*. *Difficulty* is a measure about how hard is to find a hash bellow this target. The value of the difficulty factor changes on every 2016 blocks. This will produce, on average, one block every ten minutes (a rate of the block). The current difficulty factor and *hash-rate* (the speed at which a computer is completing an operation in the Bitcoin code) determine the number of generated Bitcoins that one individual can achieve daily. The first ”miner” computer that solve the puzzle (find a hash value below the given ”target”) will broadcast its block and its group of transactions is accepted as the next in the Blockchain, Fig. 6.9

We made analyses of how difficulty factor changes daily during August 2017 and annually in period from April 2017 to April 2018. In Fig. 6.10 and in Fig. 6.11 the results of those analyses are shown.

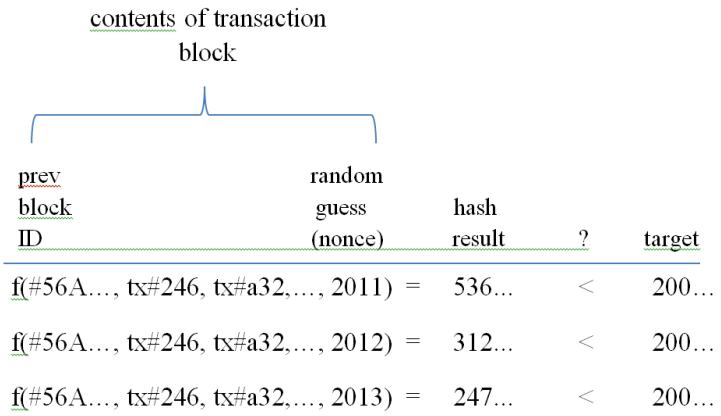



Figure 6.9: Hash function

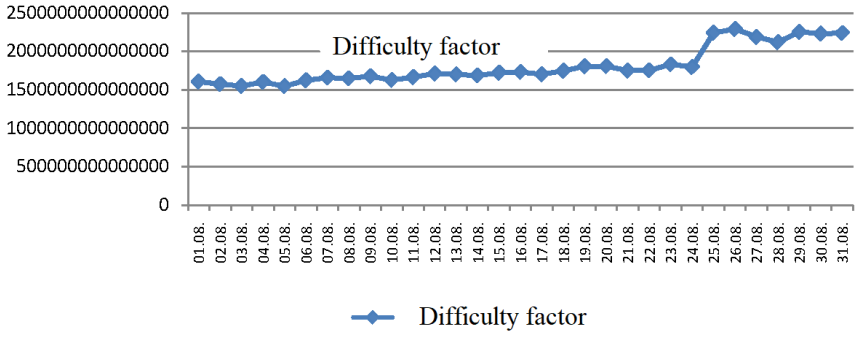



Figure 6.10: Difficulty factor in August 2017

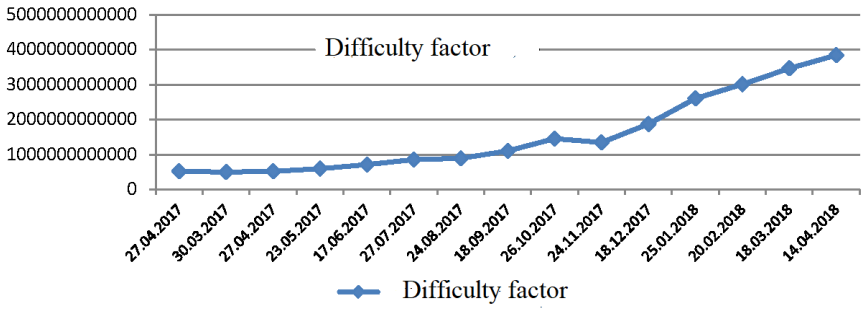



Figure 6.11: Difficulty factor from April 2017 to April 2018

### 6.2.5 Countries where Bitcoin is legal

Bitcoin is not related to any government as it is an independant currency. Whether it is legal or not, it depends on the country [22], [53].

- In *USA* Bitcoin is used partially depending on the state. Every individual who mines Bitcoins is subject to taxation. Since November 2013, Bitcoin is regarded as a ”legal means of exchange”.
- In *Australia* it is legal to trade, mine and buy Bitcoins all over the country. Bitcoin is regarded as real money since 1 July 2017.
- In 2017 the *japanese goverment* legalized Bitcoin as a method of payment. Now, the country is trying to protect the Bitcoin transactions.
- The Russian Deputy Finance Minister is thinking to legalize Bitcoin and other cryptocurrencies in 2018, although there is no offical report yet. *Russia* is also working on development of CryptoRuble - its own cryptocurrency [28].
- In *Belgium*: “The Minister of Finance indicated that government intervention with regard to the bitcoin system does not appear necessary at the present time” [53].
- The use of Bitcoin in *UK* is not regulated and Bitcoin is treated as “private money” which has to be VAT taxed when trading.
- *France* has certain regulations about Bitcoin use. Each exchange or Bitcoin wallet must contain user data, which is not a characteristic of Bitcoin.
- *Argentina, Italy* and *the Netherlands* do not prohibit it, but they are skeptical towards Bitcoin. Argentinian government manifest sceptisism due to the impossibility to establish control over the new currency by the authorities, which is interpreted as a fertile soil for illegal transactions.
- Bitcoin in *Macedonia* is still illegal.

## 6.3 Smart contract

Exchanging a value, property, shares, or anything of value in a transparent, conflict-free ways, between two owners based on a set of conditions is included in an

agreement called *smart contract*, which is controlled by decentralized consent on a Blockchain. As a traditional contract, a Smart contract defines the rules and penalties around an agreement. Furthermore, the Smart contract automatically enforses those obligations [26].

A program code, a storage file and an account balance are parts of the smart contract. A contract's storage file is stored on the Blockchain, while a contract's program logic is executed by the network of miners. Whenever a user or another contract sends a message the contract's code is executed. A contract can also receive and send money to other contracts or users into its account balance [10].

In Fig. 6.12 we give a simple model of a smart contract.

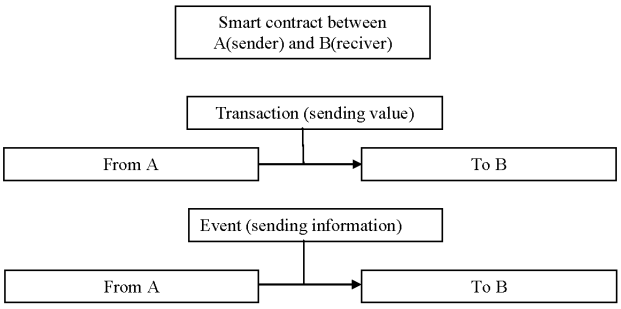



Figure 6.12: A simple model for a Smart contract

### 6.3.1 Application of smart contracts

In the following we will describe some applications of smart contracts.

– **Advantage of Blockchain technology in music industry**

  Blockchain can be also applied for a music distribution. Using this technology musicians and music companies that own music rights could receive appropriate funds anytime when their music is used for commercial purposes. Blockchain ledger can make a more direct connection between musicians

and fans. A music with a unique ID and timestamp can be made public on the ledger. In this way, problems with downloading, coping and modifying of the digital content can be prevented. Every record on the Blockchain ledger has metadata with information for ownership and the rights, so everyone can see it and verify. The monetization of the music is made with smart contract that enable payment transactions. In this way, consumers can choose the record and immediately pay to the owners using cryptocurrency [40].
Benji Rogers' on-line music platform is one of the companies that have made a globally decentralized Blockchain ledger as a solution for the problems of ownership, payment and transparency. The records of this Blockchain ledger are "dotBlockchain" - a new dynamic file format (one type of zip file) that carries all the information for each record.

– **Smart contracts in insurance policies**

The process of requirments in insurance policies is still manuel with a big amount of human involvement and it can take a long time to be paid. Smart contracts can autiomize this process. The advantages of smart contract are cost reduction, increased transparency and trust. The input conditions of the smart contract are recorded onto the Blockchain. And for example, in case of natural disaster, the claim process is activated automatically without the need of human action [47].

An example of the application of smart contracts in insurance policies is Etherisc, which recently began selling tokens using Blockchain technology.

– **How Blockchain could help the patent system**

The patent system is an instrument that helps innovators to utilize their products and protect their rights over the products. Since the patent system has still some weaknesses, using Blockchain could help to correct some of its disadvantages [3]. This can be done with the following two characteristics of Blockchain:

  – *Hashing*. All hashes are unique and even a small change in the input will result in a different hash. Same hash is produced only by hashing the identical input.

– *Proof of existence*. All hashes are recorded on the Blockchain by creating a record where the hash existed at a given time. Everyone can verify the record, but nobody can see its content. In order to prove the existence of that document at a specific time, the patent owners have to hash the identical copy of it again. This process can help innovators to protect their work by storing a hash of their patent description on the Blockchain.

## 6.4 Analysis of the possibilities for improvement of Blockchain technology

Since the process of verification on every transaction is very slow, there is a need for a third major innovation called a *Blockchain scaling*. A scaled Blockchain makes the process faster by investigating:

– how many confirmations are necessary to validate each transaction and to separate the work efficiently and

– the limits on the amount of transactions the bitcoin network can process.

