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

АЛЕКСАНДРА Б. ПОПОВСКА-МИТРОВИЌ

ПРИЛОГ КОН ПРИМЕНА НА КВАЗИГРУПИТЕ ВО ТЕОРИЈАТА НА КОДИРАЊЕ И КРИПТОГРАФИЈАТА

ДОКТОРСКА ДИСЕРТАЦИЈА

СКОПЈЕ, 2014

Ментор:	д-р Верица Бакева, редовен професор ФИНКИ при УКИМ - Скопје
Коментор:	д-р Смиле Марковски, редовен професор ФИНКИ при УКИМ - Скопје
Членови на комисијата:	Академик д-р Љупчо Коцарев, редовен професор ФИНКИ при УКИМ - Скопје
	д-р Верица Бакева, редовен професор ФИНКИ при УКИМ - Скопје
	д-р Смиле Марковски, редовен професор ФИНКИ при УКИМ - Скопје
	д-р Весна Димитрова, доцент ФИНКИ при УКИМ - Скопје
	д-р Магдалена Георгиева, редовен професор ПМФ при УКИМ - Скопје (во пензија)
Датум на одбрана:	_____
Датум на промоција:	_____
Научна област:	Теорија на кодирање и криптографија

Посветено на моето сонце Лука

Благодарност

На патот кон успешна кариера, на секој научен работник му се потребни ментори кои ќе го насочат во вистинскиот правец. На мојот пат тоа беа проф. д-р Верица Бакева и проф.д-р Смиле Марковски.

Затоа, би сакала да искажам голема благодарност на мојот ментор проф. д-р Верица Бакева за огромната поддршка и помош во текот на моите истражувања и изработката на докторската дисертација. Во текот на целата моја работа не само како научен истражувач, туку и како асистент несебично го делеше со мене сето нејзино искуство и знаење и секогаш наоѓаше време да ми помогне и да ме насочи во вистинската насока.

Исто така, сакам да му се заблагодарам и на мојот кометор проф. д-р Смиле Марковски за неговата подршка, за сите негови забелешки и нови идеи во текот на моите истражувања.

На секој научен работник, би му посакала ментор и коментор какви што имав јас и се надевам дека мојата соработка со нив ќе продолжи и во иднина.

Воедно се заблагодарувам на проф. д-р Магдалена Георгиева, која ме повика да бидам нејзин асистент на курсот по Теорија на информации, со што ме инспирираше моите научни истражувања да ги насочам токму во оваа област.

Во изработката на секоја докторска дистертација, забелешките се секогаш добредојдени. Затоа се заблагодарувам на доц. д-р Весна Димитрова, за сите забелешки и за целосната подршка при изработката на оваа дисератција. Ми беше особена чест и задоволство што во дел од моите истражувања соработував со неа.

Исто така, сакам да му се заблагодарам на академик д-р Љупчо Коцарев што прифати да биде дел од комисијата за оценка на мојата докторка дисертација.

Но освен поддршката и помошта што ја добив од моите професионални соработници, од голема помош ми беше поддршката од моите најблиски. Затоа огромна благодарност сакам да искажам на моите родители кои секогаш

ме подржуваа, несебично ми помагаа и веруваа во мене. Посебно се благодарувам на мојата мајка Славица, која во целиот период додека јас работев на истражувањата и изработката на овој докторат ме заменуваше во мојата најважна улога - мајчинството, несебично се грижеше за мојот син и се трудеше да му го надополни моето отсуство. Но, во текот на моите истражувања и изработка на овој труд, ми недостигаше присуството и подршката на мојот татко Бранко, чија поддршка ја имав на почетоците на мојата кариера, и знам дека сега тој ќе беше многу горд и на овој мој успех. Се заблагодарувам на мојот сопруг Валентино за љубовта и подршката која ми ја даде во целиот овој период, која ми беше од огромно значење за да продолжам и истраам во работата. Исто така им благодарам и на мојот брат и сите мои роднини кои веруваа во мене и секогаш кога ми требаше помош несебично ми ја даваа. Најголема благодарност, а воедно и извинување сакам да му искажам на мојот син Лука. Неговите прегратки и неизмерна љубов беа мојата најголема подршка и извор на енергија во текот на мојата работа. Од друга страна, знам дека поради мојата зафатеност со изработката на докторската дисертација понекогаш му недостигаше моето внимание и нашите заеднички дружби. Токму затоа ова дисертација ја посветувам на моето сонце Лука.

Воедно сакам да им се заблагодарам на сите мои пријатели и колегите од ФИНКИ за подршката при изработката на овој труд, посебно на мојот другар Александар за неговата техничка помош и на колешката Даниела за нашата заедничка соработка која се надевам дека ќе продолжи и понатаму.

Александра Поповска-Митровиќ

Резиме

Во оваа докторска дисертација се истражувани некои примени на квазигрупите во теоријата на кодирање и криптографијата. Разгледани се случајните кодови базирани на квазигрупи (Random Codes Based on Quasigroups - RCBQ), предложени од Данило Глигороски, Смиле Марковски и Љупчо Коцарев. Овие кодови се комбинација на криптографски алгоритми и кодови за поправање на грешки и зависат од неколку параметри. Испитано е влијанието на параметрите на кодот и должината на пораките врз перформансите на кодот. Од експериментите е заклучено дека брзината на процесот на декодирање е еден од најголемите проблеми за овие кодови.

Со цел да се подобри брзината на декодирање дефиниран е нов алгоритам за кодирање/декодирање, наречен алгоритам-за-декодирање-со-пресек (Cut-Decoding алгоритам). Модифицираниот процес на декодирање е 4.5 пати побрз од оригиналниот за код (72, 288), за квазигрупи од ред 16. Исто така, се предложени неколку методи за намалување на неуспешните декодирања. На овој начин се добиени подобри резултати за веројатностите за пакет-грешка и бит-грешка. Во оваа дисератција се испитани перформансите на случајните кодови базирани на квазигрупи кога во процесите на кодирање и декодирање се користат квазигрупи од ред 4 и ред 256. Разгледана е и примената на овие кодови за декодирање на слики кои се пренесуваат низ бинарен симетричен канал и споредени се резултатите добиени со користење на стандардниот алгоритам, алгоритамот-за-декодирање-со-пресек и Рид-Соломон кодовите.

За добивање на уште побрз процес на декодирање дефиниран е уште еден алгоритам за кодирање/декодирање, наречен алгоритам-за-декодирање-со-4пресеци (4-Sets-Cut-Decoding алгоритам). Исто така, за подобрување на веројатностите за пакет-грешка и бит-грешка, дефинирани се неколку методи за генерирање на редуцираните множества со кандидати за декодираната порака. Анализирани се перформансите на различните алгоритми за декодирање на RCBQ (стандардниот, алгоритам-за-декодирање-со-пресек и алгоритмите-за-декодирање-со-4пресеци) за код со рата $R = 1/8$ и разгледана

е примената на методите за намалување на бројот на неуспешни декодирања во предложените нови алгоритми. Испитани се перформансите на RCBQ со третата верзија на алгоритмот-за-декодирање-со-4пресеци за кодирање/декодирање на слики кои се пренесуваат низ бинарен симетричен канал и направена е споредба со Рид-Соломон кодовите.

Изведена е теориска горна граница за веројатноста за пакет-грешка добиена со користење на новите алгоритми и приближни формули за кардиналниот број на редуцираните множества со кандидати за декодирање. Со изведените формули е докажано дека со новите алгоритми подобрени се перформансите на овие кодови.

Почетната идеја за квазигрупна стринг трансформација базирана на парастрофите е дадена од Александар Крапеж. Во оваа теза, предложена е модификација на оваа трансформација, наречена парастрофна квазигрупна (PE) трансформација и разгледани се нејзините криптографски својства. Со користење на оваа трансформација направена е класификација на квазигрупите од ред 4 во три класи: 1) парастрофно-фрактални; 2) фрактални парастрофно-нефрактални; и 3) нефрактални квазигрупи. Испитани се алгебарските својства на парастрофно-фракталните квазигрупи од ред 4 и со користење на некои индентитети даден е математички модел на парастрофната фракталност. Исто така, за секоја квазигрупа од ред 4 пресметан е бројот на различни парастрофи и множеството од сите квазигрупи од ред 4 е поделено на четири класи. Со користење на оваа трансформација зголемен е бројот на квазигрупи од ред 4 кои се погодни за конструкција на криптографски примитиви. Докажано е важно криптографско својство на PE-трансформацијата. Имено, ако PE-трансформацијата се користи како функција за криптирање тогаш после n нејзини примени на произволна порака, распределбата на l-торките $(l = 1, 2, ..., n)$ е рамномерна. Ова својство обезбедува отпорност на статистички напади.

КЛУЧНИ ЗБОРОВИ: квазигрупа, квазигрупна трансформација, кодови кои поправаат грешки, случајни кодови, криптокодирање, веројатност за пакет-грешка, веројатност за бит-грешка, брзина на декодирање, криптографски својства, рамномерност, статистички напад.

Abstract

In this thesis we research some applications of quasigroups in the coding theory and the cryptography. We consider Random Codes Based on Quasigroups (RCBQ) proposed by Danilo Gligoroski, Smile Markovski i Ljupco Kocarev. These codes are a combination of cryptographic algorithms and error-correcting codes and they have several parameters. We investigate the influence of the code parameters and the length of the messages on the code performance. From the experiments we conclude that the speed of the decoding process is one of the biggest problem for these codes.

In order to improve the decoding speed, we define a new coding/decoding algorithm called Cut-Decoding algorithm. The modified decoding process is 4.5 times faster than the original one for code (72, 288), for quasigroups of order 16. Also, we propose several methods for reducing the unsuccessful decodings. In such a way we obtain better values for packet-error and bit-error probabilities. In this thesis we investigate the performances of the random codes based on quasigroups when quasigroups of order 4 and order 256 are used in the coding/decoding processes. We consider an application of these codes for decoding images transmitted through a binary symmetric channel and compare the results obtained using the standard algorithm, Cut-Decoding algorithm and Reed-Solomon codes.

For obtaining a faster decoding process we define another coding/decoding algorithm, called 4-Sets-Cut-Decoding algorithm. Also, for improving the packet-error and bit-error probabilities we define several methods for generating reduced decoding candidate sets. We analyze the performances of different decoding algorithms of RCBQ (the standard, Cut-Decoding and 4-Sets-Cut-Decoding algorithms) for a code with rate $R = 1/8$ and we consider the application of the methods for reducing the number of unsuccessful decodings in the new proposed algorithms. We investigate performances of RCBQ with 4-Sets-Cut-Decoding algorithm#3 for coding/decoding images transmitted through a binary symmetric channel and we compare these results with suitable results obtained with Reed-Solomon codes.

We derive a theoretical upper bound for the packet-error probability obtained by the new algorithms and approximate formulas for cardinality of the reduced decoding candidate sets. With the derived formulas we prove that with the new algorithms we have improved the performances of these codes.

The first idea for quasigroup string transformation based on parastrophes is given by Aleksandar Krapež. In this thesis, we propose a modification of this transformation, called parastrophic quasigroup (PE) transformation and consider its cryptographic properties. Using this transformation we classify the quasigroups of order 4 into three classes: 1) parastrophic-fractal; 2) fractal parastrophic-non-fractal; and 3) non-fractal quasigroups. We investigate the algebraic properties of parastrophic fractal quasigroups of order 4 and give a mathematical model of parastrophic fractality using some identities. Also, we find a number of different parastrophes of each quasigroup of order 4 and divide the set of all quasigroups of order 4 in four classes. Using this transformation we increase the number of quasigroups of order 4 which are suitable to design cryptographic primitives. We prove an important cryptographic property of PE-transformation. Namely, if PE-transformation is used as encryption function then after n applications of it on arbitrary message the distribution of l-tuples $(l = 1, 2, ..., n)$ is uniform. This property implies the resistance to statistical kind of attack.

KEY WORDS: quasigroup, quasigroup transformations, error-correcting codes, random codes, cryptcoding, packet-error probability, bit-error probability, decoding speed, cryptographic properties, uniformity, statistical attack.

Содржина

Вовед

Брзиот развој на комуникацискатa технологија и потребата од сигурен пренос на податоци бара постојано подобрување на постоечките методи и алгоритми и развивање на нови кои ќе овозможат 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. Заради потребата да се добие во исто време и ефикасен и безбеден пренос на податоците, сè повеќе се развива и концептот на криптокодирање во кој процесите на кодирање и криптирање се спојуваат во еден процес. Постојат многу дизајни [27, 30, 32, 56, 77, 84, 85] во кои што се испреплетуваат 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 енкрипција.

Во поново време во голем дел од истражувањата од овие две области примена наоѓаат квазигрупите. Најновите истражувања покажуваат дека квазигрупите наоѓаат примена во изработката на основни алатки кои се користат во кодирањето и криптографијата. Квазигрупите како многу погодни алгебарски структури имаат широка примена заради нивната структура, својства и нивниот голем број и за нив постои обемна научна литература. Последниве неколку години значаен придонес за примената на квазигрупите во теоријата на кодирање и криптографијата имаат и истражувањата што ги вршат неколку истражувачи од Факултетот за информатички науки и компјутерско инженерство во Скопје. Главните истражувања 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.

Со помош на квазигрупите се дефинирани и проучувани голем број квазигрупни трансформации [40, 46, 47, 51, 52, 53, 54]. Истражувањата на својствата на низите добиени со квазигрупните трансформации покажаа дека овие трансформации може да најдат огромна примена во теоријата на ко-

дирање за дизајн на кодови кои откриваат и поправаат грешки и во криптографијата за конструкција на криптографски примитиви. Добиените резултати доведуваат до низа отворени прашања, чие што решавање би дало голем придонес во теоријата на кодирање и криптографијата. Квазигрупните трансформации се пресликувања од конечни низи над конечна азбука и покажуваат својства на дискретни динамички системи. Затоа, како релативно нова област претставуваат предизвик за нивно понатамошно истражување и нивна примена во овие области.

Во мојот магистерски труд ([89]) беше направено експериментално испитување на својствата на случајните кодови кои поправаат грешки базирани на квазигрупи, предложени од Данило Глигоровски, Смиле Марковски и Љупчо Коцарев [30]. Од експериментите направени со предложените случајни кодови базирани на квазигрупи, произлегоа неколку идеи за модификации на овие кодови со цел да се добие поефикасен процес на декодирање. Добиените резултати покажаа дека перфомансите на овие кодови многу зависат од изборот на квазигрупата и другите параметри во нивниот дизајн. Од експерименталната споредба на перформансите на овие кодови со кодови познати по својата практична примена за складирање и пренос на податоци (како што се кодовите на Рид-Милер и Рид-Соломон), произлезе заклучокот дека и кодовите базирани на квазигрупи можат да најдат широка практична примена. Освен добрите перформанси, овие кодови имаат и криптографски својства кои обезбедуваат и безбеден пренос на податоци. Но, отворено остана прашањето за забрзување на процесот на декодирање на овие случајни кодови, што се покажа како голема слабост при изведувањето на експериментите.

Вообичаениот начин да се добие код кој поправа грешки и кој ќе биде отпорен на напади од трети лица се состои во примена на некој познат шифрувач на кодните зборови кодирани со познат код за поправање на грешки, пред тие да бидат пренесени низ несигурен канал со пречки ([84, 85]). Во тој случај се користат два алгоритми, еден за кодирање (со алгоритам на код кој поправа грешки настанати при преносот) и друг алгоритам за криптирање.

Во [58], авторот ја користи способноста на Goppa кодовите за корекција на грешки за да обезбеди сигурност на податоците. Неговата идеја е воведување на случаен вектор на грешки на секоја кодирана оригинална порака пред преносот. Ако тежината на векторот со грешки е мала, овој систем станува помалку безбеден, но ако се земе поголема тежина тогаш се намалува способноста за корекција на грешките. Затоа во овој систем треба да се најде содветен баланс помеѓу безбедноста и сигурноста на податоците.

Во [84], авторите предлагаат нова шема во која се комбинира криптирање и кодирање на податоците. Во неа се задржува целата способност на користениот код за корекција на грешките во каналот. Во оваа шема во процесот на енкрипција на пораката се применува прво инверзибилна функција, потоа таа порака се кодира (со користење на нелинеарен код) и на крајот се користи и случајна матрица за пермутација. Исто така, во истиот труд авторите предлагат и друга ваква шема која е базирана на техниката на верижно поврзување на блокови (block chaining) и испитуваат неколку криптоаналитички напади на предложените шеми.

Друга шема во која се комбинирани алгоритми за кодирање и криптирање е предложена во [85]. Во овој алгоритам, најпрво пораката се дели на два дела и секој дел од пораката се проширува со редундантна криптографска вредност за проверка (со должина n) со користење на криптографска функција за проверка RCF (генерирање на дигитален потпис, MAC/H-MAC). Потоа, се прави мешање на двете редундантни потпораки во една порака која се кодира со користење на конволуциски или турбо код.

Со цел да добијат поефикасен дизајн, во трудот [56] авторите дефинираат еден алгоритам во кој се комбинирани блок шифрувач и код за поправање на грешки, наречен High Diffusion (HD) шифрувач. Овој блок шифрувач е составен од неколку итерации на трансформација на рунда и операција за мешање на клучот. Трансформацијата на рунда се состои од три нивоа: линеарна замена (linear substitution), транспозиција на симболи (symbol transposition) и High Diffusion ниво на кодирање. Способноста за корекција на грешки на HD шифрувачот се должи на употребата на една нова класа на кодови, наречени High Diffusion кодови (предложени од истите автори) во дифузиското ниво на шифрувачот. Главната примена на нивниот дизајн е за криптографски цели, иако тој може да биде користен и како код за корекција на грешки.