This modification does not sacrifice security of Blockchain. A scaled Blockchain is expected to be fast enough to power the Internet of things and go parallel with the major payment middlemen (for example: VISA and PayPal) of the banking world [19].

### 6.4.1 The double spending problem

A scalable Blockchain makes the process faster without sacrificing security. In the mining process, a new block is made and included in a Blockchain approximately every 10 minutes. When the transaction is included in a block which is distributed in Blockchain network, the transaction is mined at depth of one block (one confirmation of the transaction). With each new block in the Blockchain, the depth of the existing blocks is increased by one. The transaction is verified when the block depth is at a specific level (six is a common number of confirmations) [44].

Double spending means using the same bitcoins more than once. Once a transaction is confirmed, it is impossible to be double-spend it. The probability of a

successful double spend calculated according to Poisson distribution [83] is given with this formula:

$$P = 1 - \sum_{k=0}^{z} \frac{\lambda^k e^{-\lambda}}{k!}(1 - (\frac{q}{p})^{z-k})$$

where

- $\lambda = \frac{zq}{p}$ is the mean;
- $z$ is the number of confirmations of transaction (the number of blocks by which "the honest" node has an advantage over the attacker). "The honest" nodea node that behaves to the way we expect nodes to behave, i.e., does not try to modify history, to properly serve blocks and transactions, to properly transmit messages, to properly transmit formatted messages and data, etc.;
- $q$ is the probability the attacker finds the next block (the attacker's hashrate);
- $p = 1-q$ is the probability an honest node finds the next block.

Table 6.2 and Fig. 6.13 presented the probability of successful double spend, as a function of the attacker's hashrate $q$, for different values of the number of confirmations $n$ [39].

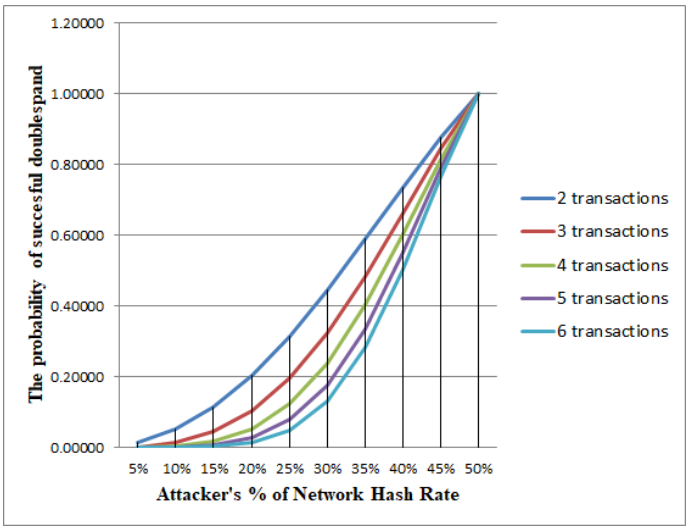



Figure 6.13: The probability of a successful doublespend

Table 6.2: The probability of success

| Attacker's % hash rate | 2 conf. | 3 conf. | 4 conf. | 5 conf. | 6 conf. |
|---|---|---|---|---|---|
| 5% | 0.01265 | 0.00164 | 0.00022 | 0.00003 | 0.00004 |
| 10% | 0.05098 | 0.01317 | 0.00346 | 0.00091 | 0.00024 |
| 15% | 0.11504 | 0.04423 | 0.01725 | 0.00678 | 0.00268 |
| 20% | 0.20393 | 0.10324 | 0.05300 | 0.02742 | 0.01425 |
| 25% | 0.31544 | 0.19612 | 0.12351 | 0.07836 | 0.04994 |
| 30% | 0.44572 | 0.32458 | 0.23913 | 0.17735 | 0.13211 |
| 35% | 0.58881 | 0.48446 | 0.40251 | 0.33637 | 0.28217 |
| 40% | 0.73640 | 0.66417 | 0.60340 | 0.55063 | 0.50398 |
| 45% | 0.87772 | 0.84440 | 0.81561 | 0.78979 | 0.76611 |
| 50% | 1 | 1 | 1 | 1 | 1 |

The number $6$ is chosen with an assumption that an attacker cannot possess more than $10\%$ of the whole hash rate and the risk of $0.1\%$ or less can be accepted [97]. But, for example, if the hash rate is around $40\%$, then $90$ confirmations are needed to have less than $0.1\%$ chance for success of the attack (an attack for double spending of the transaction called "alternative history attack").

### 6.4.2 How does Bitcoin system handle the double spending problem?

The following example explains the process of handling with the problem of double spending:

- Let A has one bitcoin and he want to send it to B. This transaction (called T1) goes to the pool of unconfirmed transactions and wait to be confirmed.
- At the same time A sends one bitcoin to C. This transaction (called T2) also goes into the pool and wait for confirmation.
- Let first transaction T1 is pulled out of the pool of unconfirmed transactions. Before the transaction is included into the Blockchain, its validity is checked. This transaction will be valid since A has one bitcoin and it is inserted into the Blockchain. Now, transaction T2 is pulled out of the pool, but it is invalid (since A has no more bitcoins) and will not be confirmed.

– If transactions T1 and T2 are validated simultaneously, then the Blockchain has two branches and the first one to reach the next block of confirmations will be confirmed.

– If T1 and T2 achieves the next block simultaneously, the race for the next block continue.

– Therefore it is recommended to wait 6 confirmations for completing transaction. As, we can see in Table 6.13, it is highly improbable that the transactions will reach the next block simultaneously more than 6 times. So, at the end only one transaction will be confirmed.

The ”fork” in the Blockchain is created when the original Blockchain code is updated, but only some of the Blockchain nodes accept the update. In effect, it is a change in the Blockchain protocol used by the software to decide whether the transaction is valid or not. In this way, the original Blockchain remains the same and the updated nodes separate from the original Blockchain and create a new Blockchain. Alternative history attack is 100% probable to succeed if the attacker is in control of more than half of the network hash rate. The attacker can continue with his private fork until the moment it becomes longer than the branch built by the honest network because he can now generate blocks faster than the other participants of the network [32].

### 6.4.3 Blockchain Scaling

Nakamoto introduced a block size limit of 1MB for every block in the public Blockchain. This was a security measure, so any block over that limit would be immediately rejected by the $P2P$ network. This limitation slows down the transactions and cannot keep up with the speed of other currencies payment and credit cards.

In Table 6.3 a review of the average number of transactions made with cryptocurrencies and standard credit card is given [42]. Bitcoin has a limit of 3 to 4 transaction per second (although theoretically process up to 7, but that number is never being reached). This is not the situation with private Blockchain. They can reach over 1000 transactions per second, because each node on the network in private Blockchain uses high-quality computer with strong bandwidth internet connection.

Table 6.3: Bitcoin and Eterium vs PayPal and Visa transactions per second

| **Currency** | **Transactions per second** |
|---|---|
| Bitcoin | 3 to 4 transactions per second |
| Etherium | 20 transactions per second |
| PayPal | 193 transactions per second average |
| Visa | 1667 transactions per second |

There are several solutions to improve Blockchain scalability that have been or will be implemented. Some of the major scalability solutions are analyzed in [75]:

- Segwit
- Block Size Increase
- Sharding
- Proof of Stake

*Segwit (Segregated Witness )* is an alternative solution for Blockchain scalability, through increasing the number of transactions in a block, without increasing the size of the block. Segregated witness helps to enlarge the space for new transactions by removing signature date from Bitcoin transactions. In fact, the proposal in segregated witness is the digital signatures of all input transaction to move to the end of the transaction data in part called *witness*. In this way ”validating” part has been separated from the ”effective” part of the transaction [37]. The digital signature takes 65 percent of transaction. If it is removed, more space will be free up in the bitcoin’s block for more transactions. A new unit for transaction size is introduced by segwit. Now, the transaction is divided in two parts: non-witness data (which must be stored on the block like before) and witness data (which is moved to the extended block). Non-witness data byte counts as 4 WU (weight units) each, while witness data byte only counts as 1 WU each. The maximum capacity of a block is 4 kWU, which would correspond to the old maximum block size of 1MB if no one uses segwit. The segwit upgrade to the Bitcoin protocol occurred in August 2017. A new format is used by around 30 percent of transactions [37].

According to BitMEX research the percentage of transaction that use segwit is given on Fig 6.14.

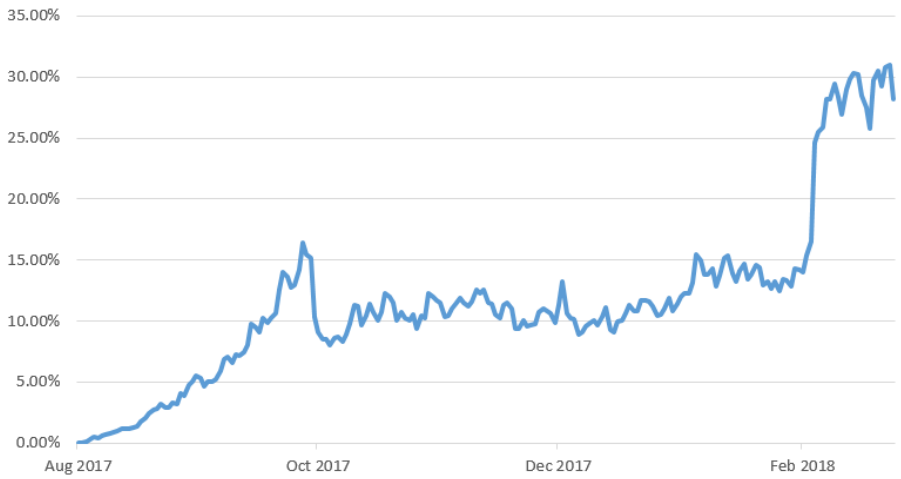



Figure 6.14: Percentage of transactions that use segwit

*Block Size Increase*. In the bitcoin's Blockchain the block size is limited to a maximum of 1MB. There are several arguments for and against increasing the size of blocks. A major argument against block size increase is that it will cause increased centralization. Each connected computer on the bitcoin network is called a node. There are two types of nodes in bitcoin Blockchain: full nodes and lightweight (or partial) nodes. Full nodes validate every block and transaction. Therefore, full nodes must store a copy of the complete Blockchain ledger locally (more than 165 GB). The lightweight nodes do not store the complete ledger. They use a simplified payment verification (SPV) mode which only requires to download a copy of the block headers of the longest proof of work chain [83].

On the other hand, a miner creates blocks in the Blockchain that the nodes keep. Bitcoin network is maintained by roughly 10.000 full nodes, while the number of miners estimated to be around 100.000. Increasing the block size makes the full nodes more expensive to operate. This leads to less hashers running full nodes and centralized entities would have more power, that weakens bitcoins value proposition. Miners have benefit from increasing of the block size, since increased block size means more transactions per block. This will increase the amount of transaction fees that a miner can make from mining a block.

*Sharding*. One of the biggest problem for cryptocurrencies is the speed of

transaction verification. Each full node in the network has to store the entire Blockchain ledger. Sharding breaks down a transaction into shards, spreads it among the network and nodes work on individual shards side-by-side. On this way, the overall time taken is decreased. A normal block has a block header and a body that contains all transactions in the block. The Merkle root (hash of all hashes from all block transactions) of all transactions is in the block header. Sharding changes this into two levels of interaction.

The first level is the transaction group which is divided into a transaction group header and a body. Each shard has its own group of transactions. The transaction group header is divided in left and right part. The left part contains: shard ID, pre state root (state of the shard root before the transactions were applied), post state root (state of the shard root after the transactions are applied) and receipt root (receipt root after all the transactions in the shard are applied). The right part is full of randomly chosen validators who verify transactions in the shard. The transaction group body has the transaction's IDs of all transactions in the shard.

The second level is the normal Blockchain, but here it contains two primary roots: the state root (represents the entire state which is broken down into shards) and the transaction group root (contains all transactions groups inside that block). Level two is like a simple Blockchain, which accepts transaction groups rather than transactions.

By dividing the pieces, many parallel transactions can take place at the same time and therefore in this way the performance is improved, i.e. there is an increase in throughput (the maximum rate at which transactions are made) and a decrease in latency (time to confirm that the transaction is involved) in the Blockchain network [24].

Zilliqa, a new Blockchain platform based on the technology of sharding that solves the scalability issue faced by current Blockchain platforms like Bitcoin and Ethereum, has announced the release of their public test net. In Table 6.4 comparison of the capability of processing transactions in Ethereum and Zilliqa is given.

If the number of nodes in Zilliqa is double to 3600 the throughput scales as well to 2488 transactions per second ([45], [101], [31]). From Table 6.4 it is clear that Zilliqa even with less number of nodes is able to process much more transactions per second than Ethereum or Bitcoin.

*Proof of Stake*. Most cryptocurrencies follow the proof of work protocol, which means that miners mine cryptocurrencies by solving crypto-puzzles using dedicated hardware. In proof of stake protocol there are validators instead of miners. The validator have to invest some of his assets as stake. Then the validator starts validating blocks on the following way: if he sees a block that he thinks can

Table 6.4: Comparison of capability of processing transactions in Ethereum and Zilliqa

| | **Number of full nodes** | **Number of transactions per sec.** |
|---|---|---|
| Ethereum | 25000 | $15 - 20$ |
| Zilliqa | 1800 | 1218 |

be added to the Blockchain, he validates the block by putting a bet on it. If the block gets appended to the Blockchain, the validator will receive a reward proportional to the invested stake. If the validator bet a malicious block, the stake he has invested will be taken from him. Ethereum implement proof of stake protocol using the Casper consensus algorithm [24]. The advantage of using proof of stake protocol instead of proof of work is that it uses considerably less energy and as a result is more cost effective.

Table 6.5: Comparison of proof of work and proof of stake protocols

| | *Proof of work* | *Proof of stake* |
|---|---|---|
| Energy consumption | High | Low |
| Required tools | Mining equipment | No equipment necessary |
| Security | High | Untested |
| Decentralized vs. Centralized | Tends to centralize | Users can remain in control of their tokens |

In Table 6.5 a comparison of some parameters of proof of stake and proof of work protocols is given [30].

## 6.5 Secure Big Data and IOT with implementation of Blockchain

As we said before Blockchain is a distributed ledger that records transactions in blocks that form a chain. This chain is secure, immutable and transparent. Therefore, solutions that use Blockchain technology can be a pillar of data handling and processing systems within organizations. Integrating Blockchain with Big Data has many advantages because it enables better management of enormous volumes and variety of information. Big Data and Blockchain are correlative: Big Data has processing capacities that can handle Blockchain complexities and its big expansion and vice versa. Implementation of Blockchain in Big Data confirms that data is accurate and secure and sharing of data will become more simple.

### 6.5.1 What is Big Data?

*Big Data* is a term used for a collection of large and complex data sets. For example, Facebook has collected 300 petabytes (PB) of personal data since its inception, They kept the records, sent messages, published videos, e.t.c. A big data are transaction data from online transactions and shopping.

It is hard to process and analyze Big Data with traditional techniques, because it refers to a big volume of structured and unstructured data. However, today almost every company uses Big Data to achieve a competitive advantage in the market. With this in mind, large data processing and analysis tools are the most favorable choice for companies in terms of costs and other benefits. Different types of data, IoT and unstructured sources can be stored and processed with tools such as Hadoop, Plotly, Bokeh, etc.
Studies show that 2.5 quintillion bytes of data are generated daily.

Despite such a contribution to the effective management of large amounts of data, there is a growing concern about user privacy and big data security. There are many examples that show it, like the large-scale scientific experiment conducted by Facebook without informing its users explicitly or the government has often come under attack for its observation on its citizens without their explicit permission. Today the great businesses are careful in accepting big data, in order to ensure their security and privacy [38].

### 6.5.2 Benefit of application of Blockchain in Big Data

Implementation of Blockchain in Big Data confirms that data is accurate and secure and sharing of data will become more simple.

Some of the benefits of using Blockchain technology in Big Data analytics are [49]:

- reduces costs (significantly reduces storage costs);
- increasing traceability (each product or document has "digital password" which ensures tracking of its origin and journey);
- enhanced data quality (data is complete and structured, weak points in a big data analytics which increase accuracy and makes analysis easier);
- facilitates data access (users from different departments can access the data for the analysis process and this shortens the time cycle of data access and analysis);
- enhancing security (the system is decentralized and transparent, so the risk of fraudulent activities is reduced).

**Analyses of several implementations of Blockchain technology in Big Data**

- *Speeding up the financial service industry.* The connection of Blockchain and Big Data in financial institutions will allow to estimate risk and identify suspicious patterns in real time. Using Blockchain as a means of conducting transactions will help to protect banks and their customers from fraud, to speed up the process of transactions and reduce the cost of money transfers [33].

  For example, in the last few years, in order to simplify money transfers between bank accounts using Blockchain technology, an organization of 47 Japanese banks joined a Blockchain startup called Ripple. The aim of this is to perform real-time transfers at a lower cost. The traditional real-time transfers are expensive, because of the potential risk factors as double spending that can be avoided with Blockchain technology. Big Data analyses together with Blockchain can identify risky transactions faster than the current ones. This reduces the cost with real-time transactions [54].

- *Security in industries outside of banking.* To handle data and prevent hacking and data leaks, healthcare, public administration, enterprises have started using Blockchain.

  For example:

  - *Healthcare*: a project at the MIT Media Lab known as Medrec is starting to create Blockchain system that gives priority to patient agency, which authorizes patients to share their records. In order to make the sharing of permissions to access data easier, Medrec uses a private Ethereum chain [4].
  - *Insurance*: A controlled master policy from the UK, and three local policies from the U.S., Singapore and Kenya, have been written into a smart contract that enables a shared insight into policy data and documentation in real-time. Clarity into coverage and premium payment at the local and master level is provided by Blockchain. It also enables automated alerting to participants in the network after payment events [46].