Случајните кодови базирани на квазигрупи (Random Codes Based on Quasigroups - RCBQ) кои ги разгледувам во моите истражување, за првпат се дефинирани во [30]. Овие случајни кодови кои поправаат грешки се дефинирани со користење на криптографски алгоритми за време на самиот процес на кодирање и декодирање. Затоа, тие овозможуваат не само корекција на одреден број на грешки во пратените податоци, туку исто така овозможуваат и безбедност на податоците, и сето тоа е вградено во еден алгоритам. Ако информацијата е кодирана со користење на овие кодови, тогаш примателот може да ја декодира и да ја добие оригиналната информација само ако тој/таа знае точно кои параметри биле користени во процесот на кодирање, дури и

ако комуникацискиот канал е без пречки. Во оваа докторска дисертација ќе бидат разгледувани перформанските на RCBQ, како кодови за поправање на грешки и затоа тука нема да бидат анализирани нивните криптографски својства.

Во дизајнот на RCBQ се користат алгоритмите за криптирање/декриптирање од имплементацијата на тотално асинхронизиран проточен шифрувач (Totally Asynchronous Stream Cipher - TASC) со користење на квазигрупни стринг трансформации [27]. Овие криптографски алгоритми користат азбука Q и квазигрупна операција $*$ над Q заедно со нејзината парастрофа $\backslash$. Ние ќе ги опишеме овие кодови со користење на квазигрупи, но од дефиницијата на алгоритмите е јасно дека во нивниот дизајн може да бидат користени и други алгоритми за криптирање и декриптирање. Авторите во дизајнот на RCBQ ја користат азбуката од нибли $Q = \{0, 1, \ldots, 9, a, b, c, d, e, f\}$, но тука алгоритмите за кодирање и декодирање ќе бидат дефинирани во поопшт случај со користење на симболи од a бита, наместо нибли.

Во мојата магистерски работа ([89]) и трудот ([68]) го испитавме влијанието на избраните параметри на перформансите на овие кодови и покажавме дека секој параметар во конструкцијата на овие кодови има големо влијание на перформансите на кодот, т.е., параметрите се вземно зависни. Исто така, предложивме метод за намалување на бројот на неуспешни декодирања со кој се постигнува подобрување на веројатноста за точно декодирање без значително намалување на брзината на декодирање. На почетокот од оваа теза накратко ќе дадеме дел од тие резултати добиени при истражувањата за мојата магистерска теза. Потоа, ќе го испитаме и влијанието на должината на пораките на перформансите на кодот.

Од направените експерименти со RCBQ заклучивме дека брзината на декодирање е еден од најголемите проблеми кај овие кодови. Зависно од избраниот патерн за додавање на редундантните нулти симболи, процесот на декодирање е бавен во некои итерации, бидејќи бројот на елементите во множествата со кандидати за декодирање е многу голем. Ако редундантните нули се распоредат порамномерно, тогаш бројот на елементи во овие множествата нема да биде многу голем, но ќе се појават многу неуспешни декодирања со повеќе кандидати за декодираната порака. Затоа во патерните треба да има повеќе нулти симболи на крајот. Всушност, многу е тешко да се најде добра рамнотежа за поставување на редундантните нули, но експериментално имаме најдено неколку задоволителни патерни.

Со цел да ја подобриме брзината на декодирање, во оваа теза ќе дефинираме нов алгоритам за кодирање/декодирање наречен алгоритам-за-декоди-

рање-со-пресек (Cut-Decoding алгоритам). 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 новиот алгоритам-за-декодирање-со-пресеци (или АДП алгоритам) кој го предлагаме, направени се модификации со цел да се намали бројот на кандидати за декодирање во сите итерации од процесот на декодирање. Во овој алгоритам, се користат две трансформации на редундантната порака со користење на различни параметри, а кандидатите за декодирана порака се добиваат со барање пресек на соодветните множества со кандидати добиени во двата паралелни процеси на декодирање. На овој начин процесот на декодирање за код (72,288) е 4.5 пати побрз од оригиналниот алгоритам. Исто така, ќе бидат разгледани и неколку методи (модификации на алгоритмот за декодирање) за намалување на неуспешните декодирање со двата типа на грешки кои се појавуваат во процесот на декодирање. Со предложените методи се постигнува намалување на веројатноста за пакет-грешка и веројатноста за бит-грешка за сите вредности на веројатноста за бит-грешка во бинарен симетричен канал.

Во дизајнот и досегашните истражувања со RCBQ користена е само азбуката од нибли и квазигрупи од ред 16. Тука ќе бидат испитани перформансите на овие кодови кога во процесите на кодирање и декодирање се користат и квазигрупи од ред 4 или ред 256. Тогаш пораките и кодните зборови се стрингови од 2-битни симболи или 8-битни симболи (бајти), соодветно. За таа цел презентирани се резултатите од неколку експерименти со стандардниот метод на кодирање/декодирање и со АДП алгоритамот кој го предлагаме. Од резултатите може да се заклучи дека и во овој случај подобри резултати се добиваат со новиот АДП алгоритам, како за веројатностите за грешка, така и за брзината на декодирање.

Исто така, испитани се перформансите на случајните кодови базирани на квазигрупи за пренесување на слики низ бинарен симетричен канал. За таа цел направени се неколку модификации со алгоритмот за декодирање за разрешување на случаите на делумно и неединствено декодирање кои се јавуваат при двата вида на неуспешно декодирање. Направени се експерименти со неколку различни (во боја и црно-бели) слики со користење на стандардниот алгоритам за кодирање/декодирање и АДП алгоритамот. За споредба направени се експерименти и со Рид-Соломон кодовите. Од добиените резултати заклучуваме дека со користење на новиот АДП алгоритам се добиваат појасни декодирани слики отколку со другите разгледувани алгоритми.

Поради значителното подобрување на брзината на декодирање добиено

со АДП алгоритамот произлезе идеја за користење на пресеци од повеќе множества со кандидати за декодирање, со цел да се забрза процесот на декодирање уште повеќе. Во вториот алгоритам кој е предложен во оваа теза направени се модификации на АДП алгоритамот при што се користат четири трансформации на редундантната порака. Со новиот алгоритам, наречен алгоритам-за-декодирање-со-4пресеци (4-Sets-Cut -Decoding или АД4П алгоритам) се добива поголемо забрзување на декодирање. Со цел да ги подобриме и веројатностите за пакет-грешка и бит-грешка дефинираме неколку различни верзии на процедурата за генерирање на множества со кандидати за декодирање (т.е., АД4П#1, АД4П#2, АД4П#3 и АД4П#4 алгоритми). Анализирани се перформансите на различни алгоритми за декодирање на RCBQ (стандардниот, АДП и АД4П алгоритмите) за код со рата $R = 1/8$. Од резултатите може да се заклучи дека за код (72,576) со новите алгоритми, АДП и АД4П алгоритми се добива големо подобрување на брзината на декодирање и многу подобри резултати за веројатностите за пакет-грешка и бит-грешка. Имено, за овој код, АДП алгоритамот е 5.2 пати побрз од стандардниот, за одредени вредности на веројатноста p за бит-грешка во бинарен симетричен канал. Од друга страна, со АД4П алгоритмите декодирањето е 6.3 пати побрзо отколку со стандардниот алгоритам, а за некои вредности на p се добиваат и 24 пати помали вредности за веројатноста за пакет-грешка. Исто така, разгледана е примената на методите за намалување на неуспешните декодирања во новите предложени алгоритми. Со тоа се добиваат уште подобри резултати за веројатноста за точно декодирање, без значително намалување на брзината на декодирање. Испитани се и перформансите на RCBQ со третата верзија на алгоритмот-за-декодирање-со-4пресеци за кодирање/декодирање на слики кои се пренесуваат низ бинарен симетричен канал и направена е споредба со Рид-Соломон кодовите. Од споредбата заклучивме дека за сите разгледани веројатности p за бит-грешка во бинарен симетричен канал RCBQ со новиот алгоритам дава појасни слики отколку Рид-Соломон кодовите.

Освен експерименталните резултати за новите алгоритми кои се предложени во оваа теза, дадени се и некои теориски резултати со кои се потврдува подобрувањето на перформансите на случајните кодови базирани на квазигрупи. Изведена е формула за горната граница за веројатноста за пакет-грешка кај новите алгоритми. Притоа, покажано е дека оваа горна граница е еднаква на теориската веројатност за пакет-грешка за стандардниот алгоритам. Оваа теориски изведена горна граница е и експериментално потврдена. Исто така, определени се приближни формули за кардиналниот

број на множествата со кандидати за декодирање после нивното редуцирање предложено во новите алгоритми. Со тоа е покажано дека со оваа редукција се добива значително намалување на бројот на елементи во множествата, од што најмногу зависи брзината на декодирање. Овие теориски резултати може да се искористат и како математички метод за избор на добри параметри во дизајнот на кодовите.

Како што е претходно спомнато, квазигрупните трансформации се покажаа како погодни за конструкција и на криптографски примитиви. Во [46] дефинирани се квазигрупните трансформации E и D. Потоа, во [47] авторите покажуваат дека E-трансформацијата може да се примени за дефинирање на функција за криптирање, која ќе биде отпорна на напади со груба сила, поради големиот број на квазигрупи. Исто така, докажуваат дека со примена на n квазигрупни E-трансформации над произволна низа, m-торките во процесираната низа се рамномерно распределени за $m \leqslant n$. Тоа значи дека при доволно големо n се обезбедува отпорност од статистички видови на напади. Истовремено, ваквите трансформации даваат можност за класифицирање на квазигрупите според погодноста за користење во криптографијата и теоријата на кодирање. Во трудовите [4, 17, 45, 88] дадени се повеќе такви класификации на квазигрупите од ред 4. Користејќи ги парастрофите на квазигрупите, Крапеж ([35]) дава идеја за нова квазигрупна стринг трансформација која исто така може да биде применета во криптографија.

Во оваа докторска теза дефинирана е нова квазигрупна трансформација, наречена парастрофна квазигрупна трансформација или PE-трансформација. Со оваа трансформација се подобруваат својствата на квазигрупно процесираните низи и се зголемува бројот на квазигрупи погодни за конструкција на криптографски примитиви. Во оваа теза дадени се две нови класификации на квазигрупите од ред 4: класификација според бројот на различни парастрофи и класификација според парастрофна фракталност. Со определување на бројот на различни парастрофи за секоја квазигрупа од ред 4, множеството од сите квазигрупи од ред 4 го поделивме во четири класи. Покажано е дека парастрофите на фракталните квазигрупи од ред 4 се исто така фрактални и сите нефрактални квазигрупи од ред 4 имаат нефрактални парастрофи.

Во трудот [17], со користење на графичко претставување на квазигрупно процесираните низи, даваат поделба на квазигрупите од ред 4 на две класи: фрактални и нефрактални квазигрупи. Се покажува дека фракталните квазигрупи се непогодни за конструкција на криптографски примитиви. Со користење на новата парастрофна PE-квазигрупна трансформација, класата на фрактални квазигрупи е поделена на две подкласи: парастрофно фрак-

тални и парастрофно нефрактални квазигрупи. Имено, направени се експерименти со графичко претставување на процесирани низи со сите 576 квазигрупи од ред 4 и добиена е нова класификација на квазигрупите од ред 4 во три класи: 1) класа на парастрофно фрактални квазигрупи (88 квазигрупи); 2) класа на фрактални парастрофно-нефрактални квазигрупи (104 квазигрупи) и 3) класа на нефрактални квазигрупи (384 квазигрупи). Со оваа класификација зголемен е бројот на квазигрупи погодни за примена во криптографија. Имено, за криптографски цели може да се користат не само квазигрупите од класата 3, туку и квазигрупите од класата 2. Понатаму, во зависност од бројот на различни парастрофи, определен е бројот на елементи во потткласите од парастрофно-фрактални и фрактални парастрофно-нефрактални квазигрупи.

Во [45], авторите даваат математички модел на фракталноста на квазигрупите со користење на идентитети. Во оваа докторска теза определен е сличен модел, но сега за парастрофната фракталност на квазигрупите. За таа цел ги испитавме алгебарските својства на парастрофно фракталните квазигрупи од ред 4. За да најдеме соодветни идентитети со кои ќе се издвојат парастрофно-фракталните квазигрупи, разгледавме многу идентитети, а посебно симетричните. Со користење на определениот математички модел за парастрофната фракталност и без користење на графичко претставување може да се провери дали дадена квазигрупа е парастрофно-фрактална.

Во оваа теза е докажано дека ако PE-трансформацијата се користи за процесирање на низи, тогаш после n последователни примени над произволна низа, исто како и кај E-трансфомацијата, распределбата на m-торките (за $m = 1, 2, \ldots, n$) е рамномерна. Како што веќе напоменавме ова е многу важно криптографско својство за една трансформација кое имплицира отпорност од статистички напади. За илустрација на теориските резултати, презентирани се и неколку експериментално добиени резултати.

Во [2], авторите докажуваат дека веројатностите на $(n + 1)$-торките во низата добиена со n примени на E-трансформацијата се поделени во класи. Користејќи ги овие резултати, авторите предлагаат и алгоритам за криптоанализа. Тука, експериментално ќе биде покажано дека доколку наместо E се користи PE-трансформацијата тогаш не можат да се одвојат вакви класи на веројатностите. Ова значи дека алгоритмот за криптоанализа предложен во [2] не може да се примени кога влезната порака е криптирана со користење на PE-трансформацијата. Оттука, може да заклучиме дека пораката криптирана со PE-трансформација е поотпорна на статистички напади.

Од сите изнесени резултати за предложената парастрофна квазигрупна

трансформацијата може да се заклучи дека оваа нова трансформација е подобра функција за криптирање од претходно дефинираната квазигрупна E-трансформацијата.

Резултатите од истражувањата дадени во оваа докторска дисертација се презентирани на неколку домашни и меѓународни конференции и семинари. Дел од трудовите се веќе објавени или пратени за објавување во меѓународни и домашни часописи и зборници од конференции. Во продолжение е даден списокот на овие трудови:

1. 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 [68]

2. 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 [67]

3. Bakeva V., Dimitrova V., Popovska-Mitrovikj A.: *Parastrophic quasigroup string processing*, Proc. of the 8^{th} Conference on Informatics and Information Technology with International Participants, Macedonia (2011) pp. 19-21 [3]

4. 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. 221-230 [14]

5. 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 [69]

6. 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 Participants, Faculty of Computer Science and Engineering, University "Ss.Cyril and Methodius" (Macedonia), Bitola, Macedonia, April 2012, pp. 214-217 [70]

7. 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, pp. 178-181[71]

8. Popovska-Mitrovikj A., Markovski S., Bakeva V.: *Error-correcting codes with cryptographic algorithms*, Proceedings of 21st Telecommunications Forum (TELFOR), 2013, Belgrade, Serbia, pp. 327-330, $http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=\&arnumber=6716236\&queryText\%3DPopovska-Mitrovikj$ [72]

9. Popovska-Mitrovikj A., Mechkaroska D., Bakeva V.: *Applying error-correcting codes based on quasigroups for image coding*, Proceedings of the 11th International Conference on Informatics and Information Technology, Faculty of Computer Science and Engineering, University "Ss.Cyril and Methodius" (Macedonia), Bitola, Macedonia, April 2014, (in print) [74]

10. Bakeva V., Popovska-Mitrovikj A., Dimitrova V.: *Resistance to statistical attacks of Parastrophic Quasigroup Transformation*, submitted to Serdica Journal of Computing [5]

11. Popovska-Mitrovikj A., Markovski S., Bakeva V.: *4-Sets-Cut-Decoding algorithms for random codes based on quasigroups*, submitted to Advances in Mathematics of Communications (AMC) [73]

Истражувањата во насока на примена на квазигрупите како во областа на кодирањето, така и во областа на криптографијата, ветуваат многу. Се очекува резултатите од овие истражувања да имаат голема примена при складирање и коректен пренос на податоци и воопшто во зголемувањето на нивото на заштита на податоци и безбедна комуникација.

Преглед на содржината

Докторската дисертација е поделена на седум глави, заклучок и прилози.

Во првата глава дадени се основните дефиниции и својства на квазигрупите и квазигрупните трансформации на кои се темелат нашите истражувања за примена на квазигрупите во теоријата на кодирање и криптографијата. Резултатите добиени од нашите истражувања и новите резултати за примена

на квазигрупите и квазигрупните трансформации во овие научни области се претставени во следните глави.

Втората глава се однесува на случајните кодови базирани на квазигрупи, предложени од Данило Глигороски, Смиле Марковски и Љупчо Коцарев. Прво е дефиниран тотално асинхронизираниот проточен шифрувач, наречен EdonZ, чии алгоритми за криптирање/декриптирање се користат во дизајнот на овие кодови. Потоа опишани се стандардните (оригиналните) алгоритми за кодирање/декодирање, но во поопшта форма од предложената каде се користат симболи од a бита, наместо нибли. Дадени се резултатите од голем број експерименти за различни параметри на овие кодови за рата 1/4 и за различни веројатности за бит грешка при пренос низ бинарен симетричен канал. Од резултатите изведени се заклучоци за влијанието на секој од параметрите на кодовите врз нивните перформанси. Исто така, дадена е идеја за метод за намалување на бројот на неуспешни декодирања со кој се добива подобрување на веројатностите за точно декодирање без значително успорување на декодирањето. На крајот од оваа глава испитано е влијанието на должината на пораките врз перформансите и дадени се резултати за кодови со различни должини на пораките при иста рата на кодот.