- *Supply Chain Monitoring.* The goods included in Supply Chain are added to the Blockchain and a Mobile App is used to monitor the status of the goods while being transported. The benefit of using Blockchain in Supply Chain are [46]:

  - reduce or eliminate fraud and errors;
  - improve inventory management;
  - minimize courier costs;
  - reduce delays from paperwork;
  - identify issues faster;
  - increase consumer
  - partner trust.

  For example, in order to improve food safety by increasing the monitoring of products from the place of origin to the time it is sold to the consumer, Walmart uses Blockchain technology. In this way, users get credible insights of the origin of food. Receiving unchangeable, credible and traceable data is of great importance to success of Walmart's operations, because it produces 40 petabytes of data every day [38].

**Examples for improving Big Data using Blockchain**

In this subsection we present several examples of implementation of Blockchain in Big Data.

- Example 1:GOLEM
  GOLEM network is a decentralized network of computers that utilizes computer power of the network users to make the supercomputer functional. Laptops, desktops, and data centers are just a few examples of devices that can contribute computing power to the network.The Golem network aims to create a decentralized computing power sharing economy, where anyone can make money by ”renting” their computing power or developing and selling software. In fact, it aims to make computing power much cheaper than it is today, thereby reducing the cost of using it [23].

- Example 2: PATH
  Path aspires to unite both Blockchain and Big Data allowing users to rent their extra bandwidth. This comes as a result of the fast acquiring of Blockchain technologies among people and companies, for example, a company wants to have an insight into how its website is presented to the public or how much coverage its network has at a given time of day, etc. ”Path Mining Nodes” are installed onto companies' computers and they work passively in the background - earning tokens, and in order to ensure the working insights of the companies mentioned earlier. This is just one of many Blockchain platforms that allow companies to take advantage of the enormous need for companies to improve the data sets they analyze and use for their products and services. The real value of Blockchain data is the quality and how it is stored on the public ledger. Thus, this platform, together with the others Blockchain platforms that help companies to improve data, can become middlemen between companies and the users [52].

- Example 3: ESTONIA
  ESTONIA strives to be the most advanced digital society based on Blockchain. It uses Keyless Signature Infrastructure (KSI)-based Blockchain technology. KSI is used to store public data in a Blockchain, designed to provide a scalable digital signature using a cryptographic hash function. Using this, the government can observe any changes in the database thus ensuring that the data is transparent. This has two benefits: cuts down external falsified/tampered records, and makes harder for unauthorized government officials to interfere with information and data [38].

– Example 4: SKRY
Financial institutions, law enforcement, and Bitcoin companie's can detect suspicious activities using the first product of Skry. It can also identify legal and illegal entities, which enables following regulations and investigation of cyber-crimes like ransomware extortions, terror financing or drug trafficking on the dark web.

### 6.5.3 Internet of Things (IoT) and Blockchain

Application of Blockchain in IoT enables IoT devices to participate in Blockchain transactions and invents new styles of digital interactions. This technology will provide a simple infrastructure for devices to directly and securely transfer data or money using Smart contract.

#### Challenges of a secured IoT model

The current IoT ecosystem is based on a centralized model – client/server model, where all devices are identified, authenticated and connected through cloud servers. This structure is the biggest challenge of IoT security. Even, if the devices are at a short distance the connection between them will have to go through the cloud. These models continue to be used in today IoT networks, although they might not be able to handle the growing needs of giant IoT ecosystems in the future. The costs for existing IoT solutions, that use centralized model, are another big obstacle for the future growth of the IT network. The clouds, large server farms, and networking devices have higher costs for infrastructure and maintenance. Also, in order to protect IoT devices and platforms form physical tampering, new security technologies are needed. In addition, many devices in IoT system use simple operating systems and processors that not support advanced security systems [43].

Blockchain in IoT can be used in tracking connected devices, processing of transactions and coordination between devices. The decentralized nature of Blockchain technology will produce a more flexible system for running the devices and with the cryptographic algorithms data will be more private. The Blockchain ledger cannot be manipulated by malicious users since it does not exist in any single location, which means that it is a tamper-proof (the data stored on the Blockchain is tamper-proof and cannot be changed). Also, Blockchain enables secure peer-to-peer messaging between IoT devices. Capabilities of Blockchain, such as

decentralization, autonomy and confidentiality, make this technology ideal for a fundamental component of IoT solutions [43].

### 6.5.4 Example of IoT and insurance

Usage-based insurance (UBI) models are some of the new products that will develop accurate actuarial models which is valuable to insurers. For example, in the auto insurance, we can use encrypted data for driving times, distances, acceleration, braking and other behaviors for identification of high-risk drivers and validation of the information included in the applications. On this way, we can provide consumers bigger control over their premiums. The question is how to manage the enormous volume of data and logic because of the communication between millions devices. Large complex networks can be managed by Blockchain with devices communicating between each other on a peer to peer basis, securely and accurately, instead of building an expensive data center. This type of managing devices is cheaper than the data central model [48].

# Chapter 7

# Network Coding based on quasigroups

In the first chapter, we explained the impact of quasigroups and quasigroup transformations on the construction of cryptographic primitives and in designing error detection and error correction codes. We have listed the reasons for this, such as: the structure of quasigroups, their large number, the properties of quasigroup transformations and others. In this chapter we will see how quasigroup transformations can be applied in network coding. Namely, we will propose an application of these quasigroup transformations in network coding in order to provide faster and simpler decoding process.

## 7.1 Network coding

Network coding is a promising technique that improves network throughput and provides high reliability. The transmitted data is encoded in order to increase throughput, reduce delays and make the network more robust. In a single node, algebraic algorithms are used to generate output messages by encoding its received messages. Then, the received messages are decoded and transmitted to their destinations. In this way, fewer transmissions are required to transfer all the data, but this requires more processing at intermediary and terminal nodes. The most popular network coding is the linear network coding. It is a mathematical technique that improves throughput, efficiency and scalability of the network, as well as resistance to attacks and eavesdropping. In linear network encoding, instead of simply transferring packets of information, nodes generate new packets that are linear combinations of earlier received packets, multiplying them by coefficients selected from the final field. The nodes receive these network coded messages and collect them in a matrix. The original messages can be recovered by performing

Gaussian elimination on the matrix [5], [63], [58].

A butterfly network presented in Fig. 7.1 is a form of network topology that can be used for connecting different nodes in a multiprocessor system [20]. This network is often used to illustrate how network coding can outperform routing. Two source information $M^1$ and $M^2$ must be transmitted to two destination nodes $R_1$ and $R_2$. Each node can carry only single information. The central link would be only able to carry $M^1$ or $M^2$, but not both. Suppose we send $M^1$ through the center, then $R_1$ would receive $M^1$ twice and not receive $M^2$ at all. Sending $M^2$, poses a similar problem for $R_2$. So, the routing is insufficient because no routing scheme can transmit both $M^1$ or $M^2$ simultaneously to both destinations. Fig 7.1 shows the "butterfly" network where the messages $M^1$ and $M^2$ can be transmitted to the end nodes $R_1$ and $R_2$, simultaneously, using encoding of the messages $M^1$ и $M^2$ with $M^1 + M^2$.

The complexity of decoding due to the use of Gaussian method of elimination gives us an idea of constructing a new algorithm for quasigroup network coding, with the aim of increasing the throught in the "butterfly" communication network.

Our idea for network coding based on quasigroups is presented on a simple network shown on Fig. 7.2. In this algorithm we use:

– Quasigroup $(Q, *)$ of order $2^n$ together with its parastrophe $\backslash$ .

– $E_{l,*}$ - transformation with leader $l$ and quasigroup operation $*$ (for coding and encrypting of messages) and $D_{l,\backslash}$ -transformation with the same leader $l$ and the corresponding quasigroup operation $\backslash$ (for decoding and decrypting).

*Network Coding Algorithm*

At the source $S$, a finite amount of information is generated and multicast to other nodes on the network. Let, $M^1$ and $M^2$ are original messages of $m$ symbols from $Q$.

**Step 1.** The node $A$ receives the message $M^1$, applies on it the $E$ - transformation with a given lider $l_1$ and it sends the coded message $E_{l_1}(M^1)$ to the intermediate node $B$ and to the receiver $R_1$. In the same way the node $C$ receives the messages $M^2$, applies on it the $E$ - transformation with a given lider $l_2$ and it sends the encoded message $E_{l_2}(M^2)$ to the intermediate node $B$ and to the receiver $R_2$.

**Step 2.** The intermediate node $B$ makes an addition modulo $2^n$ of two received messages $E_{l_1}(M^1)$ and $E_{l_2}(M^2)$ and sends the obtained message $b$ to node $D$.

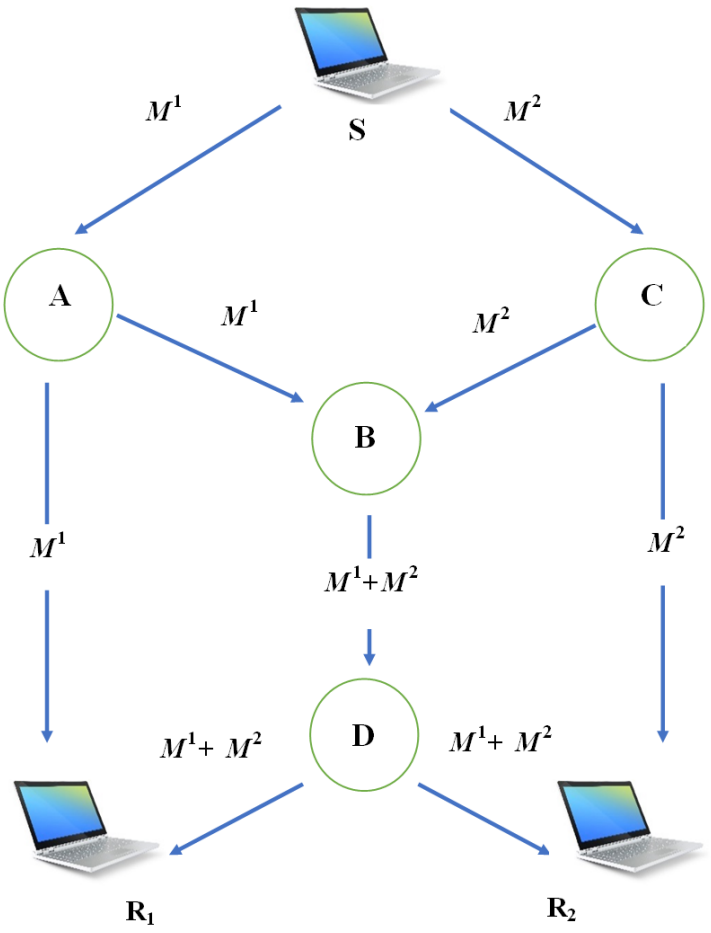



Figure 7.1: Butterfly network

**Step 3.** The node $D$ sends encoded message $E_{l_3}(b)$ to receivers $R_1$ and $R_2$. In this way, the node $R_1$ receives $E_{l_1}(M^1)$ and $E_{l_3}(b)$, and node $R_2$ receives $E_{l_2}(M^2)$ and $E_{l_3}(b)$.

Destinations nodes $R_1$ и $R_2$, decoded original information $M^1$ and $M^2$ on the following way:
*Decoding Algorithm*

In node $R_1$:

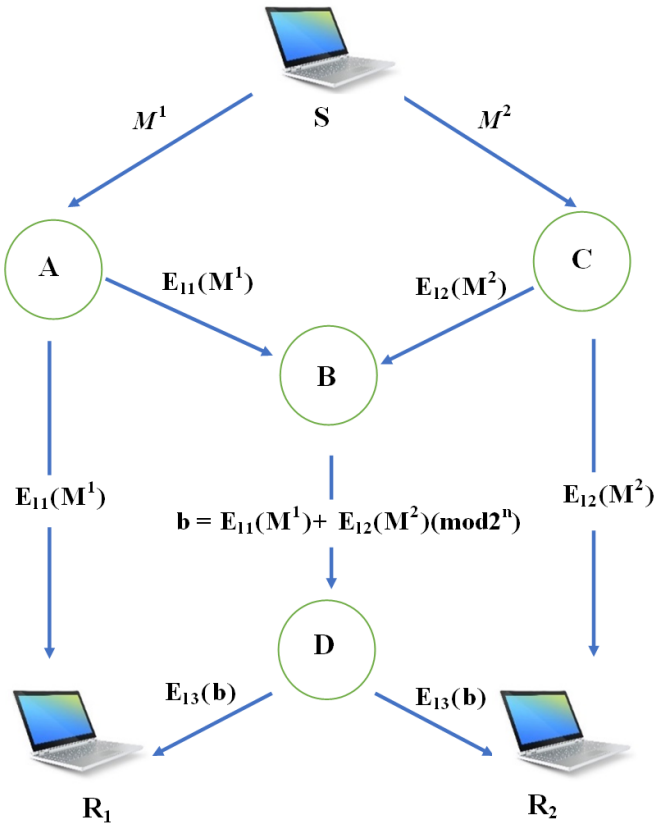



Figure 7.2: Network coding based on quasigroup

The received messages are $E_{l_1}(M^1)$ and $E_{l_3}(b)$.
**Step 1.** The message $b$ is obtained applying inverse transformation $D_{l_3}$ on $E_{l_3}(b)$, i.e., $b = D_{l_3}(E_{l_3}(b))$
**Step 2.** The message $E_{l_2}(M^2)$ is obtained with subtraction modulo $2^n$ of $b$ and $E_{l_1}(M^1)$.
**Step 3.** The original messages $M^1$ and $M^2$ are obtained by applying the inverse transformations $D_{l_1}$ and $D_{l_2}$ on $E_{l_1}(M^1)$ and $E_{l_2}(M^2)$ correspondingly, i.e., $M^1 = D_{l_1}(E_{l_1}(M^1))$ and $M^2 = D_{l_2}(E_{l_2}(M^2))$.

In node $R_2$:
The received messages are $E_{l_2}(M^2)$ and $E_{l_3}(b)$.

**Step 1.** The message $b$ is obtained applying $D_{l_3}$ on $E_{l_3}(b)$, i.e., $b = D_{l_3}(E_{l_3}(b))$
**Step 2.** The message $E_{l_2}(M^2)$ is obtained with subtraction modulo $2^n$ of $b$ and $E_{l_2}(M^2)$.
**Step 3.** The original messages $M^1$ and $M^2$ are obtained by applying the inverse transformations $D_{l_1}$ and $D_{l_2}$ on $E_{l_1}(M^1)$ and $E_{l_2}(M^2)$ correspondingly, i.e., $M^1 = D_{l_1}(E_{l_1}(M^1))$ and $M^2 = D_{l_2}(E_{l_2}(M^2))$.

Next we will give an example for process of coding and decoding with this algorithm using quasigroup of order $2^2$ .

The process of encoding and decoding with this algorithm using a quasigroup of order $2^2$ is described by the following example:

**Example 7.1.** *Let $Q = \{0, 1, 2, 3\}$ . We consider the following quasigroup operation $*$ and the corresponding operation $\setminus$ on $Q$.*

| $*$ | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0 | 0 | 2 | 1 | 3 |
| 1 | 1 | 3 | 2 | 0 |
| 2 | 2 | 0 | 3 | 1 |
| 3 | 3 | 1 | 0 | 2 |

(7.1)

| $\setminus$ | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0 | 0 | 2 | 1 | 3 |
| 3 | 0 | 1 | 2 | 0 |
| 2 | 3 | 0 | 1 | 1 |
| 1 | 2 | 3 | 0 | 2 |

(7.2)

– *Coding*
*If $M^1 = 1233$, $M^2 = 2323$, $l_1 = 1$, $l_2 = 2$ u $l_3 = 3$, then:*

– *A sends $E_1(1233) = 2032$ to $B$ and $R_1$,*

  - *$C$ sends $E_2(2323) = 0310$ to $B$ and $R_2$*
  - *$B$ receives $E_1(1233) = 2032$, $E_2(2323) = 0310$ and sends the message $b = (2032 + 0310)$ mod $4 = 2302$ to $D$.*
  - *Then $D$ sends $E_3(2302) = 1002$ to $R_1$ and $R_2$.*

- *Decoding*
  *$R_1$ receives the next one:*

  - $E_1(1233) = 2032$
  - $E_3(2302) = 1002$

  *Source messages are obtained with the following operations:*

  - $b = D_3(1002) = 2302$
  - $E_2(1233) = (2302 - E_1(1233))$ *mod* $4 = (2302–2032)$ *mod* $4 = 0310$
  - $M_1 = D_1(2032) = 1233$
  - $M_2 = D_2(0310) = 2323$

  *$R_2$ receives the next one:*

  - $E_2(2323) = 0310$
  - $E_3(2302) = 1002$

  *Source messages are obtained with the following operations:*

  - $b = D_3(1002) = 2302$
  - $E_1(2323) = (2302 - E_2(2323))$ *mod* $4 = (2302–0310)$ *mod* $4 = 2032$
  - $M_1 = D_1(2032) = 1233$
  - $M_2 = D_2(0310) = 2323$

This new proposed algorithm has several advantages over linear network coding. Here, decoding uses much simpler operations than in linear network coding. Namely, instead of solving a system of linear equations using method of Gaussian elimination, we apply only a $D$-transformation. Therefore, our algorithm provides faster decoding process. Also, due to the already proved cryptographic properties of used quasigroups transformations this algorithm provides an information security of transmitted data. These results are published in [81].

# Chapter 8

# Algorithm for reducing storage in Blockchain based on secret sharing

As we mentioned before, Blockchain is a distributed database of records or public ledger of all time stamped transactions saved in all computers in one $P2P$ network. This is a growing list of linked blocks [75].The blocks consist of valid transactions, a timestamp and a hash pointer as a link to the previous block in the chain. Each transaction in the public ledger need to be verified by participants with consensual majority. Information that is entered in the ledger can never be changed or deleted. The Blockchain ledger contains an information for every transaction ever done [7].