Нови резултати во оваа глава се: детално испитување на својствата и перформансите на овие кодови и начинот на кој секој од параметрите во дизајнот на кодовите влијае над текот на процесот на декодирање и бројот на успешно завршени декодирања; изнесени се идеи и заклучоци за изборот на овие параметри и начини како да се подобрат перформансите; дефиниран е нов метод за намалување на неуспешните декодирања кои завршуваат предвреме.

Во третата глава дефиниран е нов алгоритам за кодирање/декодирање на случајните кодови базирани на квазигрупи, наречен алгоритам-за-декодирање-со-пресек (Cut-Decoding или АДП алгоритам). Овој нов алгоритам е дефиниран со цел да се забрза процесот на декодирање, бидејќи експериментите покажаа дека брзината на декодирање е најголемиот проблем кај овие кодови. Презентирани се голем број на експериментални резултати добиени со предложениот алгоритам со користење на различни параметри. Од споредбата на овие резултати со резултатите добиени со стандардниот алгоритам, покажано е дека со новиот алгоритам процесот на декодирање се забрзува за 4.5 пати за код со рата 1/4. Притоа, за поголеми вредности на веројатноста за бит грешка во бинарен симетричен канал се добиваат и подобри вредности на веројатностите за пакет-грешка и бит-грешка отколку со стандардниот алгоритам. Дефиниран е нов метод за намалување на неуспеш-

ните декодирања кои завршуваат со повеќе кандидати за декодираната порака. Исто така, предложена е и комбинација на двата дефинирани методи за намалување на неуспешните декодирања, со која се добиваат подобри вредности за веројатноста за успешно декодирање, повторно со само мало намалување на брзината на декодирањето. Испитано е влијанието на должината на пораките на перформансите на случајните кодови базирани на квазигрупи со новиот АДП алгоритам. На крајот од оваа глава испитани се перформансите на овие кодови кога во процесот на кодирање/декодирање се користат квазигрупи од ред 4 или ред 256, наместо квазигрупи од ред 16 (кои се користат во претходните експерименти). Притоа направени се повеќе експерименти со стандардниот и АДП алгоритамот за квазигрупи од ред 4 од различни класи на фракталност и линеарност.

Нови резултати во овој дел од трудот се: нов алгоритам за кодирање/декодирање на случајните кодови базирани на квазигрупи со кој процесот на декодирање за кодови со рата 1/4 се забрзува за 4.5 пати; нов метод за намалување на неуспешните декодирања кои завршуваат со повеќе кандидати; комбинација на двата дефинирани методи за намалување на неуспешните декодирања, со која се добиваат подобри вредности за веројатноста за успешно декодирање, како и испитување на перформансите на овие кодови кога во процесот на кодирање/декодирање се користат квазигрупи од ред 4 или ред 256, наместо квазигрупи од ред 16.

Перформансите на случајните кодови базирани на квазигрупи при нивна примена во кодирање/декодирање на слики кои се пренесуваат низ бинарен симетричен канал се испитани во четвртата глава. За таа цел, направени се некои модификации во алгоритмот за декодирање за разрешување на проблемот со неуспешните декодирања (случај на делумно и нееднозначно декодирана порака). Презентирани се експериментално добиените веројатности за пакет-грешка и бит-грешка добиени со користење на стандардниот алгоритам, АДП алгоритамот и Рид-Соломон кодовите. Дадени се и сликите добиени по декодирањето со соодветниот алгоритам и заклучено е дека предложениот АДП алгоритам во комбинација со методот за намалување на неуспешните (предвреме завршени) декодирања дава подобри резултати и појасни слики од другите алгоритми.

Во четвртата глава нови резултати се: модификации во алгоритмите за декодирање на случајните кодови базирани на квазигрупи за нивна примена во кодирање/декодирање на слики и испитување на перформансите на овие кодови при кодирање/декодирање на слики пренесени низ бинарен симетричен канал.

Со цел да се добие поголемо забрзување на процесот на декодирање, во петтата глава предложен е уште еден алгоритам за кодирање/декодирање за кодовите базирани на квазигрупи, наречен алгоритам-за-декодирање-со-4пресеци (4-Sets-Cut-Decoding или АД4П алгоритам). За да се подобрат и веројатностите за пакет-грешка и бит-грешка, дефинирани се неколку различни верзии на процедурата за генерирање на множествата со кандидати за декодирање (т.е., АД4П#1, АД4П#2, АД4П#3 и АД4П#4 алгоритми). Анализирани се перформансите на различните алгоритми за декодирање на RCBQ (стандардниот, АДП и АД4П алгоритмите) за код со рата $R = 1/8$. Исто така, разгледана е примената на методите за намалување на неуспешните декодирања во новите предложени алгоритми со што се добиваат уште подобри резултати за веројатноста за точно декодирање, без значително намалување на брзината на декодирањето. На крајот од оваа глава испитани се и перформансите на RCBQ со третата верзија на алгоритмот-за-декодирање-со-4пресеци за кодирање/декодирање на слики кои се пренесуваат низ бинарен симетричен канал. Добиените резултати се споредени со соодветни резултати добиени со Рид-Соломон кодовите. Од споредбата заклучено е дека за сите разгледани веројатности p за бит-грешка во бинарен симетричен канал RCBQ со новиот алгоритам дава појасни слики отколку Рид-Соломон кодовите.

Нови резултати во петтата глава се: нов алгоритам за кодирање/декодирање за кодовите базирани на квазигрупи, наречен алгоритам-за-декодирање-со-4пресеци (4-Sets-Cut-Decoding или АД4П алгоритам со четири различни методи за генерирање на множествата со кандидати за декодирање) со кој за код со рата 1/8 декодирањето е 6.3 пати побрзо отколку со стандардниот алгоритам, а за некои вредности на p се добиваат 24 пати помали вредности за веројатноста за пакет-грешка и детално испитување на перформансите на различните алгоритми за декодирање (стандардниот, АДП и АД4П алгоритмите) за код со рата $R = 1/8$; испитување на перформансите на овие кодови со новиот алгоритам за декодирање при кодирање/декодирање на слики пренесени низ бинарен симетричен канал и споредба со Рид-Соломон кодовите.

Во шестата глава изведени се некои теориски резултати за новите алгоритми предложени во претходните глави. Прво е изведена формула за теориска горна граница за веројатноста за пакет-грешка добиена со новите алгоритми. Потоа, експериментално е потврдена точноста на изведената горна граница. На крајот се изведени и приближни формули за кардиналниот број на множествата со кандидати за декодирање после редукцијата на нивните

елементи предложена во новите алгоритми.

Нови резултати во овој дел од тезата се: Теорема 6.1 за теориската веројатност за пакет-грешка добиена со новите алгоритми за кодирање/декодирање дефинирани во претходните глави и приближните формули за кардиналниот број на множествата со кандидати за декодирање после редукцијата на нивните елементи предложена во новите алгоритми.

Во седмата глава е разгледана примена на квазигрупите во криптографија. Поточно, разгледувани се квазигрупни трансформации базирани на парастрофи на квазигрупи. Првата идеја за ваква трансформација е дадена од Александар Крапеж. Во овој дел од тезата, предложена е модификација на квазигрупната трансформација за процесирање на низи базирана на парастрофи, наречена парастрофна квазигрупна трансформација (или PE-трансформација) и испитани се нејзините криптографски својства. Дадени се две нови класификации на квазигрупите од ред 4 корисни во криптографија: класификација според бројот на различни парастрофи и класификација според парастрофна фракталност. Покажано е дека со новата трансформација е зголемен бројот на квазигрупи погодни за примена во криптографијата. Даден е и математички модел за парастрофната фракталност на квазигрупите. На крајот изведен е теориски доказ за отпорност од статистички напади на пораката криптирана со користење на парастрофната квазигрупна трансфомација. За илустрирање на теориските резултати прикажани се и дел од експериментално добиените резултати.

Во последната глава нови резултати се: нова квазигрупна трансформација (PE-трансформација); две нови класификации на квазигрупите од ред 4 корисни во криптографија; математички модел за парастрофната фракталност на квазигрупите; Теорема 7.1 и Теорема 7.2.

Во прилог дадена е и англиска верзија на дисертацијата.

На крајот од оваа докторска дисертација дадени се некои заклучоци и насоки за понатамошни истражувања во овие области.

Глава 1

Квазигрупи и квазигрупни стринг трансформации

Квазигрупите и квазигрупните трансформации се многу погодни за конструкција на криптографски примитиви, кодови за откривање на грешки и кодови кои поправаат грешки. Причини за тоа се: структурата на квазигрупите, нивниот голем број, својствата на квазигрупните трансформации и друго. Затоа, во оваа глава, ќе бидат дадени дефинициите и својствата на квазигрупите и квазигрупните стринг трансформации кои ќе ги користиме во следните глави. Повеќе за квазигрупи и нивните својства може да се види во [6, 11, 36, 81].

1.1 Квазигрупи

Квазигрупа $(Q, *)$ е групоид, т.е., множество Q со бинарна операција $* : Q^2 \to Q$, за која важи следниот закон:

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

Всушност, (1.1) значи дека еден групоид $(Q, *)$ е квазигрупа ако и само ако равенките $x * u = v$ и $u * y = v$ имаат единствени решенија x и y за секои $u, v \in Q$.

Понатаму ќе претпоставиме дека множеството Q е конечно. Кардиналниот број $|Q|$ на ова множество се нарекува ред на квазигрупата. Главната табела на Келиевата шема на квазигрупата е латински квадрат на множеството Q.

За дадена квазигрупа $(Q, *)$ од операцијата $*$ може да се изведе нова операција $\backslash$, која се дефинира на следниот начин:

$$x * y = z \Leftrightarrow y = x \backslash z. \tag{1.2}$$

Тогаш, алгебрата $(Q, *, \backslash)$ ги задоволува следните идентитети:

$$x \backslash (x * y) = y \quad \text{и} \quad x * (x \backslash y) = y, \tag{1.3}$$

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

Понатаму, ќе дадеме некои својства на квазигрупите.

Дефиниција 1.1. *Квазигрупата $(Q, *)$ е комутативна ако го задоволува следниот идентитет:*

$$x * y = y * x. \tag{1.4}$$

Дефиниција 1.2. *Квазигрупата $(Q, *)$ е косо симетрична ако го задоволува следниот идентитет:*

$$(x * y) * (y * x) = const. \tag{1.5}$$

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

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

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

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

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

$$(\forall\, x \in Q)\ (x * e = x = e * x) \tag{1.8}$$

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

Дефиниција 1.6. *Квазигрупата $(Q, *)$ е десно-симетрична квазигрупа ако го задоволува следниот идентитет:*

$$(x * y) * y = x. \tag{1.9}$$

Дефиниција 1.7. *Квазигрупата $(Q, *)$ е лево-симетрична квазигрупа ако го задоволува следниот идентитет:*

$$y * (y * x) = x. \tag{1.10}$$

Дефиниција 1.8. *Квазигрупата $(Q, *)$ е тотално симетрична ако е комутативна и лево-симетрична квазигрупа (или комутативна и десно-симетрична).*

1.2 Квазигрупни стринг трансформации

Квазигрупните стринг трансформации се дефинирани на конечно множество Q (т.е., азбука Q) со квазигрупна операција $*$ и претставуваат пресликување од Q^+ во Q^+, каде Q^+ е множеството од сите непразни зборови (стрингови) формирани од елементите на Q. Ќе дефинираме две квазигрупни трансформации ([46]). Нека $l \in Q$ е даден елемент, наречен лидер. За секои $a_i, b_i \in Q$, трансформациите E_l и D_l се дефинирани на следниот начин:

$$E_l(a_1a_2\dots a_n) = (b_1b_2\dots b_n) \Leftrightarrow \left\{ \begin{array}{rcccl} b_1 & = & l & * & a_1 \\ b_i & = & b_{i-1} & * & a_i, \quad 2 \leqslant i \leqslant n \end{array} \right. \tag{1.11}$$

$$D_l(a_1a_2\dots a_n) = (c_1c_2\dots c_n) \Leftrightarrow \left\{ \begin{array}{rcccl} c_1 & = & l & * & a_1 \\ c_i & = & a_{i-1} & * & a_i, \quad 2 \leqslant i \leqslant n \end{array} \right. \tag{1.12}$$

Со користење на идентитетите (1.3) се покажува дека за трансформациите E и D точно е следното својство:

Теорема 1.1. *За секој стринг $M \in Q^+$ и за секој лидер $l \in Q$ точно е $D_{l,\backslash}(E_{l,*}(M)) = M = E_{l,*}(D_{l,\backslash}(M))$, односно $E_{l,*}$ и $D_{l,\backslash}$ се меѓусебно инверзни пермутации на Q^+.*

Ако е дадена низа $*_1, *_2, \dots, *_n$ од квазигрупни операции над Q и лидери $l_1, l_2, \dots, l_n \in Q$, може да се дефинираат пресликувања $E_1, E_2, \dots, E_n$, (каде E_i е пресликувањето со операција $*_i$ и лидер l_i). Тогаш

$$E^{(n)} = E^{(n)}_{l_n,\dots,l_1} = E_n \circ E_{n-1} \circ \dots \circ E_1,$$

каде што $\circ$ е композиција на пресликувања ($n \geqslant 1$).

Аналогно, може да се дефинира и композиција $D^{(n)}$ на квазигрупни трансформации $D_1, D_2, \dots, D_n$ со лидери $l_1, l_2, \dots, l_n$, соодветно:

$$D^{(n)} = D^{(n)}_{l_n,\dots,l_1} = D_n \circ D_{n-1} \circ \dots \circ D_1.$$

Пример 1.1. Нека $Q = \{0,1,2,3\}$ и нека квазигрупата $(Q,*)$ и нејзината парастрофа $(Q,\backslash)$ се дадени со (1.13).

$*$	0	1	2	3
0	0	1	2	3
1	1	2	3	0
2	2	3	0	1
3	3	0	1	2

$\backslash$	0	1	2	3
0	0	1	2	3
1	3	0	1	2
2	2	3	0	1
3	1	2	3	0

(1.13)

Нека $\alpha = 1\ 0\ 2\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 1\ 1\ 1\ 2\ 1\ 0\ 2\ 2\ 0\ 1\ 0\ 1\ 3\ 0$ и нека е избран лидерот 0. Во Табела 1.1 претставени се четири последователни примени на E- трансформацијата.

	$1\ 0\ 2\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 1\ 1\ 1\ 2\ 1\ 0\ 2\ 2\ 0\ 1\ 0\ 1\ 3\ 0 = \alpha$
0	$1\ 1\ 3\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 2\ 3\ 0\ 1\ 3\ 0\ 0\ 2\ 0\ 0\ 1\ 1\ 2\ 1\ 1 = E_{0,*}(\alpha)$
0	$1\ 2\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 2\ 0\ 3\ 3\ 0\ 3\ 3\ 3\ 1\ 1\ 1\ 2\ 3\ 1\ 2\ 3 = E^{(2)}_{0,*}(\alpha)$
0	$1\ 3\ 0\ 1\ 2\ 3\ 0\ 1\ 2\ 3\ 0\ 2\ 2\ 1\ 0\ 0\ 3\ 2\ 1\ 2\ 3\ 0\ 2\ 1\ 2\ 0\ 3 = E^{(3)}_{0,*}(\alpha)$
0	$1\ 0\ 0\ 1\ 3\ 2\ 2\ 3\ 1\ 0\ 0\ 2\ 0\ 1\ 1\ 1\ 0\ 2\ 3\ 1\ 0\ 0\ 2\ 3\ 1\ 1\ 0 = E^{(4)}_{0,*}(\alpha)$

Табела 1.1: Последователни E-трансформации

Во Табела 1.2 дадени се стринговите добиени со четири последователни примени на D-трансформацијата $D_{0,\backslash}$ на последниот стринг во Табела 1.1 $\beta = E^{(4)}_{0,*}(\alpha)$.

	$1\ 0\ 0\ 1\ 3\ 2\ 2\ 3\ 1\ 0\ 0\ 2\ 0\ 1\ 1\ 1\ 0\ 2\ 3\ 1\ 0\ 0\ 2\ 3\ 1\ 1\ 0 = \beta$
0	$1\ 3\ 0\ 1\ 2\ 3\ 0\ 1\ 2\ 3\ 0\ 2\ 2\ 1\ 0\ 0\ 3\ 2\ 1\ 2\ 3\ 0\ 2\ 1\ 2\ 0\ 3 = D_{0,\backslash}(\beta)$
0	$1\ 2\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 2\ 0\ 3\ 3\ 0\ 3\ 3\ 3\ 1\ 1\ 1\ 2\ 3\ 1\ 2\ 3 = D^{(2)}_{0,\backslash}(\beta)$
0	$1\ 1\ 3\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 2\ 3\ 0\ 1\ 3\ 0\ 0\ 2\ 0\ 0\ 1\ 1\ 2\ 1\ 1 = D^{(3)}_{0,\backslash}(\beta)$
0	$1\ 0\ 2\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 1\ 1\ 1\ 2\ 1\ 0\ 2\ 2\ 0\ 1\ 0\ 1\ 3\ 0 = D^{(4)}_{0,\backslash}(\beta)$

Табела 1.2: Последователни D-трансформации

Од Табела 1.2 може да се забележи дека $\alpha = D^{(4)}_{0,\backslash}(\beta) = D^{(4)}_{0,\backslash}(E^{(4)}_{0,*}(\alpha))$.