Since each node needs to store all transactions, the storage space rapidly increases and storage cost of an entire Blockchain network grows. Therefore, there is a need of changing the storage concept in Blockchain. In Fig. 8.1 the total size of all block headers and transactions in Bitcoin Blockchain over the last year is given.

In [8] authors propose a solution for this problem based on network coding. They divide a block with size $G$ into $k$ equal length packets. These $k$ packets are encoded into $n$ encoded packets using linear combinations of original packets.

The concept of *Distributed Storage Blockchain* (DSB) is introduced in [94]. In Distributed Storage Blockchain the transaction block is encrypted using different private keys and then distributed to subsets of nodes. Also, hash values and encryption keys are stored among the peers of the subset using a secret-sharing scheme. DSB greatly reduces the storage space in a Blockchain system. In order to improve Distributed Storage Blockchain a local secret-sharing scheme is proposed in [56].

In [95] authors enhance Blockchain performance and reduce storage costs using coding-theoretic techniques. In this part, we propose how to reduce the stor-

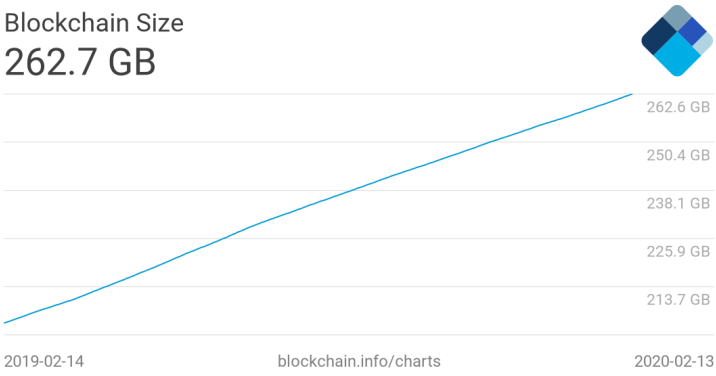



Figure 8.1: Blockchain size [21]

age space in Blockchain through a novel use of Shamir's secret sharing scheme. Therefore, we will first explain the concept of this scheme.

## 8.1 Concept of Shamir's Secret-Sharing scheme

In [100] Shamir introduced the concept of secret sharing through a $(k, n)$ threshold scheme. His model was based on polynomials. The concept of Shamir's secret-sharing scheme is to distribute a secret $S$ to a group of $n$ participants, such that each participant receives one share of the secret. Knowledge of any $k$ or more shares makes $S$ easy to compute, but less than $k$ shares do not reconstruct the secret message $S$. In order to share the message $S$, a random $(k–1)$-degree polynomial $f(x) = a_0 + a_1x + a_2x^2 + \ldots + a_{k-1}x^{k-1}$ is chosen in this way:

- the secret message $S$ is the constant term, i.e., the coefficient $a_0$;
- $a_1, a_2, \ldots, a_{k-1}$ are randomly chosen coefficients from a finite field $F_p$, where $p$ is prime number;
- the $n$ points (shares) $(i, f(i))$ for different values of $i$ (for instance $i = 1, 2, \ldots, n$) together with the prime $p$, are given to $n$ participants (nodes) [100] [9].
- every participant obtains one point $(i, f(i))$ and with any subset of $k$ of these points, the coefficients of the polynomial can be find using interpolation.

Shamir's ($k$, $n$) secret sharing scheme has the several advantages. Some of them are:

- For fixed $k$, we can dynamically add or delete pieces without changing the other pieces;
- We can easily improve the security without changing the secret, constructing only new shares for the participants, keeping the same polynomial constant term;
- If the hierarchy of participants is important, we can give a different number of pieces to participants according to their level in the hierarchy.

## 8.2 Reducing Blockchain storage using Shamir's Secret-Sharing Scheme

In this Section we propose a simple algorithm for reducing Blockchain storage based on Shamir's Secret Sharing described above. The algorithm is the following one.

**Step 1**. Divide the transaction block $B$ into $k$ equal length packets, denote as $b_0, b_1, \ldots, b_{k-1}$, that are expressed as integers.
(All following arithmetic operations are performed in a finite field $Z_p$, where $p$ is a prime number larger than number of nodes $n$ and $b_i$, for $1 \leqslant t \leqslant k$.)

**Step 2**. Construct a $(k$–$1)$-degree polynomial $f(x)$ with coefficients $b_0, b_1, \ldots,$ $b_{k-1}$ i.e.,

$$f(x) = b_0 + b_1 x + b_2 x^2 + \ldots + b_{k-1} x^k\text{–}1$$

**Step 3**. Compute $n$ shares $(i, f(i))$ for different values of $i$ (for instance $i = 1, 2, \ldots, n$) and distribute this shares to corresponding Blockchain nodes.

Each node obtains one point $(i, f(i))$ and the coefficients of the polynomial can be find knowing any $k$ of these shares, using interpolation. With this, the block will be recovered since the coefficients of the polynomial are parts of the block. So, the reconstruction process is reduced to the interpolation of the polynomial.

On this way, each node in a Blockchain system stores one share, instead of whole block. Therefore, we reduce the storage space and the recovery process is simple. With the following example we will illustrate the concept of our algorithm.

**Example 8.1.** *Let $B = 123416$ is a block. We divide it into 3 equal length pieces:*

$$b_0 = 12, b_1 = 34, b_2 = 16$$

*We take $p = 37$ and form the 2-degree polynomial*

$$f(x) = 12 + 34x + 16x^2$$

*If we want to share this block to 4 nodes, we compute 4 shares $(i, f(i))$ for different values of $i = 1, 2, 3, 4$. We obtain points $(1, 62), (2, 144), (3, 258), (4, 404)$ and we distribute these points to corresponding 4 Blockchain nodes.*

*Reconstruction process::*

For reconstruction of the message $B$ any 3 points are enough. For example, if we know points: $(1, 62), (2, 144), (3, 258)$ we will compute Lagrange basis polynomials:

$$l_1 = \frac{x-2}{1-2} \cdot \frac{x-3}{1-3} = \frac{x-2}{-1} \cdot \frac{x-3}{-2} = \frac{x^2}{2} - \frac{5x}{2} + 3$$

$$l_2 = \frac{x-1}{2-1} \cdot \frac{x-3}{2-3} = \frac{x-2}{1} \cdot \frac{x-3}{-1} = -x^2 + 4x - 3$$

$$l_2 = \frac{x-1}{3-1} \cdot \frac{x-2}{3-2} = \frac{x-1}{2} \cdot \frac{x-2}{1} = \frac{x^2}{2} - \frac{3x}{2} + 1$$

Then, using these polynomials we obtain the original polynomial:

$$f(x) = l_1 \cdot 62 + l_2 \cdot 144 + l_3 \cdot 258 =$$

$$= (\frac{x^2}{2} - \frac{5x}{2} + 3) \cdot 62 + (-x^2 + 4x - 3) \cdot 144 + (\frac{x^2}{2} - \frac{3x}{2} + 1) \cdot 258 =$$

$$= 12 + 34x + 16x^2$$

From the coefficients of the obtained polynomial, we reconstruct the original message $B = 123416$.

# Bibliography


[1] Bakeva V., Popovska-Mitrovikj A., Mechkaroska D., Dimitrova V., Jakimovski B., Ilievski V.: *Gaussian channel transmission of images and audio files using cryptcoding*, IET Communications, Vol. 13, Issue 11, (2019), p. 1625 – 1632. (**IF. 1.779**)

[2] Belousov V. D.: *$n$-arnie Kvazigruppi* (*n-aryQuasigroups*), Stiinca, Kisiniev, 1972.

[3] Boucher P. ,Figueiredo Do Nascimento S., Kritikos M.: *How Blockchain technology could change our lives*, European Parliament. PE 581.948, 2017.

[4] Casey M., Crane J., Gensler G., Johnson S., Narula N.: *The Impact of Blockchain Technology on Finance*: A Catalyst for Change, Geneva, International Center for Monetary and Banking Studies 2, 2018.

[5] Chou P. A., Wu Y., Jain K.: *Practical network coding*, Allerton Conference on Communication, Control, and Computing, October 2003.

[6] Cover T.M., Thomas A.J. : *Elements of Information Theory.*

[7] Crosby M., Nachiappan, Pattanayak P., Verma S. ,Kalyanaraman V., *Blockchain technology: Beyond bitcoin*, Applied Innovation Review, Berkeley, Issue No.2, June 2016.

[8] Dai M., Zhang S., Wang H., and Jin S., *A low storage room requirement framework for distributed ledger in Blockchain*, IEEE Access, vol. 6, 2018, pp. 22970–22975.

[9] Dawson E., Donovan D.:*The breadth of Shamir's secret-sharing scheme*, Computers & Security, Volume 13, Issue 1, 1994, ISSN 0167-4048, Pages 69-78.

[10] Delmolino K., Arnett M., Kosba A.E., Miller A., Shi E.: *Step by step towards creating a safe Smart contract: lessons and insights from a cryptocurrency lab.*, IACR Cryptol ePrint Arch 460, 2015.

[11] Denes J., Keedwell A. D: *Latin Squares and their Applications*, The English Universities Press Ltd., 1974.

[12] Dimitrova V., Bakeva V., Popovska-Mitrovikj A., Krapež A.: *Cryptographic Properties of Parastrophic Quasigroup Transformation*, Markovski S., Gusev M. (eds.) ICT-Innovations 2012, Springer (2012), pp. 235-243.

[13] Dimitrova, V., Markovski, J.: *On Quasigroup Pseudo Random Sequence Generators*, Proc. 1st Balkan Conference in Informatics, Thessaloniki, Greece, 2003, pp. 393–401

[14] Dimitrova V., Markovski S.: *Classification of quasigroups by image patterns*, Proc. of the Fifth International Conference for Informatics and Information Technology, Macedonia, (2007) pp. 152-160.

[15] Gligoroski D., Dimitrova V., Markovski S.: *Quasigroups as Boolean functions, their equation systems and Groebner bases*, Book: "Groebner Bases, Coding, and Cryptography", ISBN 978-3-540-93805-7, Springer 2009, pp. 415-420.

[16] Gligoroski D., Markovski S., Kocarev Lj.: *Error-correcting codes based on quasigroups*, Proc. 16th Intern. Confer. Computer Communications and Networks (2007), pp. 165-172.

[17] Gligoroski D., Markovski S., Kocarev Lj.: *Totally asynchronous stream ciphers + Redundancy = Cryptcoding*, S. Aissi, H.R. Arabnia (Eds.): Proc. Internat. Confer. Security and management, SAM 2007, Las Vegas, CSREA Press (2007) pp. 446-451.

[18] Godoy W., Periera D.: *A proposal of a cryptography algorithm with techniques of error correction*, Computer Communications 20 (1997), pp. 1374-1380.

[19] Gupta V. , *A brief history of Blockchain*, Harvard Business Review, Fevruary 28, 2017.

[20] Fragouli Ch.,Soljanin E.: *Network Coding Fundamentals*, Foundations and Trends in NetworkingVol. 2, No. 1 (2007) 1–133, 2007

[21] https://api.blockchain.info/charts/preview/s-t/blocks-size.png?lang=en&start=1550148820

[22] https://atozforex.com/news/top-countries-where-bitcoin-is-legal/

[23] https://golem.network/index.html

[24] https://blockgeeks.com/guides/blockchain-scalability/

[25] https://blockgeeks.com/guides/ethereum/

[26] https://blockgeeks.com/guides/smart-contracts/

[27] https://blog.dataone.io/big-data-and-blockchain-c84c6c02544c

[28] https://cointelegraph.com/news/russian-minister-we-will-never-consider-bitcoin-legal

[29] https://cointelegraph.com/tags/segwit

[30] https://cryptocurrencyhub.io/an-introduction-to-consensus-algorithms-proof-of-stake-and-proof-of-work-cd0e1e6baf52

[31] https://cryptodaily.co.uk/2018/05/zilliqa-zil-answer-blockchain-scalability/

[32] https://commodity.com/cryptocurrency/what-are-forks/

[33] https://jaxenter.com/blockcahin-big-data-146218.html

[34] http://improving-bpm-systems.blogspot.com/2016/01/entarch-view-on-blockchain.html

[35] https://gendal.me/2015/02/10/a-simple-model-for-smart-contracts/

[36] https://github.com/capiman/sha256-sat-bitcoin

[37] http://learnmeabitcoin.com/faq/segregated-witness

[38] https://medium.com/ethereum-dapp-builder/how-blockchain-will-improve-big-data-da1547dd268

[39] https://people.xiph.org/ greg/attack-success.html

[40] https://techcrunch.com/2016/10/08/how-Blockchain-can-change-the-music-industry/

[41] https://techcrunch.com/2016/11/24/Blockchain-has-the-potential-to-revolutionize-the-supply-chain/

[42] http://www.altcointoday.com/bitcoin-ethereum-vs-visa-paypal-transactions-per-second/

[43] https://www.bbvaopenmind.com/en/a-secure-model-ofiotwithblockchain/?utm_source=views &utm_medium=article06& utm_campaign=MITcompany&utm_content=banafajan07

[44] https://www.buybitcoinworldwide.com/confirmations/

[45] https://www.coinbureau.com/review/zilliqa-zil/

[46] https://www.datanami.com/2018/06/28/blockchain-solutions-usher-in-era-of-trusted-big-data/

[47] https://www.draglet.com/Blockchain-applications/smart-contracts/use-cases

[48] https://www.ey.com/Publication/vwLUAssets/EYblockhain-in-insurance/$FILE/EY-blockhain-in-insurance.pd

[49] https://www.forbes.com/sites/bernardmarr/2018/02/07/5- blockchain-opportunities-no-company-can-afford-to-miss/ 9893de51a83d

[50] http://www.iamwire.com/wp-content/uploads/2016/12/how-blockchain-works.png

[51] https://www.ibm.com/blockchain/industries/supply-chain

[52] https://www.inc.com/nicolas-cole/web-analytics-willget-a-massive-improvement-with-blockchain-technology.html

[53] http://www.loc.gov/law/help/bitcoin-survey/

[54] https://www.kdnuggets.com/2017/09/introduction-blockchain-big-data.html

[55] https://www.reuters.com/article/us-jpmorgan-cyber-bitcoin-idUSKCN11P2DE

[56] Kim Y., Raman R. K., Kim Y.S., Varshney L. R., Shanbhag N. R., *Efficient local secret sharing for distributed Blockchain systems*, IEEE Communications Letters, vol. 23, no. 2, pp. 282–285, 2018.

[57] Knag, J., Stark, W., Hero, A.: *Turbo codes for fading and burst channels*, IEEE Theory Mini Conference, pp. 40–45 (1998)

[58] Koetter R.,MédardM.: *An Algebraic Approach to Network Coding*, IEEE/ACM Transactions on Networking, Vol. 11, No. 5, pp. 782–795, October 2003.

[59] Kozlenkova I.,Hult G.T.M., Lund D.J., Mena J.A., Kekec P.: *The role of marketing channels in supply chain managemen*, Journal of Retailing, 91 (4), 2015, pp 586–609.

[60] Krapež A.: *An Application of Quasigroups in Cryptology*, Math. Maced. Vol. 8 (2010), pp. 47-52.

[61] Labiod, H.: *Performance of Reed Solomon error-correcting codes on fading channels*, In: IEEE International Conference on Personal Wireless Communications (Cat. No.99TH8366), Jaipur, India, pp. 259-263 (1999)

[62] Laywine C. F., Mullen G. L.: *Discrete Mathematics using Latin Squares*, John Wiley & Sons, Inc., 1998.

[63] Li S.Y. R., Yeung R. W., Cai N., *Linear Network Coding*, IEEE Transactions on Information Theory, Vol. 49, No. 2, pp. 371–381, February 2003.

[64] Markovski S.: *Quasigroup string processing and applications in cryptography*, Proceedings of the 1stMII 2003 Conference, Thessaloniki, April 14-16,2003,pp. 278-290.

[65] Gligoroski D., Knapskog S. J., Andova S.:*Cryptcoding - Encryption and Error-Correction Coding in a Single Step*, The 2006 International Conference on Security and Management. Las Vegas, Nevada, USA: CSREA Press 2006.

[66] Markovski S., Gligoroski D., Andova S.: *Using quasigroups for one-one secure encoding*, Proc.VIII Conf.Logic and Computer Science"LIRA '97", Novi Sad, 1997, pp. 157-162.

[67] Markovski S., Gligoroski D., Bakeva V.: *Quasigrouop string processing: Part* 1, Contributions, Sec. Math. Tech. Sci., MANU, Vol. XX 1-2 (1999) pp. 13-28.

[68] Markovski S., Gligoroski D., Bakeva V.: *Quasigroup and Hash Function*, Discrete Mathematics and Applications: Proceedings of Sixth International Conference, South-West University, Blagoevgrad, Bulgaria (2001), pp. 43-50.

[69] Markovski S., Gligoroski D., Kocarev L.: *Unbiased Random Sequences from Quasigroup String Transformations*, Proceededings of Fast Software Encryption 2005, LNCS 3557, Springer-Verlag, 2005, pp. 163-180.

[70] Markovski S., Gligoroski D., Markovski J.: *Classification of quasigroups by random walk on torus*, Journal of applied mathematics and computing, Vol. 19, No. 1-2, September 2005, pp. 57-75.

[71] Markovski S., Kusakatov V.: *Quasigroups string processing: Part 2*, Contributions, Sec. Math. Tech. Sci., MANU, XXI, 1-2, 2000, pp. 15-32.