1.3 Својства на квазигрупните стринг трансформации

Во [47] авторите тврдат дека трансформацијата $E^{(n)}$ може да се користи како функција за енкрипција и ја докажуваат следната теорема.

Теорема 1.2. *Нека $\alpha = a_1a_2...a_k \in Q^+$ е произволен стринг и нека $\beta = E^{(n)}(\alpha)$. Ако k е доволно големо, тогаш m-торките од букви во β имаат рамномерна распределба за $m \leqslant n$.*

Теорема 1.2 тврди дека после n примени на E-трансформацијата на произволна порака, распределбата на подстринговите во добиената порака е рамномерна. Ова својство е многу важно за примена на една трансформација во криптографија. Имено, рамномерната распределба на подстринговите во криптираната порака гарантира отпорност од статистички напади. Во истиот труд, авторите забележуваат дека за $m > n$ распределбата на подстринговите со должина m не е рамномерна. Бидејќи, распределбата на овие подстрингови не е рамномерна, тоа може да биде искористено за напад на пораката со цел да биде откриена оригиналната порака.

Во [2] авторите ја наоѓаат распределбата на $(n + 1)$-торките од букви после примена на $E^{(n)}$- трансформацијата и ја докажуваат следната теорема и последица.

Теорема 1.3. *Нека $(p_1, p_2, \ldots, p_s)$ е распределбата на буквите во влезната порака α и нека $p_1, p_2, \ldots, p_s$ се различни веројатности, т.е., $p_i \neq p_j$ за $i \neq j$. Веројатностите на $(n + 1)$-торките во излезната порака $\beta = E^{(n)}(\alpha)$ се поделени во s класи. Секоја класа содржи s^n елементи кои имаат иста веројатност и веројатноста на секоја $(n+1)$-торка во i-тата класа е $\frac{1}{s^n}p_i$, for $i = 1, 2, \ldots, s$.*

Последица 1.1. *Ако $p_{i_1} = p_{i_2} = \cdots = p_{i_\nu}$ за некое $1 \leqslant i_1 < \cdots < i_\nu \leqslant s$ во Теорема 1.3, тогаш класите со веројатности $\frac{1}{s^n}p_{i_1}$, $\frac{1}{s^n}p_{i_2}$,..., $\frac{1}{s^n}p_{i_\nu}$ ќе бидат споени во една класа со νs^n елементи.*

Во истиот труд ([2]) авторите даваат доказ и за поопшт облик на овие резултати, т.е., тие ја наоѓаат распределбата на m-торките од букви ($m > n$) после примена на $E^{(n)}$-трансформацијата. Оваа генерализација на претходните резултати е дадена во следните теореми.

Теорема 1.4. *Нека* $(p_1, p_2, \ldots, p_s)$ *е распределбата на буквите во влезната порака* α *и нека* K *е бројот на сите различни производи* $p_{i_1} \ldots p_{i_{m-n}}$ *за* $i_1, i_2, \ldots, i_{m-n} \in A$. *Тогаш веројатностите на* m*-торките* $(m > n)$ *во излезната порака* $\beta = E^{(n)}(\alpha)$ *се поделени во* K *класи и притоа* m*-торките во една класа имаат иста веројатност.*

Последица 1.2. *Нека* r *е бројот на различни веројатности* $p_1, p_2, \ldots, p_s$ *на буквите во азбуката* A. *Ако сите производи* $p_{i_1} \ldots p_{i_{m-n}}$ *се различни, тогаш бројот на класите е* $\overline{C}_r^{m-n}$, *каде што* $\overline{C}_r^k$ *е вкупниот број на комбинации со повторување од* r *елементи од класа* k.

Од доказот на Теорема 1.4 (даден во [2]) следува дека веројатноста на m-торките од букви е $\frac{1}{s^n} p_1^{n_1} p_2^{n_2} \ldots p_r^{n_r}$ каде што $n_1 + n_2 + \cdots + n_r = m - n$. Исто така, докажани се и теореми за бројот на елементи во секоја класа. Користејќи ги овие теоретски резултати, во истиот труд авторите го даваат и долунаведениот алгоритам за статистички напад на порака криптирана со $E^{(n)}$-трансформација.

Нека пораката M е криптирана со користење на алгоритмот за криптирање базиран на квазигрупната $E^{(n)}$- трансформација за дадено n и нека е пратена порака $C = E^{(n)}(M)$. Нека некој напаѓач ја фати криптираната порака C и таа/тој знае дека за криптирање е користен алгоритмот базиран на квазигрупната $E^{(n)}$- трансформација. Ако претпоставиме дека е позната распределбата $(p_1, p_2, \ldots, p_s)$ на буквите во јазикот користен во оригиналната порака, тогаш напаѓачот може да направи статистички напад со користење на алгоритмот даден во Табела 1.3.

Првите две букви од оригиналната порака M може да не бидат откриени на овој начин, но тие може лесно да бидат најдени со пребарување (ако тоа е важно за пораката).

За примена на овој алгоритам пораката C мора да биде доволно долга за да добиените релативни фреквенции на $(n+1)$-торките бидат што поблиски до нивните веројатности во јазикот. Статистичка анализа не може да биде направена доколку пораката C е кратка. Во тој случај декриптираната порака M_1 нема да се совпаѓа со оригиналната порака M.

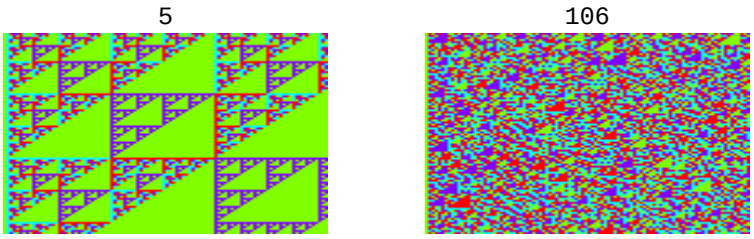
Алгоритам за статистички напад	
Влез: Криптирана порака C и распределбата $(p_1, p_2, \ldots, p_s)$ на буквите во јазикот.	
Излез: Оригиналната порака M.	
Чекор 1.	Најди ги релативните фреквенции (статистички веројатности) на буквите, паровите, тројките и.т.н., се додека не се добие распределба која не е рамномерна. Нека n е најмалиот број т.ш. $(n+1)$-торките не се рамномерно распределени.
Чекор 2.	Процесирај ја целата порака C на следниот начин. Земи ја првата $(n+1)$-торка и најди ја нејзината релативна фреквенција f.
	2.1. Најди i т.ш. $\lvert f - \frac{1}{s^n} p_i \rvert = \min\limits_{1 \leqslant j \leqslant s} \lvert f - \frac{1}{s^n} p_j \rvert$.
	2.2. Декриптирај ја оваа $(n+1)$-торка со буквата i.
	2.3. Земи ја следната $(n+1)$-торка и најди ја нејзината релативна фреквенција f и оди на чекор 2.1.

Табела 1.3: Алгоритам за статистички напад

1.4 Класификација на квазигрупи од ред 4 според фракталност и линеарност

Со користење на графички приказ на квазигрупно процесираните низи добиени со примена на квазигрупните стринг трансформации, Димитрова и Марковски во [17] даваат класификација на квазигрупите од ред 4 на фрактални и нефрактални. Оваа класификација е направена на следниот начин. Авторите почнуваат со периодична низа 123412341... со должина 100 и ја применуваат 100 пати E-трансформацијата дадена во (1.11) со дадени лидери. Трансформираната низа ја прикажуваат визуелно користејќи различна боја за секој од симболите 1, 2, 3, 4. На овој начин, добиваат графички патерн (image pattern) за секоја квазигрупа и потоа ја анализираат структурата на добиените патерни. Ако патернот има фрактална структура, соодветната квазигрупа се нарекува фрактална. Во спротивно, квазигрупата се нарекува нефрактална. Бројот на фрактални квазигрупи од ред 4 е 192, а бројот на нефрактални ква-

зигрупи е 384. На Слика 1.1 даден е по еден пример на графички приказ добиен за фрактална (а)) и нефрактална (б)) квазигрупа.



а) б)

Слика 1.1: Фрактална и нефрактална квазигрупа

Во [21], авторите даваат репрезентација на квазигрупите како 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 $x, y \in Q$

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

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 [21], квазигрупите се поделени во две класи: линеарни квазигрупи и нелинеарни квазигрупи според Булова репрезентација. Нека $(Q, *)$ е квазигрупа од ред 2^n и нека

$$f(x_1, \ldots, x_{2n}) = (f_1(x_1, \ldots, x_{2n}), \ldots, f_n(x_1, \ldots, x_{2n}))$$

е нејзиното соодветно претставување како векторско вредносна Булова функција. Двата типа на квазигрупи дефинирани се на следниот начин.

Дефиниција 1.9. *Квазигрупата е линеарна, ако сите функции f_i за $i = 1, 2, \ldots n$ се линеарни полиноми.*

Дефиниција 1.10. *Квазигрупата е нелинеарна, ако постои функција* f_i *за некое* $i = 1, 2, \ldots n$ *која не е линеарна.*

Според оваа класификација 144 квазигрупи од ред 4 се линеарни, а 432 се нелинеарни. 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

Во оваа глава опишани се случајните кодови базирани на квазигрупи (Random Codes Based on Quasigroups - RCBQ). Опишан е начинот на кој се додава редундантната информација, алгоритмите за кодирање и декодирање. RCBQ се предложени од Данило Глигороски, Смиле Марковски и Љупчо Коцарев во [30]. Овие кодови имаат неколку параметри и претставуваат комбинација од криптографски алгоритми и кодови кои поправаат грешки.

Најчесто за да се добие код кој поправа грешки и кој е отпорен на напади добиените кодни зборови се криптираат со некои од познатите шифрувачи пред тие да бидат пратени низ несигурен канал. Во тој случај се користат два алгоритми, еден за кодирање со код кој порава грешки и друг алгоритам за криптирање. Концептот на криптокодирање ги комбинира овие два процеси на кодирање и криптирање во еден алгоритам. RCBQ се всушност крипто-кодови. Имено, овие кодови се дефинирани со користење на криптографски алгоритам во самиот процес на кодирање/декодирање и затоа тие со еден алгоритам овозможуваат не само поправање на одреден број на грешки настанати при преносот на пораките низ канал со пречки, туку исто така тие овозможуваат и безбедност на информациите. Ако информацијата е кодирана со користење на овие кодови, тогаш примателот може да ја декодира и да ја добие оригиналната информација само ако тој/таа знае точно кои параметри биле користени во процесот на кодирање, дури и ако комуникацискиот канал е без пречки. Во оваа теза ние ги разгледуваме RCBQ како кодови кои поправаат грешки и затоа тука нема да бидат анализирани нивните криптографски својства.

Во дизајнот на RCBQ се користат алгоритмите за криптирање/декриптирање од имплементацијата на тотално асинхронизиран проточен шифрувач (Totally Asynchronous Stream Cipher - TASC) со користење на квазигрупни стринг трансформации [27]. Овие криптографски алгоритми користат азбука Q и квазигрупна операција $*$ над Q заедно со нејзината парастрофа $\backslash$. Ние ќе ги опишеме овие кодови со користење на квазигрупи, но од дефиницијата на алгоритмите јасно е дека во нивниот дизајн може да бидат користени и други алгоритми за криптирање и декриптирање. Авторите во дизајнот на RCBQ ја користат азбуката од нибли $Q = \{0, 1, \ldots, 9, a, b, c, d, e, f\}$. Но, тука ние ќе ги дефинираме алгоритмите за кодирање и декодирање во поопшт случај со користење на симболи од a бита, наместо нибли.

Во мојата магистерска работа [89] и трудот [68] го испитавме влијанието на избраните параметри на перформансите на овие кодови и предложивме метод за намалување на бројот на неуспешни декодирања. Во ова поглавје, прво накратко ќе дадеме дел од тие резултати. Потоа, ќе го испитаме влијанието на должината на пораките врз перформансите на кодот.

2.1 TASC и EdonZ

Концептот на TASC е дефиниран во [27]. Овој криптографски концепт е основата на случајните кодови базирани на квазигрупи. Затоа во ова поглавје накратко ќе ги објасниме овие шифрувачи.

TASC е концепт на проточен шифрувач кој се разликува од концептите на синхронизирани и само-синхронизирани проточни шифрувачи.

Дефиниција 2.1. *Синхронизиран проточен шифрувач е оној во кој низата клучеви се генерира независно од влезната порака и шифрираниот текст. Процесот на криптирање со сихронизиран проточен шифрувач може да се опише со следните равенки:*

$$\sigma_{i+1} = f(\sigma_i, k), \quad z_i = g(\sigma_i, k), \quad c_i = h(z_i, m_i),$$

каде σ_0 е почетната состојба и може да биде определена од клучот k, f е функција за следната состојба, g е функцијата со која се определува низата клучеви z_i, и h е излезната функција со која од клучевите и влезниот текст m_i се добива шифрираниот текст c_i.

Дефиниција 2.2. *Само-синхронизиран или асинхронизиран проточен шифрувач е оној во кој низата клучеви се генерира како функција од клучот и фиксен број на претходно шифрирани цифри. Функцијата за криптирање на само-синхронизиран проточен шифрувач може да се опише со следните равенки:*

$$\sigma_i = (c_{i-l}, c_{i-l+1}, ..., c_{i-1}), \quad z_i = g(\sigma_i, k), \quad c_i = h(z_i, m_i),$$

каде $\sigma_0 = (c_{-l}, c_{-l+1}, ..., c_{-1})$ е почетната состојба, k е клучот, g е функцијата со која се формира низата клучеви z_i, и h е излезната функција со која од низата клучеви и влезниот текст m_i се формира шифрираниот текст c_i.

Дефиниција 2.3. *Тотално асинхронизиран проточен шифрувач е оној во кој низата клучеви се генерира како функција од меѓу-клучот и сите претходни букви во влезната порака.*

Функцијата за криптирање на еден тотално асинхронизиран 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 декриптирање на тотално асинхронизиран 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 асинхронизираниот 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. EdonZ работи со азбуката $Q = \{0, 1, \ldots, 9, a, b, c, d, e, f\}$ од нибли, чии елементи се 4-битните зборови. Функцијата за криптирање ја користи квазигрупата $(Q, *)$, а функцијата за декриптирање ја користи нејзината парастрофа $(Q, \backslash)$. Во функциите се користи и операцијата исклучително или - $\oplus$ (XOR) на 4-битни променливи. Функциите за криптирање и декриптирање на EdonZ се дадени подолу преку нивните алгоритми (Слика 2.2).

Важно е да се забележи следното својство на EdonZ. Нека $M = M_1M_2...M_k$ е претставувањето на пораката M во подблокови M_i, каде $M_1 = m_1...m_{i_1}$, $M_2 = m_{i_{1+1}}...m_{i_2}$, ..., и нека C е шифрираниот текст добиен од M. Тогаш $C = C_1C_2...C_k$ каде секој блок C_i е шифрираниот текст добиен од соодветниот влезен (блок) текст M_i. Ова својство значи дека иако EdonZ е проточен шифрувач, влезниот текст може да се обработува во помали делови (блокови) и притоа без разлика кое претставување на M во блокови ќе се избере, крајниот шифриран текст ќе биде ист (Ова својство се користи при воведувањето на редундантната информација во процесот на кодирање на RCBQ).

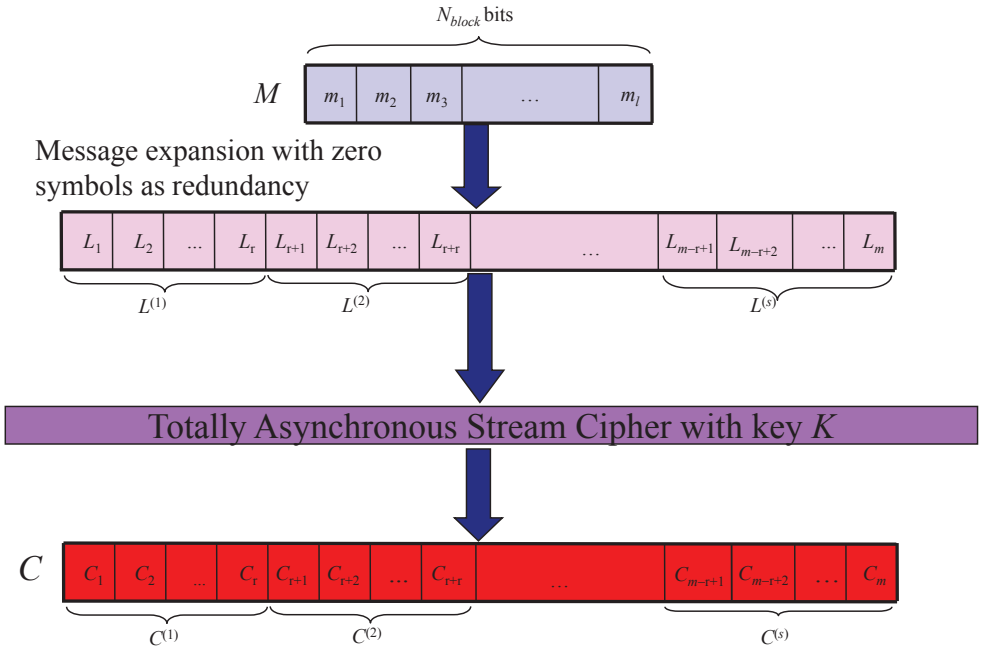
Главна карактеристика на TASC е тоа што пренесувањето на грешката е неограничено и се шири сè до крајот на низата. Но, со додавање на редундантна информација во низата, може дел од грешките да бидат поправени. Ова е всушност главната идеја на TASC Error Correction.

2.2 Опис на кодирањето

Прво, низата од битови, добиена од изворот на информација, се дели во пораки (блокови) од N_{block} битови. Потоа, секој од $2^{N_{block}}$ блоковите се пресликува во коден збор со должина од $N > N_{block}$ битови.

Нека $M = m_1m_2...m_l$ е еден блок (порака) од $N_{block} = la$ битови, каде $m_i \in Q$ и Q е азбуката од сите a-битни симболи (значи Q има 2^a симболи). Прво, се додава редундантна информација од v a-битни нулти симболи и се добива блок $L = L^{(1)}L^{(2)}...L^{(s)} = L_1L_2...L_m$ од N битови, каде што $L^{(i)}$ е подблок од r симболи од азбуката Q, а $L_i \in Q$. Да забележиме дека $N = ma$, $m = l + v$ и $m = rs$. Со бришење на редундантните нулти симболи во секој подблок $L^{(i)}$, од пораката L се добива оригиналната порака M. На овој начин се добива (N_{block}, N) код со рата $R = N_{block}/N$. Кодниот збор се добива со примена на алгоритмот за криптирање (даден на Слика 2.2) на блокот L. За таа цел, претходно треба да се избере клуч $k = k_1k_2 \dots k_n \in Q^n$. Добиениот коден збор за M е $C = C_1C_2...C_m$, каде $C_i \in Q$.

Процесот на кодирање шематски е претставен на Слика 2.1.



Слика 2.1: Кодирање со стандардниот алгоритам

2.3 Опис на декодирањето

После преносот низ канал со пречки (во нашите експерименти користиме бинарен симетричен канал), кодниот збор C може да се трансформира и нека примената порака е $D = D^{(1)}D^{(2)}...D^{(s)} = D_1D_2...D_m$, каде $D^{(i)}$ се подблокови од r симболи од Q, а $D_i \in Q$. Процесот на декодирање се состои од четири чекори:

(i) процедура за генерирање на множествата со преддефинирано Хамингоново растојание,

(ii) инверзен алгоритам на алгоритмот за кодирање,

(iii) процедура за генерирање на множествата со кандидати за декодирање,

(iv) правило за декодирање.

Криптирање	**Декриптирање**
Влез: Клуч $k = k_1k_2 \dots k_n$ и $L = L_1L_2 \dots L_m$ **Излез**: коден збор $C = C_1C_2...C_m$	**Влез**: Парот $(a_1a_2 \dots a_s, k_1k_2 \dots k_n)$ **Излез**: Парот $(c_1c_2 \dots c_s, K_1K_2 \dots K_n)$
За $j = 1$ to m $X \leftarrow L_j$; $T \leftarrow 0$; За $i = 1$ до n $X \leftarrow k_i * X$; $T \leftarrow T \oplus X$; $k_i \leftarrow X$; $k_n \leftarrow T$ **Излез**: $C_j \leftarrow X$	За $i = 1$ до n $K_i \leftarrow k_i$; За $j = 0$ до $s - 1$ $X, T \leftarrow a_{j+1}$; $temp \leftarrow K_n$; За $i = n$ до 2 $X \leftarrow temp \setminus X$; $T \leftarrow T \oplus X$; $temp \leftarrow K_{i-1}$; $K_{i-1} \leftarrow X$; $X \leftarrow temp \setminus X$; $K_n \leftarrow T$; $c_{j+1} \leftarrow X$; **Излез**: $(c_1c_2 \dots c_s, K_1K_2 \dots K_n)$

Слика 2.2: Алгоритми за криптирање и декриптирање

Генерирање на множества со преддефинирано Хамингоново растојание – Веројатноста дека во $D^{(i)}$ најмногу t битови ќе бидат погрешно пренесени е

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

каде p е веројатноста за бит-грешка во бинарен симетричен канал. Нека B_{max} е цел број така што $1 - P(p; B_{max}) \leqslant q_B$ за дадено q_B $(0 < q_B \leqslant 1)$. Се формираат множествата

$$H_i = \{\alpha | \alpha \in Q^r, \;\; H(D^{(i)}, \alpha) \leqslant B_{max}\},$$

за секое $i = 1, 2, \dots, s$, каде $H(D^{(i)}, \alpha)$ е Хамингоновото растојание помеѓу $D^{(i)}$ и α. Тогаш со веројатност најмалку $1 - q_B$, блокот $C^{(i)}$ е елемент на

множеството H_i, за $i = 1, 2, \ldots, s$. Бројот на елементи во множествата H_i е

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

Бројот B_{checks}, всушност, 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 алгоритмот за кодирање – Инверзниот алгоритам на алгоритмот за кодирање е всушност алгоритмот за декриптирање на TASC даден на Слика 2.2.

Генерирање на множествата со кандидати за декодирање – Множествата со кандидати за декодирање S_0, S_1, S_2,..., S_s се дефинираат итеративно. Нека $S_0 = (k_1 \ldots k_n; \lambda)$, каде λ 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 пар $(\beta, w_1 w_2 \ldots\ w_{ra(i-1)}) \in S_{i-1}$ и секој елемент $\alpha \in H_i$, се применува инверзниот алгоритам (т.е. алгоритмот за декриптирање даден на Слика 2.2) со влез (α, β). Ако излез од алгоритмот е парот (γ, δ) и притоа двете низи γ и $L^{(i)}$ имаат редундантни нулти симболи на исти позиции, тогаш парот $(\delta, w_1 w_2 \ldots w_{ra(i-1)} c_1 c_2 \ldots c_{ra}) \equiv (\delta, w_1 w_2 \ldots w_{rai})$ 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 $(d_1 \ldots d_n, w_1 \ldots w_{ras})$, тогаш $L = w_1 \ldots w_{ras}$ е декодираната (редундантна) порака. Во тој случај, велиме дека имаме *успешно декодирање*.

- Ако добиената декодирана порака не е точна, тогаш имаме *некоригирана-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 $j \in \{1, \ldots, s\}$, тогаш процесот на декодирање се стопира (велиме дека се појавила грешка од типот *грешка-празно-множество*). Тогаш може да се заклучи дека при преносот на $D^{(i)}$, за некое $i \leqslant j$, се појавиле повеќе од B_{max} грешки и затоа $C_i \notin H_i$.

Теорема 2.1. [30] *Веројатноста за пакет-грешка на овие кодови е*

$$PER_t = 1 - (1 - q_B)^s.$$

Од доказот на оваа теорема даден во [30], јасно е дека во веројатноста PER_t не се предвидени неуспешните декодирања со *грешка-повеќе-кадидати*.

2.4 Избирање параметри за оптимален RCBQ

Кодовите за поправање на грешки базирани на квазигрупи имаат неколку параметри и ние го испитавме влијанието на изборот на параметрите врз перформансите на кодот. Бидејќи овие кодови се дефинирани со користење на квазигрупните стринг трансформации на пораки проширени со додавање на редундантност, укажавме како

– патернот за редундантноста

– должината на почетниот клуч

– избраната квазигрупа

делува на перформансите на кодот. Во овие експерименти користена е азбуката од нибли.

Експериментите се направени на следниот начин. Прво, влезната порака се проширува со различни патерни за додавање на редундантните нулти нибли. Потоа, проширената порака се кодира и се пренесува низ бинарен симетричен канал со веројатност p за бит-грешка. Излезната порака од симулираниот канал се декодира и ако процесот на декодирање заврши успешно (последното множество S_s со кандидати за декодирање има само еден елемент), декодираната порака се споредува со влезната порака. Ако тие се разликуваат во барем еден бит (се појавила *некорегирана-грешка*) се определува бројот на неточно декодирани битови како Хамингоново растојание

помеѓу влезната и декодираната порака. Експериментите покажаа дека овој тип на грешка се појавува многу ретко.

Исто така, во нашите експерименти се пресметува и бројот на неточно декодирани битови кога процесот завршува со *грешка-повеќе-кандидати* или со *грешка-празно-множество*. Тогаш, бројот на неточни битови се пресметува на следниот начин.

Кога ќе се појави *грешка-празно-множество*, т.е., $S_i = \emptyset$, ги земаме сите елементи (без редундантните симболи) од множеството S_{i-1} и го наоѓаме нивниот заеднички префикс. Ако овој заеднички префикс е со должина k бита и должината на пратената порака е m бита ($k \leqslant m$), тогаш го споредуваме овој подстринг со првите k бита од пратената порака. Ако тие се разликуваат во b бита, тогаш бројот на неточно декодирани бита е $m - k + b$.

Ако се појави *грешка-повеќе-кандидати*, тогаш ги земаме сите елементи од множеството S_s и го наоѓаме нивниот заеднички префикс. Потоа бројот на неточно декодирани бита се пресметува како и претходно.

Вкупниот број на неточно декодирани битови е збир на бројот на неточно декодирани битови во сите појавени неуспешни декодирања.

Веројатноста за пакет-грешка се пресметува на следниот начин

$$PER = \frac{\#(\textit{неточно декодирани пакети (пораки)})}{\#(\textit{сите пакети})}$$

и веројатноста за бит-грешка

$$BER = \frac{\#(\textit{неточно декодирани битови во сите пакети})}{\#(\textit{битови во сите пакети})}.$$

Експерименталните резултати се дадени во табели. Во секоја табела дадени се:

- теориската веројатност за пакет-грешка (PER_t);
- експерименталната веројатност за пакет-грешка без грешките од тип *грешка-повеќе-кандидати* (PER_{s_1});
- експерименталната веројатност за пакет-грешка со сите неуспешно декодирани пораки (PER_s);

– експериментално добиената веројатност за кумулативна бит-грешка (BER_s).

Дадени се резултати добиени за различни вредности на веројатноста p за бит-грешка на бинарен симетричен канал и $B_{max} = 3$ и $B_{max} = 4$ додека $BER_s < p$. За $B_{max} > 4$, експериментите не завршуваат во реално време.

2.4.1 Патерн за редундантност

Направени се експерименти за 6 различни патерни за додавање на редундантните нулти нибли за кодот (72,288) со рата R=1/4. Во овие експерименти користена е квазигрупата дадена во Табела 2.1 (и нејзината парастрофа дадена во Табела 2.2), почетен клуч $k = 01234$ и шест патерни дадени во Табела 2.3.

$*$	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

Табела 2.1: Квазигрупата од ред 16 која се користи во експериментите

\	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	8	7	2	0	d	3	6	5	c	e	f	9	1	a	b	4
1	0	5	a	1	f	9	8	6	4	2	b	7	c	3	e	d
2	1	0	f	9	4	5	a	b	d	7	c	e	3	8	2	6
3	b	3	c	8	5	f	0	9	a	4	7	1	d	e	6	2
4	2	f	9	7	0	1	4	3	b	6	a	5	e	c	d	8
5	3	2	5	a	6	c	f	8	e	d	1	b	7	9	4	0
6	7	d	0	3	b	e	c	f	5	a	2	8	4	6	9	1
7	d	4	b	f	c	8	7	e	6	1	3	a	2	5	0	9
8	9	8	3	e	a	7	2	1	f	b	4	6	0	d	c	5
9	f	6	e	5	2	a	b	c	8	3	d	0	9	4	1	7
a	4	9	d	b	1	6	5	7	3	0	e	c	f	2	8	a
b	a	e	4	6	7	2	9	0	1	f	5	d	8	b	3	c
c	6	c	1	d	e	0	3	4	9	5	8	2	a	f	7	b
d	c	a	8	4	3	b	1	d	2	9	0	f	6	7	5	e
e	5	1	6	2	8	d	e	a	7	c	9	4	b	0	f	3
f	e	b	7	c	9	4	d	2	0	8	6	3	5	1	a	f

Табела 2.2: Парастрофата на квазигрупата дадена во Табела 2.1

Во патерните, дадени во Табела 2.3, со 1 означено е местото на симбол од пораката, а со 0 редундантниот симбол.

patt.1	**patt.2**	**patt.3**	**patt.4**	**patt.5**	**patt.6**
1000 1000	1100 1100	1100 1100	1100 1100	1100 1000	1100 1100
1000 1000	0000 1100	1000 0000	1100 0000	0000 1100	1000 0000
1000 1000	1100 0000	1100 1000	0000 1100	1000 0000	1100 1100
1000 1000	1100 1100	1000 0000	1100 1100	1100 1000	1000 0000
1000 1000	0000 1100	1100 1100	0000 0000	0000 1100	1100 1100
1000 1000	1100 0000	1000 0000	1100 1100	1000 0000	1000 0000
1000 1000	1100 0000	1100 1000	1100 0000	1100 1000	1000 1000
1000 1000	0000 0000	1000 0000	0000 0000	0000 1100	1000 0000
1000 1000	0000 0000	0000 0000	0000 0000	1000 0000	0000 0000

Табела 2.3: Патерни за додавање на редундантните нулти нибли

Резултатите добиени со првиот патерн дадени се во Табела 2.4. Тие покажуваат дека со овој патерн, за додавање на редундантните нибли, се јавуваат

голем број на неуспешно завршени декодирања со *грешка-повеќе-кандидати*. Но, важно е да се забележи дека бројот на елементи во множествата S_i во процесот на декодирање е мал и затоа процесот на декодирање трае пократко од декодирањето при користење на другите патерни.

$B_{max} = 3$				
p	PER_t	PER_{s_1}	PER_s	BER_s
0.02	0.004314	0.004392	0.1185916	0.008851
0.03	0.019674	0.019153	0.1329925	0.016889
0.04	0.055435	0.055659	0.1720190	0.036934
0.05	0.118838	0.117728	0.2258065	0.071293
$B_{max} = 4$				
0.03	0.001447	0.000792	0.6057028	0.084506
0.04	0.005541	0.004896	0.6107431	0.085522

Табела 2.4: Експериментални резултати за **patt.1**

$B_{max} = 3$				
p	PER_t	PER_{s_1}	PER_s	BER_s
0.02	0.004314	0.004248	0.0060484	0.003320
0.03	0.019674	0.020449	0.0225374	0.010699
0.04	0.055435	0.054147	0.0559476	0.025255
0.05	0.118838	0.116431	0.1183036	0.053244
$B_{max} = 4$				
0.03	0.001447	0.000500	0.2425000	0.221236
0.04	0.005541	0.004902	0.2504456	0.225082

Табела 2.5: Експериментални резултати за **patt.2**

Од резултатите добиени со вториот патерн, дадени во Табела 2.5, може

да се забележи дека со овој патерн се добиваат подобри резултати отколку со првиот, но само за $B_{max} = 3$ во процесот на декодирање. За $B_{max} = 4$ процесот на декодирање многу често завршува неуспешно со *грешка-повеќе-кандидати*, иако во овој патерн имаа доволен број на нулти нибли на крајот. Исто така, за $B_{max} = 4$ со овој патерн се добиваат поголеми вредности за BER_s во споредба со резутатите за првиот патерн.

Експерименталните резултати добиени со користење на третиот патерн за додавање на редундантноста дадени се во Табела 2.6. Може да се забележи дека за овој патерн веројатностите за пакет-грешка и бит-грешка се многу помали отколку за претходните два патерни.

$B_{max} = 3$				
p	PER_t	PER_{s_1}	PER_s	BER_s
0.02	0.004314	0.003888	0.0050403	0.00268
0.03	0.019674	0.018073	0.0191532	0.00949
0.04	0.055435	0.052275	0.0531394	0.02529
0.05	0.118838	0.119599	0.1201037	0.05558
$B_{max} = 4$				
0.03	0.001447	0.001658	0.0129353	0.00969
0.04	0.005541	0.005484	0.0170968	0.01128
0.05	0.015319	0.021759	0.0323283	0.02188
0.06	0.034361	0.037812	0.0484375	0.03051
0.07	0.066467	0.065333	0.0756667	0.04645
0.08	0.114889	0.119062	0.1281250	0.07662

Табела 2.6: Експериментални резултати за **patt.3**

Резултатите добиени за четвртиот патерн дадени се во Табела 2.7. Четвртиот патерн има доволно нулти блокови на крајот и после блоковите со информациски нибли, но повторно експериментално добиените веројатности за пакет и бит грешка се добри само за $B_{max} = 3$.