[72] Markovski S., Kusakatov V.: *Quasigroup string processing: Part 3*, Contributions, Sec. Math. Tech. Sci., MANU, XXIII-XXIV, 1-2, 2002-2003, pp.7-27. 7

[73] Mathur C.N., Narayan K., Subbalakshmi K.P.: *High Diffusion Cipher: Encryption and Error Correction in a Single Cryptographic Primitive*, Applied Cryptography and Network Security Lecture Notes in Computer Science, Vol. 3989 (2006), pp. 309-324. 9

[74] Mechkaroska D., Dimitrova V., Popovska-Mitrovikj A.: *A survey on applications of Blockchain technology*, Proceedings of 15th International Conference on Informatics and Information Technologies, April 2018 Mavrovo, Macedoinia, pp. 66–69.

[75] Mechkaroska D., Dimitrova V., Popovska-Mitrovikj A.: *Analysis of the possibilities for improvement of Blockchain technology*, 2018 26th Telecommunications Forum (TELFOR), Bel-

grade, Serbia, Nov.2018, doi: 10.1109/TELFOR.2018.8612034 https://ieeexplore.ieee.org/document/8612034

[76] Mechkaroska D., Popovska-Mitrovikj A. and Bakeva V.: *Cryptcodes Based on Quasigroups in Gaussian channel*, Quasigroups and Related Systems 24 (2016) No.2, pp. 249-268.

[77] Mechkaroska D., Popovska-Mitrovikj A., Bakeva V.: *A filter for Images Decoded using Cryptocodes Based on Quasigroups*, Proceedings of the 14th International Conference on Informatics and Information, Mavrovo, april 2017

[78] Mechkaroska D., Popovska-Mitrovikj A., Bakeva Smiljkova V.: *Performances of Fast Algorithms for Random Codes Based on Quasigroups for Transmission of Audio Files in Gaussian Channel*, In: Kalajdziski, S., Ackovska, N. (eds): ICT Innovations 2018. Engineering and Life Sciences, Springer International Publishing, pp. 286–296.

[79] Mechkaroska D., Popovska-Mitrovikj A., Bakeva V.: *New Cryptcodes for Burst Channels*, In: Ćirić M., Droste M., Pin JÉ. (eds) Algebraic Informatics. CAI 2019. Lecture Notes in Computer Science, vol 11545. Springer, Cham, pp. 202–212.

[80] Mechkaroska D., Popovska-Mitrovikj A., Bakeva V.: *Cryptcoding of Images for Transmission Trough a Burst Channels*, Journal of Engineering Science and Technology Review (JESTR), pp 65—69, ISSN: 1791-2377, Proceedings of Fourth International Scientific Conference "Telecommunications, Informatics, Energy and Management", September 2019, Kavala, Greece

[81] Mechkaroska D., Popovska-Mitrovikj A., Bakeva V., Dimitrova V.: *Network Coding based on quasigroup*, Proceedings of the 10th International Conference on Information Society and Technology, March 2020, Kopaonik, Serbia, pp. 240-243

[82] Mechkaroska D., Popovska-Mitrovikj A., Dimitrova V.: *Secure Big Data and IoT with implementation of Blockchain*, International Scientific Journal Security & Future, vol. 2(4), 2018, pp. 183 – 185 ISSN (PRINT) 2535-0668, ISSN (Online) 2535-082X International Scientific Conference on Security „CONFSEC 2018", December 2018, Bulgaria

[83] Nakamoto S., *Bitcoin: A peer-to-peer electronic cash system*, 2008, http://bitcoin.org/bitcoin.pdf.

[84] Nomura Research Institute, *Survey on Blockchain technologies and related services*, March 2016.

[85] Popovska-Mitrovikj A., Bakeva V., Markovski S.: *On random error correcting codes based on quasigroups*, Quasigroups and Related Systems Vol. 19, (2011), pp. 301-316.

[86] Popovska-Mitrovikj A., Bakeva V., Mechkaroska D.: *New Decoding Algorithm for Cryptcodes Based on Quasigroups for Transmission Through a Low Noise Channel*, In: Trajanov, D., Bakeva, V. (eds.): Communications in Computer and Information Science Series (CCIS), Vol.778, ICT-Innovations 2017, Springer, pp. 196–204.

[87] Popovska-Mitrovikj A., Bakeva V., Mechkaroska D.: *Fast Decoding with Cryptcodes for Burst Errors*, In: Dimitrova V., Dimitrovski I. (eds.): Communications in Computer and Information Science Series (CCIS), ISTInnovations (accepted).

[88] Popovska-Mitrovikj A., Markovski S., Bakeva V.: *Performances of error-correcting codes based on quasigroups*, D.Davcev, J.M.Gomez (Eds.): ICT-Innovations 2009, Springer (2009), pp. 377-389. 15

[89] Popovska-Mitrovikj A., Markovski S., Bakeva V.: *Increasing the decoding speed of random codes based on quasigroups*, S. Markovski, M. Gusev (Eds.): ICT Innovations 2012, Web proceedings, ISSN 1857-7288, pp. 93-102.

[90] Popovska-Mitrovikj A., Markovski S., Bakeva V.: *On improving the decoding of random codes based on quasigroups*, Proceedings of the 9th Conference on Informatics and Information Technology with International Particpants, Faculty of Computer Science and Engineering, University "Ss.Cyril and Methodius" (Macedonia), Bitola, Macedonia, April 2012, pp. 214-217.

[91] Popovska-Mitrovikj A., Markovski S., Bakeva V.: *Some new results for random codes based on quasigroups*, Proceedings of the 10th Conference on Informatics and Information Technology with International Participants, Faculty of Computer Science and Engineering, University "Ss.Cyril and Methodius" (Macedonia), Bitola, Macedonia, April 2013

[92] Popovska-Mitrovikj A., Markovski S., Bakeva V.: *4-Sets-Cut-Decoding algorithms for random codes based on quasigroups*, International Journal of Electronics and Communications (AEU), Elsevier, Vol.69, Issue 10, 2015, pp. 1417–1428.

[93] Popovska-Mitrovikj A., Mechkaroska D., Dimitrova V., Bakeva V.: *Algorithm for Reducing Storage in Blockchain based on Secret Sharing*, 2nd International Conference on Electrical, Communication and Computer Engineering (ICECCE 2020), Istanbul, Turkey, 12-13 June 2020, pp. 567–569

[94] Raman R.K., Varshney L.R.: *Distributed storage meets secret sharing on the blockchain*, in Proceedings of Information Theory and Applications Workshop (ITA), San Diego, CA, USA, February 2018.

[95] Raman R.K., Varshney L.R. :*Dynamic distributed storage for blockchains*, in Proc. IEEE Int. Symp. Inf. Theory (ISIT), Jun. 2018, pp. 2619–2623. [Online]. Available: https://arxiv.org/pdf/1711.07617.pdf

[96] Rao T. R. N, Nam K.H.: *Private-Key Algebraic-Code Encryptions*, IEEE Transaction on information theory, Vol. 35, No 4, (1989) pp. 829- 833.

[97] Rosenfeld M.: *Analysis of hashrate-based double-spending*, arXiv:1402.2009.

[98] Sankar K.: *Bit Error Rate (BER) for BPSK modulation*, August 2007 http://www.dsplog.com/2007/08/05/bit-error-probability-for-bpsk-modulation/

[99] Satyavolu P., Sangamnerkar A.: *Blockchain's smart contracts: Driving the next wave of innovation across manufacturing value chains*. https://www.cognizant.com/whitepapers/BlockChains-smart-contracts-driving-the-next-wave-of-innovation-across-manufacturing-value-chains-codex2113.pdf

[100] Shamir: *How to share a secret*, Communications of the ACM, vol. 22, no. 11, pp. 612-613, 1979.

[101] The Zilliqa team: *The Zilliqa Techical Whitepaper*, https://docs.zilliqa.com/whitepaper.pdf, August 10, 2017.

[102] Tzonelih H., Rao T.R.N.: *Secret error-correcting codes*, Goldwasser, R. (Ed.): Advances in Cryptology - CRYPTO ‘88, LNCS 403, (1990), pp.540-563.

[103] Zivic N., Ruland C.: *Parallel Joint Channel Coding and Cryptography*, Inter. Jour. of Electrical and Electronics Rngineering 4:2 (2010), pp. 140-144.

[104] Yang M., Yang Y., Fellow: *Applying Network Coding to Peer-to-Peer File Sharing*, IEEE transactions on computers, Vol. 63, No. 8, August 2014

[105] Zhang P., Lin C., Senior Member, IEEE, Jiang Y., Fan Y., Xuemin (Sherman) Shen: *A Lightweight Encryption Scheme for Network-Coded Mobile Ad Hoc Networks*, IEEE Transactions on Parallel and Distributed Systems 25 (9), pp. 2211-2221.

[106] Мечкароска Д.: *Перформанси на Случајните кодови базирани на квазигрупи при пренос низ Гаусов канал*, магистерски труд, Универзитет ”Св. Кирил и Методиј”, Скопје, 2014.

[107] Поповска-Митровиќ А.: *Прилог кон примена на квазигрупите во теоријата на кодирање и криптографијата*, докторска дисертација, Универзитет ”Св. Кирил и Методиј”, Скопје, 2014