$B_{max} = 3$				
p	PER_t	PER_{s_1}	PER_s	BER_s
0.02	0.004314	0.005112	0.0115927	0.006317
0.03	0.019674	0.020593	0.0273617	0.014002
0.04	0.055435	0.056596	0.0623559	0.030743
0.05	0.118838	0.118159	0.1250000	0.061019
$B_{max} = 4$				
0.03	0.001447	0.002273	0.1859091	0.147449
0.04	0.005541	0.006364	0.1959091	0.149223

Табела 2.7: Експериментални резултати за **patt.4**

Експерименталните резултати за PER_s (Табела 2.8) добиени со петтиот патерн се слични со резултатите добиени со претходниот патерн, но има подобри резултати за веројатностите за бит-грешка.

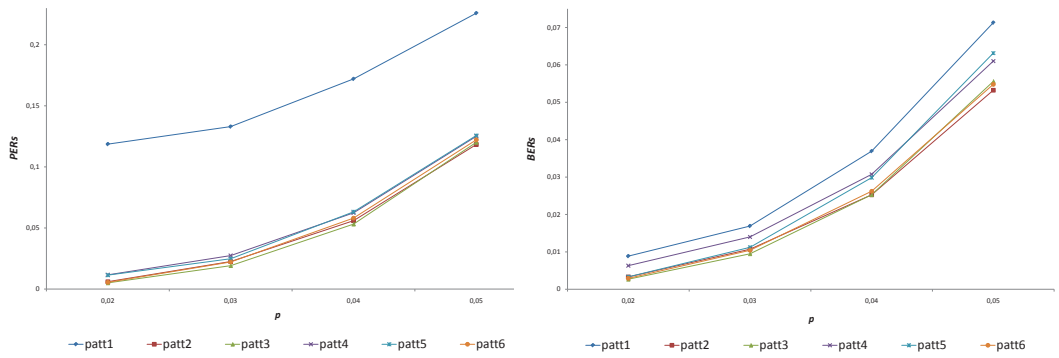
$B_{max} = 3$				
p	PER_t	PER_{s_1}	PER_s	BER_s
0.02	0.004314	0.005184	0.0113767	0.003302
0.03	0.019674	0.019441	0.0247696	0.011202
0.04	0.055435	0.058179	0.0631480	0.029845
0.05	0.118838	0.121039	0.1257201	0.063158
$B_{max} = 4$				
0.03	0.001447	0.000625	0.2534375	0.047635
0.04	0.005541	0.005000	0.2606250	0.052908
0.05	0.015319	0.012188	0.2596875	0.053342
0.06	0.034361	0.030938	0.2615625	0.063424

Табела 2.8: Експериментални резултати за **patt.5**

$B_{max} = 3$				
p	PER_t	PER_{s_1}	PER_s	BER_s
0.02	0.004314	0.003672	0.0054723	0.002928
0.03	0.019674	0.020521	0.0221774	0.010418
0.04	0.055435	0.056164	0.0581077	0.026228
0.05	0.118838	0.120679	0.1221198	0.054786
$B_{max} = 4$				
0.03	0.001447	0.001667	0.0456667	0.038995
0.04	0.005541	0.006333	0.0533333	0.045074
0.05	0.015319	0.013333	0.0590000	0.048574
0.06	0.034361	0.035000	0.0800000	0.063222
0.07	0.066467	0.063667	0.1103333	0.082676

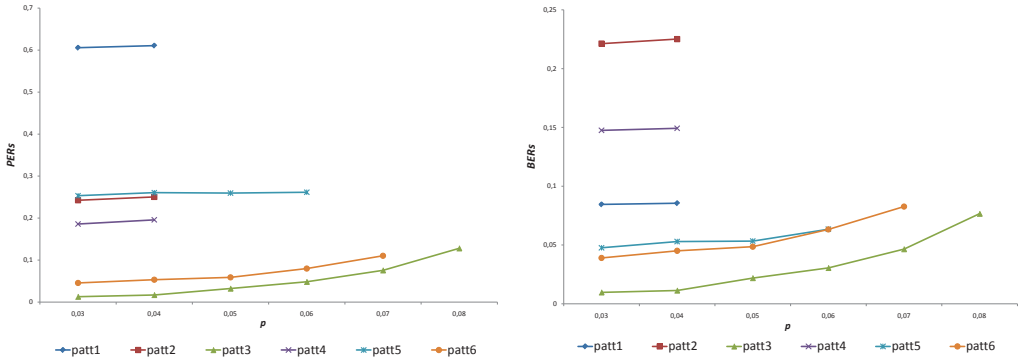
Табела 2.9: Експериментални резултати за **patt.6**

Резултатите добиени со шестиот патерн, дадени во Табела 2.9, покажуваат дека овој патерн дава добри резултати за $B_{max} = 3$ и $p < 0.05$ и за $B_{max} = 4$ и $p < 0.07$.



a) b)

Слика 2.3: PER_s и BER_s за сите шест патерни и $B_{max} = 3$



a) b)

Слика 2.4: PER_s и BER_s за сите шест патерни и $B_{max} = 4$

Од експерименталните резултати добиени за сите шест предложени патерни (претставени на Слика 2.3 за $B_{max} = 3$ и Слика 2.4 за $B_{max} = 4$) може да се заклучи дека најдобри резултати за PER и BER во декодирањето на пораки кои се пренесуваат низ бинарен симетричен канал се добиени со третиот предложен патерн, посебно за $B_{max} = 4$.

2.4.2 Должина на клуч

Теориската веројатност за пакет-грешка дадена во Теорема 2.1 е определена под претпоставка дека кодот е случаен (т.е., r-торките во секој коден збор со должина N, $r \leqslant N$ имаат рамномерна распределба). Затоа, во оваа теорема неуспешните декодирања со *грешка-повеќе-кандидати* не се предвидени. Во Теорема 1.2 докажано е дека доколку на еден стринг се применат t квазигрупни трансформации, тогаш во добиениот стринг n-торките од букви имаат рамномерна распределба за $n \leqslant t$. Во дизајнот на овие скоро-случајни кодови, должината на клучот k одредува колку пати на проширената порака ќе се примени квазигрупната трансформација за од неа да се добие кодниот збор. Затоа, колку е клучот подолг толку кодот ќе биде послучаен. Ова значи дека при поголема должина на клучот експериментално добиените вредности за PER треба да бидат поблиски до теориските вредности за PER, т.е., бројот на неуспешни декодирања со *грешка-повеќе-кандидати* да се намали. Затоа, направени се експерименти со користење на третиот патерн за редундантност (кој даде најдобри резултати) и клуч со должина 10. Од добиените

резултати, дадени во Табела 2.10, може да се забележи дека во некои експерименти воопшто нема неуспешни декодирања со *грешка-повеќе-кандидати* (во тој случај во табелата вредностите за PER_{s_1} и PER_s се болдирани) или ако вакви грешки се појават, тогаш нивниот број е многу мал (максимум 5 во експеримент со 3500 пораки и затоа немаше потреба да правиме експерименти со подолг клуч). Од експериментите може да се заклучи дека со користење на подолг клуч, може да се добијат подобри резултати за PER со скоро исто времетраење на процесот на декодирање.

$B_{max} = 3$				
p	PER_t	PER_{s_1}	PER_s	BER_s
0.02	0.004314	**0.004752**	**0.004752**	0.002093
0.03	0.019674	**0.018433**	**0.018433**	0.008493
0.04	0.055435	**0.055588**	**0.055588**	0.026289
0.05	0.118838	**0.117584**	**0.117584**	0.054876
$B_{max} = 4$				
0.03	0.001447	0.002188	0.003125	0.001359
0.04	0.005541	0.005313	0.005938	0.003429
0.05	0.015319	0.014688	0.015938	0.009279
0.06	0.034361	**0.035938**	**0.035938**	0.022396
0.07	0.066467	0.065000	0.066563	0.040647
0.08	0.114889	0.112500	0.113125	0.064852

Табела 2.10: Експериментални резултати за должина на клуч 10

Од друга страна, должината на клучот не е единствен параметар кој има влијание на вредностите за PER. Имено, направивме експерименти со должина на клучот 10 и првиот патерн за редундантност, но бројот на *грешки-повеќе-кандидати* не беше помал споредено со претходните експерименти со користење на клуч со помала должина. Оттука, може да се заклучи дека секој параметар во конструкцијата на овие кодови има големо влијание на перформансите на кодот, т.е., параметрите се взаемно зависни.

2.4.3 Избор на квазигрупа

Исто така, направени се експерименти и со неколку различни квазигрупи, кои покажаа дека изборот на квазигрупата не влијае само на вредностите за PER и BER, туку има големо влијание и на брзината на декодирањето. Со ова уште еднаш се потврди дека изборот на сите параметри при дефинирањето на кодот има големо влијание на перформансите на добиениот код.

Прво направен е експеримент со цикличната квазигрупа од ред 16 и клуч со должина 10. Експериментот со третиот патерн беше премногу спор. Затоа, направен е експеримент со користење на првиот патерн за бинарен симетричен канал со веројатност за бит-грешка $p = 0.02$ и $B_{max} = 3$. Притоа, добиено е $PER_s = 0.734087$ и $BER_s = 0.460359$, што е многу полошо од веројатностите ($PER_s = 0.1186$, $BER_s = 0.0089$) добиени со квазигрупата во Табела 2.1).

Потоа, направени се експерименти со квазигрупа од ред 16 добиена на следниот начин. Со директен производ на квазигрупа од ред 2 добиена е квазигрупа од ред 4. Потоа, со директен производ на оваа квазигрупа добиена е квазигрупа од ред 16. Експериментални резултати за вака добиената квазигрупа се полоши и од резултатите добиени со цикличната квазигрупа. За првиот патерн, $p = 0.02$ и $B_{max} = 3$ добиено е $PER_s = 0.99424$ и $BER_s = 0.80869$.

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 2.1 е пример на нефрактална квазигрупа и резултатите добиени со оваа квазигрупа се сосема задоволителни.

Од претходните примери, може да се заклучи дека изборот на квазигрупата има огромно влијание над перформансите на кодот.

Во трудот [15], авторите го анализираат коефициентот на растење на периодата за квазигрупите. Овој коефициент покажува за колку пати во просек се зголемува периодата на една случајна низа после една примена на квазигрупната трансформација. На крајот од овој труд дадени се два примери на квазирупи од ред 16, една со многу висок и една со многу низок коефициент на растење на периодата. Направени се експерименти со користење на овие две квазигрупи и третиот патерн за редундантност. Од добиените експериментални резултати дадени во [68] може да се заклучи дека има многу мали разлики во вредностите за PER и BER, иако разликата во коефициентот на растење на периодата добиен за овие квазигрупи е голема. Оттука може да се претпостави дека овој коефициент на растење на периодата нема големо

влијание на перформансите на кодот, но сепак за потврда потребни се експерименти со повеќе различни квазигрупи.

Од експерименталните резултати, може да се заклучи дека најдобри резултати за RCBQ(72,288) се добиени со користење на третиот патерн, клуч со должина 10, квазигрупата дадена во Табела 2.1 (заедно со нејзината парастрофа) и $B_{max} = 4$. Затоа, понатамошните подобрувања на перформансите на RCBQ ќе бидат споредувани со резултатите добиени за овие параметри.

2.5 Метод за намалување на бројот на *грешки-празно-множество*

Сите претходни експерименти со различни патерни и квазигрупи се направени за да се види како овие параметри на кодот влијаат на бројот на неуспешни декодирања со *грешка-повеќе-кандидати*. Промената на овие параметри нема големо влијание на бројот на неуспешни декодирања со *грешка-празно-множество*, бидејќи нивниот број е предвиден со теориската веројатност за пакет-грешка, дадена во Теорема 2.1, која не зависи од овие параметри.

Неуспешните декодирања со *грешка-празно-множество* се јавуваат кога при преносот низ каналот во некој од подблоковите на кодираната порака ќе се појават повеќе од предвидените B_{max} бит грешки. Оттука, јасно е дека некои од овие грешки може да бидат елиминирани доколку се поништат одреден број итерации од процесот на декодирање и потоа сите или дел од нив да се повторат со поголема вредност на B_{max}. Со оваа постапка ќе бидат разрешени само дел од овие неуспешни декодирања, бидејќи не е познато точно во која итерација се случило во множеството со кандидати за декодирање да не влезе точниот подблок и колку точно грешки се појавиле во подблокот при преносот, $B_{max} + 1$, $B_{max} + 2$ или повеќе. Освен тоа, поништувањето на итерациите го успорува декодирањето и може да дојде до натрупување на елементи во множествата S_i што повторно ќе доведе до неуспешно декодирање, но сега со *грешка-повеќе-кандидати*.

За да го покажеме ова, ги повторивме експериментите со третиот патерн на следниот начин. Ако во некоја итерација од процесот на декодирање (на пример во i-тата) се добие празно множество со кандидати за декодирање, тогаш се поништуваат претходните две итерации ($(i-1)$-та и $(i-2)$-та). Потоа, $(i-2)$-та итерација се извршува со $B_{max} = B_{max}+1$, а следните итерации продолжуваат со старата вредност на B_{max} . Ако повторно се добие празно множество во истата итерација (i-тата итерација), тогаш процесот на декоди-

рање завршува со неуспешно декодирање. Но, ако се добие празно множество во некоја следна итерација $((i+1)$-та, $(i+2)$-та, ...$)$, тогаш се повторува истата постапка со поништување итерации за таа итерација.

$B_{max} = 3$	
p	Процент на разрешени *грешки-празно-множество*
0.02	16.67 %
0.03	18.33 %
0.04	15.01 %
0.05	18.30 %
$B_{max} = 4$	
0.03	0.00 %
0.04	17.65 %
0.05	2.86 %
0.06	14.88 %
0.07	14.29 %
0.08	10.26 %

Табела 2.11: Процент на разрешени неуспешни декодирања со *грешка-празно-множество*

Во Табела 2.11 даден е процентот на разрешени неуспешни декодирања со *грешка-празно-множество* добиен со користење на опишаната модификација во процесот на декодирање. Може да се заклучи дека оваа модификација дава добри резултати, а процесот на декодирање е само малку побавен од претходно.

2.6 Влијанието на должината на пораките на перформансите на кодот

Во ова поглавје испитано е влијанието на должината на пораките на перформансите на кодовите кои поправаат грешки базирани на квазигрупи. За таа цел направени се експерименти за кодови со иста рата, но различни должини на пораки.

Кај познатите кодови кои поправаат грешки, со зголемување на должината на пораките веројатностите за бит-грешка и пакет-грешка се намалуваат. Во ова поглавје ќе покажеме дека ова не важи за кодовите базирани на квазигрупи. Во Теорема 2.1, дадена е формулата за теориска веројатност за пакет-грешка која гласи

$$PER_t = 1 - (1 - q_B)^s.$$

Од оваа формула, јасно е дека за поголем број s на блокови во кодниот збор веројатноста за пакет-грешка е поголема. Ова следува од тоа што во овој случај во процесот на декодирање има повеќе итерации и во секоја од нив потребно е точниот блок да биде во соодветно множество S со кандидати за декодирана порака. Кога пораката е подолга, за добивање на истата рата R, бројот на блокови во редундантната порака мора да биде поголем и оттука и кодниот збор е подолг отколку кај пократки пораки. Затоа во тој случај, бројот на итерации во процесот на декодирање е поголем, а со тоа е поголема и веројатност за пакет-грешка. Овие заклучоци се проверени и експериментално.

Во Поглавјето 2.4, дадени се експериментални резултати, добиени за различни патерни за додавање на редундантните симболи за кодот (72, 288) со рата $R = 1/4$ за бинарен симетричен канал. Од добиените резултатите заклучивме дека најдобри резултати се добиваат за третиот патерн, клуч со должина 10, квазигрупата дадена во Табела 2.1 (заедно со нејзината парастрофа) и $B_{max} = 4$. За споредба направени се експерименти со истата квазигрупа, почетен клуч и рата за кодот (144,576) со двапати подолги пораки. Во овие експерименти најдобри резултати се добиени со следниот патерн за редундантност:
1100 1100 1000 0000 1100 1000 1000 0000 1100 1100 1000 0000 1100 1000 1000 0000 0000 1100 1100 1000 0000 1100 1000 1000 0000 1100 1100 1000 0000 1100 1000 1000 0000 0000 0000 0000.

Добиените резултати за PER и BER за различни вредности на веројатноста p за бит-грешка на бинарен симетричен канал и $B_{max} = 4$ дадени се во Табела 2.12 и Табела 2.13.

Од резултатите прикажани во Табела 2.12 и Табела 2.13 може да се заклучи дека за кодот со подолги кодни зборови веројатностите за пакет-грешка и бит-грешка се приближно двапати поголеми отколку за кодот со пократки

кодни зборови. Во Табела 2.14 дадени се веројатностите $p_{повеќе}$ за појавување на *грешка-повеќе-кандидати* и веројатности $p_{празно}$ за појавување на *грешка-празно-множество* за овие два кода ($p_{повеќе} + p_{празно} = PER$).

p	$PER_s(72, 288)$	$PER_s(144, 576)$
0.03	0.00313	0.00343
0.04	0.00594	0.01200
0.05	0.01594	0.03200
0.06	0.03594	0.06800
0.07	0.06656	0.13200
0.08	0.11313	0.22114
0.09	0.18875	0.32571

Табела 2.12: Експериментални резултати за веројатноста за пакет-грешка

p	$BER_s(72, 288)$	$BER_s(144, 576)$
0.03	0.00136	0.00228
0.04	0.00343	0.00750
0.05	0.00928	0.01746
0.06	0.02239	0.04121
0.07	0.04065	0.08373
0.08	0.06485	0.14621
0.09	0.11357	0.21843

Табела 2.13: Експериментални резултати за веројатноста за бит-грешка

Од веројатностите дадени во Табела 2.14, може да се види дека за кодот (72,288) во процесот на декодирање се јавуваат повеќе неуспешни декодирања од типот *грешка-повеќе-кандидати*. Додека, во експериментите за кодот (144, 576), освен за $p = 0.05$, овој тип на грешки не се појавил. Причина

	$RCBQ(72, 288)$		$RCBQ(144, 576)$	
p	$p_{празно}$	$p_{повеќе}$	$p_{празно}$	$p_{повеќе}$
0.03	0.00219	0.00094	0.00343	0
0.04	0.00531	0.00063	0.01200	0
0.05	0.01469	0.00125	0.03143	0.00057
0.06	0.03594	0	0.06800	0
0.07	0.06500	0.00156	0.13200	0
0.08	0.11250	0.00063	0.22114	0
0.09	0.18688	0.00188	0.32571	0

Табела 2.14: Веројатности за *грешка-повеќе-кандидати* и *грешка-празно-множество*

за ова е помалиот број на редундантни нулти нибли во кодот (72,288). Затоа, кај овој код нема доволно редундантни нулти блокови на крајот на редундантната порака L кои се потребни за отфрлање на неточните блокови од множествата S во последните итерации од процесот на декодирање. Од друга страна, бројот на неуспешни декодирања со *грешка-празно-множество*, кои се предвидени во теориската веројатност за пакет-грешка, е приближно двапати помал за кодот со двапати помали кодни зборови.

Овие разлики во бројот на неуспешно декодирани пораки, посебно во бројот на *грешки-повеќе-кандидати*, се поголеми ако се земат уште пократки кодни зборови, на пример кодот (36,144). За овие пократки кодови бројот на *грешки-повеќе-кандидати* е уште поголем заради малиот број на редундантни нулти блокови во патерните за додавање на редундантните симболи.

За потврда на ова тврдење направени се експерименти и за кодот (36,144) со рата $R = 1/4$. Во експериментите користен е патернот: 1100 1100 1000 0000 1100 1100 0000 0000 0000, истиот почетен клуч $k = 0123456789$, вредност на $B_{max} = 4$ и истата квазигрупа, како и во претходните експерименти. Добиените резултати за PER, BER, $p_{празно}$ и $p_{повеќе}$ за различни вредности на веројатноста за бит-грешка p на бинарен симетричен канал дадени се во Табела 2.15 и Табела 2.16. Да забележиме дека за овој код, за $p > 0.05$ не важи равенството $p_{повеќе} + p_{празно} = PER$, бидејќи во овие експерименти се појавија неколку декодирања со *некоригирана-грешка* (декодирања кои завршуваат со еден кандидат во последната итерација, но добиената деко-

дирана порака не е точната).

Со споредба на резултатите за $p_{празно}$ и $p_{повеќе}$ дадени во Табела 2.14 и Табела 2.16 може да се види дека за најкраткиот код (36,144), веројатностите за појавување на *грешка-празно-множество* се најмали, но веројатностите за *грешка-повеќе-кандидати* се најголеми.

p	$PER_s(36,144)$	$BER_s(36,144)$
0.03	0.02057	0.01646
0.04	0.02286	0.01769
0.05	0.02543	0.02082
0.06	0.03471	0.02948
0.07	0.04814	0.04109
0.08	0.08071	0.07189
0.09	0.10757	0.09838

Табела 2.15: Експериментални резултати за PER и BER за кодот (36,144)

p	$p_{празно}$	$p_{повеќе}$
0.03	0.00086	0.01971
0.04	0.00257	0.02029
0.05	0.00743	0.01800
0.06	0.01814	0.01629
0.07	0.03186	0.01614
0.08	0.06057	0.01914
0.09	0.08971	0.01657

Табела 2.16: Веројатности за *грешка-повеќе-кандидати* и *грешка-празно-множество* за кодот (36,144)

Исто така, може да се забележи дека веројатностите $p_{празно}$ за кодот (36,144) се приближно двапати помали од веројатностите $p_{празно}$ добиени за кодот (72,288). Ова следува и од формулата за теориската веројатност за пакет-грешка во која се предвидени овој тип на грешки.

За $p < 0.06$, во експериментите со кодот (36,144), бројот на *грешки-повеќе-кандидати* е многу поголем од бројот на *грешки-празно-множество*. За овие вредности на p, вредностите за PER се полоши отколку кај кодот (72,288) (Табела 2.12 и Табела 2.15). Од друга страна, за кодот (36, 144) и $p > 0.06$, бројот на *грешки-празно-множество* е повеќе од двапати поголем од бројот на *грешки-повеќе-кандидати*. За овие p, добиените вредности за PER се подобри за кодот (36,144) отколку за кодот (72,288).

Со споредба на вредностите на $PER_s(36, 144)$ и $BER_s(36, 144)$ може да се забележи дека за овој код веројатностите за бит-грешка се поблиски до веројатностите за пакет-грешка отколку кај другите кодови. Причина за ова е малиот број на редундантни нули во првите блокови од патернот. Затоа пораките во последното непразно множество со кандидати за декодирање имаат мал заеднички префикс што доведува до поголеми вредности на бит-грешката.

Заклучок

Во оваа глава детално се испитани својствата и перформансите на случајните кодови базирани на квазигрупи и начинот на кој секој од параметрите во дизајнот на кодовите влијае над текот на процесот на декодирање и бројот на успешно завршени декодирања. Дадени се идеи и заклучоци за изборот на овие параметри и начини како да се подобрат перформансите. Исто така, дефиниран е нов метод за намалување на неуспешните декодирања кои завршуваат предвреме.

Глава 3

Алгоритам-за-декодирање-со-пресек

Случајните кодови базирани на квазигрупи имаат неколку параметри и во претходната глава ние испитавме како изборот на параметрите влијае на неговите перформанси. Од направените експерименти заклучивме дека брзината на декодирање е еден од најголемите проблеми кај овие кодови. Зависно од избраниот патерн за додавање на редундантните нулти симболи, процесот на декодирање е спор во некои итерации, бидејќи бројот на елементите во множествата S е многу голем. Ако редундантните нули се распоредат порамномерно, тогаш бројот на елементи во множествата S нема да биде многу голем, но ќе се појават многу неуспешни декодирања со *грешка-повеќе-кандидати*. Затоа во патерните треба да има повеќе нулти симболи на крајот. Всушност, многу е тешко да се најде добра рамнотежа за поставување на редундантните нули, но експериментално имаме најдено неколку доволно задоволителни патерни.

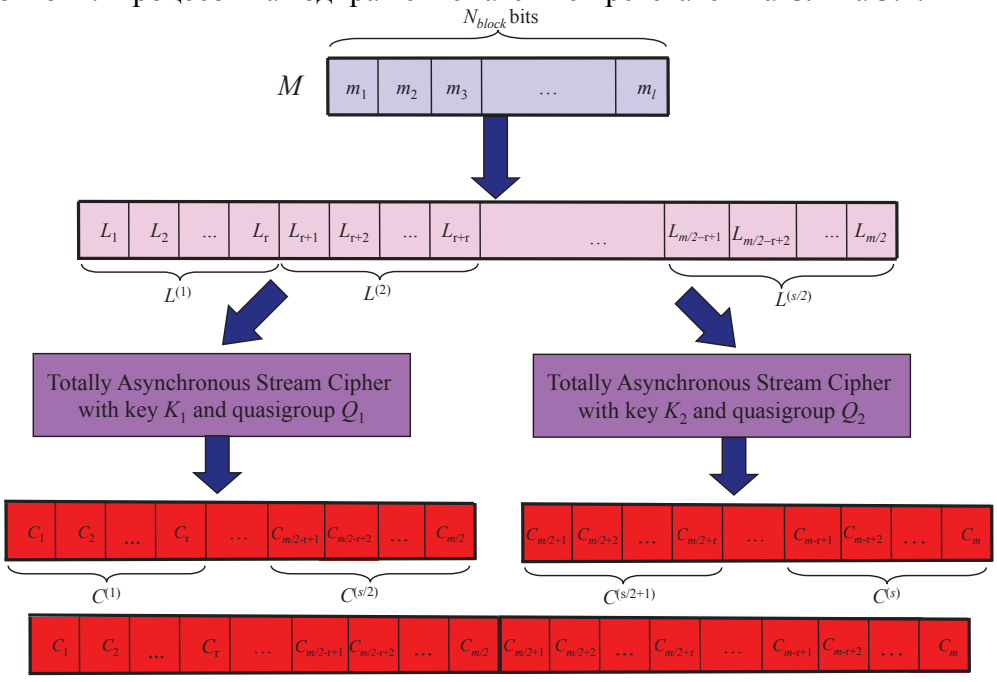
Со цел да ја подобриме брзината на декодирање, ние дефинираме нов алгоритам за кодирање/декодирање наречен алгоритам-за-декодирање-со-пресек (Cut-Decoding или АДП алгоритам). Бидејќи декодирањето на RCBQ е всушност декодирање со листа (list decoding), брзината на декодирањето и веројатноста за точно декодирање зависи од големината на листите со можни кандидати за декодираната порака. Затоа, во новиот АДП алгоритам кој го предлагаме, 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 ал-

горитам. Исто така, во оваа глава ќе бидат разгледани и некои други модификации на алгоритмот за декодирање со цел да се намали веројатноста за пакет-грешка (PER) и веројатноста за бит-грешка (BER).

3.1 Опис на кодирањето со АДП алгоритамот

Во АДП алгоритмот, наместо код (N_{block}, N) со рата R, се користат два $(N_{block}, N/2)$ кодови со рата $2R$, со кои се кодира/декодира иста порака од N_{block} битови.

Прво, во пораката $M = m_1 m_2 ... m_l$ се додаваат редундантните 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 $L_i \in Q$. За добивање на кодирана порака алгоритмот за криптирање, даден на Слика 2.2, се применува двапати на истата редундантна порака L со користење на различни параметри (различни клучеви или квазигрупи). На овој начин, кодниот збор за пораката се добива со конкатенација на двата кодни збора од $N/2$ битови. Процесот на кодирање шематски е претставен на Слика 3.1.



Слика 3.1: Кодирање со АДП алгоритамот

3.2 Опис на декодирањето со АДП алгоритамот

После преносот низ бинарен симетричен канал, излезната порака $D = D^{(1)}D^{(2)}...D^{(s)}$, каде што $D^{(i)}$ се подблокови од r симболи од азбуката Q, 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 и овие две пораки се декодираат паралелно со соодветните параметри. Во АДП алгоритамот е направена модификација во делот (iii) од процесот на декодирање, т.е., во процедурата за генерирање на множествата со кандидати за декодирање. Во новиот алгоритам овие множества се генерираат на следниот начин.

Чекор 1. Нека $S_0^{(1)} = (k_1^{(1)} \dots k_n^{(1)}; \lambda)$ и $S_0^{(2)} = (k_1^{(2)} \dots k_n^{(2)}; \lambda)$, каде λ е празен стринг, $k_1 = k_1^{(1)} \dots k_n^{(1)}$ и $k_2 = k_1^{(2)} \dots k_n^{(2)}$ се почетните клучеви кои се користени во формирањето на двата кодни зборови.

Чекор 2. Нека $S_{i-1}^{(1)}$ и $S_{i-1}^{(2)}$ се дефинирани за $i \geqslant 1$.

Чекор 3. Нека во двата 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.

Чекор 4. Нека $V_1 = \{w_1w_2 \dots w_{rai} | (\delta, w_1w_2 \dots w_{rai}) \in S_i^{(1)}\}$,
$V_2 = \{w_1w_2 \dots w_{rai} | (\delta, w_1w_2 \dots w_{rai}) \in S_i^{(2)}\}$ и $V = V_1 \cap V_2$.

Чекор 5. За секој елемент $(\delta, w_1w_2 \dots w_{rai}) \in S_i^{(1)}$, ако $w_1w_2 \dots w_{rai} \notin V$ тогаш $S_i^{(1)} \leftarrow S_i^{(1)} \setminus \{(\delta, w_1w_2 \dots w_{rai})\}$.
Исто така, за секој $(\delta, w_1w_2 \dots w_{rai}) \in S_i^{(2)}$, ако $w_1w_2 \dots w_{rai} \notin V$ тогаш $S_i^{(2)} \leftarrow S_i^{(2)} \setminus \{(\delta, w_1w_2 \dots w_{rai})\}$.

(* Всушност, 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 $S_i^{(2)}$. *)

Чекор 6. Ако $i < s/2$ тогаш зголеми го i и врати се на Чекор 3.

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 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*.

- 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*.

Во експериментите со новиот алгоритам за декодирање забележавме значително намалување на бројот на елементи во множествата S и постигнавме големо подобрување на брзината на процесот на декодирање. Имено, новиот метод на декодирање е 4.5 пати побрз за кодот (72,288). Во Табела 3.1 даден е еден пример на просечниот број на елементи во множествата $S_i^{(1)}$ и $S_i^{(2)}$, во секоја итерација, пред и после редукцијата во **Чекор 4** и **Чекор 5** од новиот метод за декодирање. Овие просеци се добиени за 1000 декодирања. Од резултатите во Табела 3.1 може да се види дека предложената редукција значително го намалува бројот на елементи во множествата со кандидати за декодирање (после втората итерација бројот на елементи во редуцираните множества е приближно 20 пати помал).

Проблемот кој се јавува во АДП алгоритамот е што за добивање на код со рата R потребен е патерн за код со двапати поголема рата. Но, за поголеми рати тешко е да се направи добар патерн, бидејќи во овие патерни бројот на редундантни нули е помал. Затоа, со овој метод за декодирање се добиваат полоши резултати за бројот на неуспешни декодирања од тип *грешка-повеќе-кандидати*, но бројот на неуспешни декодирања со *грешка-празно-множество* е помал.

За решавање на проблемот со поголемиот број на неуспешни декодирања со *грешка-повеќе-кандидати* предложена е една хевристика во правилото за

Бр. на итерација	Бр. на елементи во $S_i^{(1)}$ (пред редукција)	Бр. на елементи во $S_i^{(2)}$ (пред редукција)	Бр. на елементи во редуцираните $S_i^{(1)}$, $S_i^{(2)}$
1	10.4	9.9	1.6
2	244.2	244.2	18.2
3	178.5	181.9	8.1
4	79.7	80.5	4.4
5	693.5	695.1	34.8
6	343.9	339.5	14.2
7	140.6	139.3	6.2
8	60.8	61.5	3.4
9	1.1	1.0	1.0

Табела 3.1: Просечен број на елементи во множествата со кандидати за декодирање пред и после редукцијата

декодирање за елиминација на овој тип грешки. 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 (како втора компонента на некој елемент во двете множества). Затоа, во овој случај може случајно да се избере една порака од едно од множествата во последната итерација и таа порака да биде декодираната порака. Ако се избере точната порака, тогаш бит-грешката е 0 и затоа на овој начин и BER се намалува. Во експериментите направени со оваа модификација добивме дека во приближно половина од случаите се избира точната порака.

3.3 Споредба на стандардниот и АДП алгоритамот за рата $R = 1/4$

Во Поглавје 2.4 дадени се експериментални резултати за стандардниот алгоритам на RCBQ за кодови (72,288) со рата $R = 1/4$. Од резултатите доби-

ени за различни патерни за додавање на редундансата, различни должини на почетниот клуч и различни квазигрупи, заклучивме дека најдобрите резултати за овие кодови се добиени за следните параметри:

- третиот патерн за редундантност: 1100 1100 1000 0000 1100 1000 1000 0000 1100 1100 1000 0000 1100 1000 1000 0000 0000 0000,
- клучот $k = 0123456789$ од 10 нибли, и
- квазигрупата дадена во Табела 2.1.

Затоа, резултатите добиени со овие параметри ќе ги споредуваме со соодветните резултати за новиот АДП алгоритам. Направени се експерименти со користење на новиот алгоритам за кодови (72,288) над азбуката од нибли и при пренос низ бинарен симетричен канал. Во експериментите со АДП алгоритамот разгледани се 17 различни патерни за додавање на редундантните симболи за код (72, 144) со рата $R = 1/2$, различни должини на почетните клучеви и различни квазигрупи од ред 16.

Исто така, разгледани се и разликите во перформансите на кодовите кога во двата процеси на кодирање/декодирање во АДП алгоритамот, се користат само различни клучеви или различни клучеви и различни квазигрупи. Заради редукцијата на елементите во множествата со кандидати за декодирање (воведена во **Чекор 4** и **Чекор 5**), јасно е дека патернот за редундантноста во АДП треба да биде ист во двата паралелни процеси на кодирање/декодирање. Имено, ако се користат различни патерни, тогаш не може да се бара пресек на множествата S во **Чекор 4**. Тука ќе бидат дадени само најдобрите добиени резултати.

3.3.1 Експерименти со различни клучеви

Прво, направени се експерименти со користење само на различни клучеви во двата процеси на кодирање/декодирање и иста квазигрупа. Како што е споменато погоре, разгледани се 17 различни патерни за додавање на редундантните симболи. Најдобрите резултати се добиени за следните параметри:

- патернот за редундантност: 1100 1110 1100 1100 1110 1100 1100 1100 0000,
- два различни клучеви од 5 нибли: $k_1 = 01234$ и $k_2 = 56789$, и

– квазигрупата дадена во Табела 2.1.

Исто така, направени се експерименти и со клучеви со должина од 10 нибли, но резултатите беа слични.

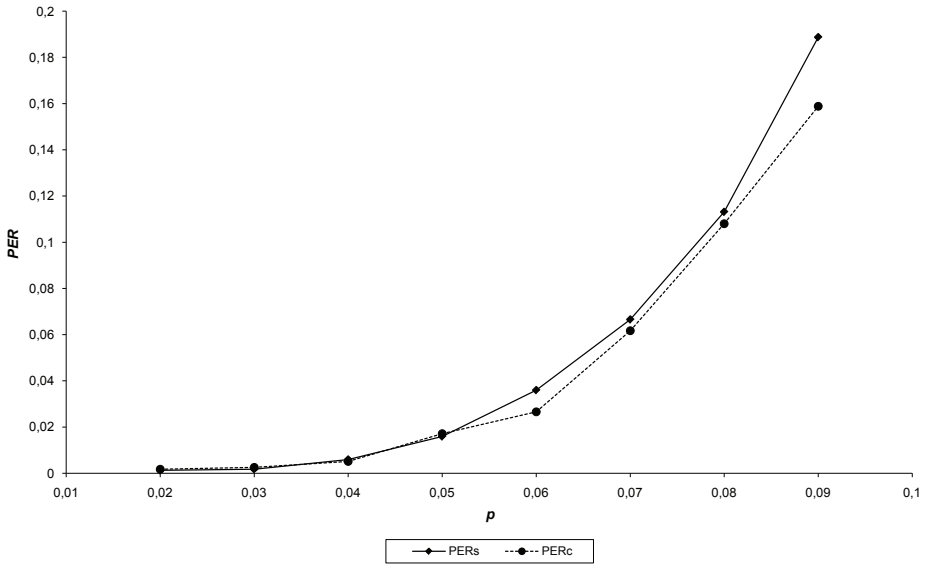
Нека PER_c е веројатноста за пакет-грешка и BER_c веројатноста за бит-грешка добиени со новиот АДП алгоритам. Како и претходно, PER_s и BER_s се соодветните веројатности за стандардниот метод за кодирање/декодирање на RCBQ. Резултатите за PER_s и PER_c (BER_s и BER_c), за различни вредности на веројатноста p за бит-грешка на бинарен симетричен канал и $B_{max} = 4$, дадени се во Табела 3.2 (Табела 3.3) и претставени на Слика 3.2 (Слика 3.3).

Од резултатите дадени во Табела 3.2 и Табела 3.3, може да се заклучи дека со двата алгоритми за декодирање се добиваат приближно исти резултати за вредностите на PER и BER. Но, може да се забележи дека за $p > 0.05$ со новиот АДП алгоритам се добиваат малку подобри вредности на PER.

Од резултатите добиени за BER може да се забележи дека за сите вредности на p разликите помеѓу BER_s и BER_c се многу мали. За некои вредности на p има малку подобри резултати за BER со стандардниот алгоритам, а за други p со новиот алгоритам.

p	PER_s	PER_c
0.02	0.00125	0.00171
0.03	0.00313	0.00257
0.04	0.00594	0.00514
0.05	0.01594	0.01714
0.06	0.03594	0.02657
0.07	0.06656	0.06171
0.08	0.11313	0.10800
0.09	0.18875	0.15886

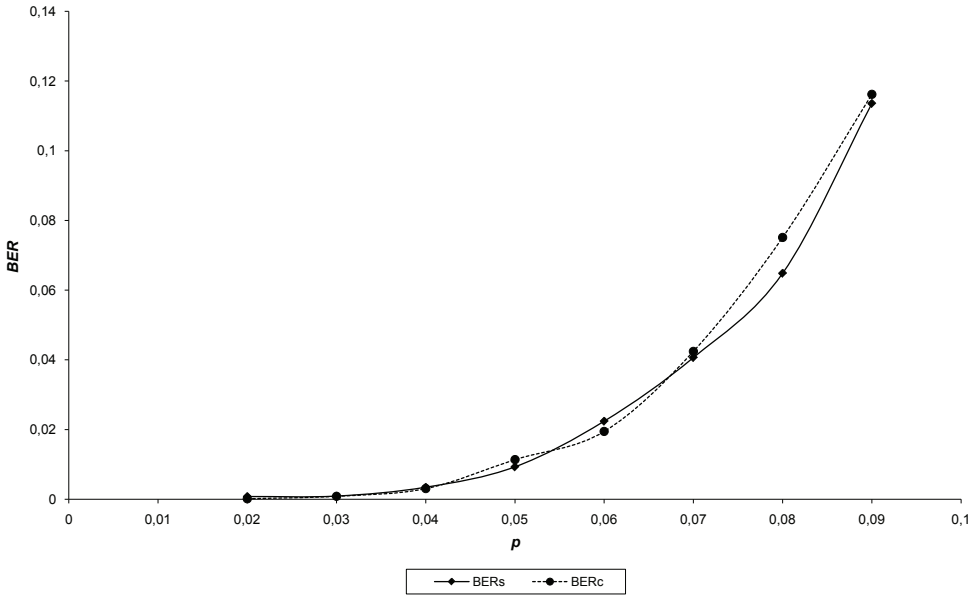
Табела 3.2: Експериментални резултати за веројатноста за пакет-грешка



Слика 3.2: Споредба на PER

p	BER_s	BER_c
0.02	0.00076	0.00011
0.03	0.00136	0.00083
0.04	0.00343	0.00302
0.05	0.00928	0.01134
0.06	0.02239	0.01943
0.07	0.04065	0.04243
0.08	0.06485	0.07512
0.09	0.11357	0.11621

Табела 3.3: Експериментални резултати за веројатноста за бит-грешка



Слика 3.3: Споредба на BER

3.3.2 Експерименти со различни клучеви и различни квазигрупи

Направени се експерименти со новиот метод за декодирање за код (72,288) со користење на два кода (72,144) со различни клучеви и различни квазигрупи. Прво, направени се експерименти со користење на квазигрупата дадена во Табела 2.1 и квазигрупата со висок коефициент на растење на периодата дадена на крајот од трудот [15]. Во овие експерименти користен е истиот патерн за рата $R = 1/2$ и $B_{max} = 4$. Во овој случај со користење на два различни клучеви од 10 нибли ($k_1 = 0123456789$ и $k_2 = 5432897610$) добиени се малку подобри резултати отколку со користење на клучеви од 5 нибли. Во Табела 3.4 претставени се резултати за веројатноста за пакет-грешка PER_{c_2} и веројатноста за бит-грешка BER_{c_2} добиени за овие параметри.

Со споредба на резултатите во Табела 3.2 и Табела 3.3 со резултатите во Табела 3.4 може да се заклучи дека вредностите на PER_{c_2} и BER_{c_2} се слични на претходните резултати добиени со користење на иста квазигрупа

p	PER_{c_2}	BER_{c_2}
0.02	0.00143	0.00053
0.03	0.00314	0.00150
0.04	0.00571	0.00299
0.05	0.01400	0.01013
0.06	0.03000	0.02183
0.07	0.06314	0.04544
0.08	0.09857	0.07198
0.09	0.16086	0.11839

Табела 3.4: Експериментални резултати за веројатностите за пакет-грешка и бит-грешка за различни квазигрупи и различни клучеви

во двата процеси. Исто така, направени се експерименти и со користење на две различни квазигрупи и ист клуч во двата процеси на кодирање/декодирање, но не се добиени подобри резултати.

Меѓутоа, ако во еден од кодовите се користи цикличната квазигрупа од ред 16, тогаш резултатите за PER и BER се многу полоши. На пример, за $p = 0.02$ добиено е $PER = 0.18314$ и $BER = 0.05432$.

3.4 Метод за намалување на *грешки-празно-множество* во АДП алгоритмот

Во Поглавје 2.5 дефиниран е метод со враќање за намалување на бројот на неуспешни декодирања со *грешка-празно-множество*. Покажано е дека во RCBQ со стандардните алгоритми, некои од овие грешки ќе бидат елиминирани, ако се поништат неколку итерации од процесот на декодирање и дел од нив се извршат со поголема вредност на B_{max}.

Во ова поглавје ќе биде разгледано користењето на оваа идеја кога во новиот АДП алгоритам ќе се појави *грешка-празно-множество* (т.е. кога двете редуцирани множества се празни). Направени се експерименти со враќање назад две или три итерации и со користење на $B_{max} + 1$ или $B_{max} + 2$ во првата поништена итерација (останатите итерации ја користат претходната

вредност на B_{max}). Мора да напоменеме дека со ова враќање назад во некои случаи наместо *грешка-празно-множество* се добива *грешка-повеќе-кандидати*.

Најдобри резултати се добиени со враќање назад две итерации и $B_{max}+2$ во првата поништена итерација. Во Табела 3.5 дадени се процентите на елиминирани неуспешни декодирања со *грешка-празно-множество*. Овие резултати се добиени со примена на оваа модификација на неуспешните декодирања со *грешка-празно-множество* во експериментите чии резултати се дадени во Табела 3.2 и Табела 3.3. Може да се заклучи дека оваа модификација дава добри проценти на елиминирани *грешки-празно-множество* во АДП алгоритмот.

p	Процент на елиминирани *грешки-празно-множество*
0.03	28.57%
0.04	20.37%
0.05	24.00%
0.06	23.17%
0.07	27.23%
0.08	25.37%
0.09	22.64%

Табела 3.5: Процент на елиминирани неуспешни декодирања со *грешка-празно-множество*

Многу важно за оваа модификација е што со неа се добива само значително мало намалување на брзината на декодирање. Исто така, со елиминирање на некои од неуспешните декодирања со *грешка-празно-множество* се добиваат и подобри резултати за BER.

Во Табела 3.6 и Табела 3.7 споредени се вредностите на PER_c и BER_c од Табела 3.2 и Табела 3.3, добиени со АДП алгоритмот без враќање и вредностите на PER_{c_back} и BER_{c_back} добиени со користење на методот со враќање (две итерации назад и $B_{max}+2$).

Од резултатите во Табела 3.6 и Табела 3.7, може да се заклучи дека за поголеми вредности на p се добиваат поголеми подобрувања за PER. Исто

p	PER_c	PER_{c_back}
0.03	0.00257	0.00257
0.04	0.00514	0.00514
0.05	0.01714	0.01429
0.06	0.02657	0.02171
0.07	0.06171	0.04857
0.08	0.10800	0.08200
0.09	0.15886	0.12543

Табела 3.6: Експериментални резултати за PER со и без враќање

p	BER_c	BER_{c_back}
0.03	0.00083	0.00077
0.04	0.00302	0.00165
0.05	0.01134	0.00510
0.06	0.01943	0.00892
0.07	0.04243	0.02194
0.08	0.07512	0.03795
0.09	0.11621	0.05383

Табела 3.7: Експериментални резултати за BER со и без враќање

така, со оваа модификација за сите вредности на p постигнати се приближно двапати подобри резултати за BER. Ова се случува бидејќи во случаите кога со враќањето нема да биде елиминирано неуспешното декодирање со *грешка-празно-множество*, празните редуцирани множества може да се појават во некоја подоцнежна итерација, така што поголем дел од пораката ќе биде декодиран и веројатноста за бит-грешка ќе биде помала.

Направени се експерименти и со користење на оваа модификација во случаите кога само едно од множествата $S_i^{(1)}$ или $S_i^{(2)}$ е празно во некоја итерација. Но, во овие експерименти не се добиени подобри резултати од прет-

ходно кога едниот процес на декодирање продолжува ако другото множество е празно. Во тој случај, декодирањето често завршува успешно, посебно ако празното множество се појави во некоја од последните итерации.

3.5 Метод за намалување на *грешки-повеќе-кандидати*

Како што е споменато погоре, во првите експерименти со АДП алгоритмот, добиени се полоши резултати во бројот на неуспешни декодирања со *грешка-повеќе-кандидати*, но бројот на неуспешни декодирања со *грешка-празно-множество* беше помал. За да се разреши овој проблем со поголем број на *грешки-повеќе-кандидати*, воведена е една хевристика во правилото за декодирање за елиминирање на овој тип на грешки. Во таа хевристика случајно се одбира еден елемент од редуцираните множества со кандидати за декодирање во последната итерација и неговата втора компонента се зема за декодирана порака. Исто таа, од резултатите во претходното поглавје може да се види дека со примена на методот со враќање за намалување на бројот на *грешки-празно-множество* во АДП алгоритмот, добиени се добри подобрувања на PER and BER.

Во ова поглавје, предложена е слична модификација со враќање во АДП алгоритмот, 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}.

Направени се експерименти со користење на оваа модификација во АДП алгоритмот за кодот (72, 288) со параметрите кои дадоа најдобри резултати за овој алгоритам, т.е., патернот: 1100 1110 1100 1100 1110 1100 1100 1100 0000, клучевите $k_1 = 01234$, $k_2 = 56789$ и квазигрупата дадена во Табела 2.1. Разгледани се подобрувањата на веројатностите за успешно декодирање добиени со овој метод користејќи различен број на поништени итерации и $B_{max} - 1$ или $B_{max} - 2$ во првата поништена итерација.

Најдобри резултати се добиени ако се поништат последните две итерации и се користи $B_{max} - 1$ во првата од поништените итерации. Во Табела 3.8

и Табела 3.9 споредени се вредностите на PER_c и BER_c (од Табела 3.2 и Табела 3.3) добиени без враќање и вредностите на $PER_{c\_back\_more}$ и $BER_{c\_back\_more}$ добиени со враќање за *грешка-повеќе-кандидати*.

p	PER_c	$PER_{c\_back\_more}$
0.02	0.00171	0.00029
0.03	0.00257	0.00086
0.04	0.00514	0.00343
0.05	0.01714	0.01543
0.06	0.02657	0.02600
0.07	0.06171	0.05971
0.08	0.10800	0.10543
0.09	0.15886	0.15743

Табела 3.8: Експериментални резултати за PER без и со враќање за *грешка-повеќе-кандидати*

p	BER_c	$BER_{c\_back\_more}$
0.02	0.00011	0.00001
0.03	0.00083	0.00051
0.04	0.00302	0.00253
0.05	0.01134	0.01085
0.06	0.01943	0.01932
0.07	0.04243	0.04208
0.08	0.07512	0.07495
0.09	0.11621	0.11583

Табела 3.9: Експериментални резултати за BER без и со враќање за *грешка-повеќе-кандидати*

Од резултатите во Табела 3.8 и Табела 3.9 може да се види дека со користење на оваа модификација за намалување на бројот на неуспешни декоди-

рања со *грешка-повеќе-кандидати* добиени се подобрувања на вредностите на PER и BER, за сите вредности на p.

Подобрувањата на вредностите на веројатностите за пакет-грешка и бит-грешка добиени со двата методи за намалување на грешките со враќање, ни дадоа идеја за користење на комбинација од овие два методи. Затоа, направени се експерименти со користење на двата методи со враќање, за *грешка-празно-множество* и за *грешка-повеќе-кандидати*. Во овие експерименти, ако се добие *грешка-празно-множество*, т.е., празни множества во некоја итерација од процесот на декодирање, тогаш се поништуваат две итерации и првата од поништените итерации се извршува со користење на $B_{max} + 2 = 6$. Од друга страна, ако процесот на декодирање заврши со повеќе елементи во последните множества $S_{s/2}^{(1)}$ и $S_{s/2}^{(2)}$, тогаш процесот се враќа две итерации назад и $(s/2 - 1)$-та итерација се извршува со $B_{max} - 1 = 3$. Во еден процес на декодирање се прави само едно враќање, бидејќи кога се прават повеќе враќања се добива многу голема кардиналност на множествата S во одредена итерација. Исклучок за ова правило е следниот случај. Ако после враќањето за *грешка-празно-множество* се добијат повеќе кандидати во последната итерација, тогаш се прави уште едно враќање за *грешка-повеќе-кандидати*.

Нека со $PER_{c\_back\_2}$ и $BER_{c\_back\_2}$ се означени веројатностите за пакет-грешка и бит-грешка добиени во тој случај. Во Табела 3.10 споредени се PER_c, PER_{c_back}, $PER_{c\_back\_more}$ и $PER_{c\_back\_2}$, а во Табела 3.11 вредностите на BER_c, BER_{c_back}, $BER_{c\_back\_more}$ and $BER_{c\_back\_2}$.

Од резултатите во Табела 3.10 и Табела 3.11, може да се види дека за помали вредности на p, подобри резултати се добиваат со враќањето за *грешка-повеќе-кандидати*, затоа што за помали p поголем број од неуспешните декодирања завршуваат со овој тип на грешка. Од друга страна, за поголеми вредности на p бројот на неуспешни декодирања со *грешка-празно-множество* е поголем. Оттука, за овие вредности на p, подобри подобрувања се добиваат со користење на методот со враќање за *грешка-празно-множество*. Исто така, може да се заклучи дека со предложената комбинација на методите со враќање за двата типа на грешки се добиваат најдобри подобрувања на вредностите за PER и BER, за сите p. Освен тоа, вредности на $BER_{c\_back\_2}$ се повеќе од двапати помали од BER_c.

p	PER_c	PER_{c_back}	$PER_{c\_back\_more}$	$PER_{c\_back\_2}$
0.02	0.00171	0.00171	0.00029	0.00029
0.03	0.00257	0.00257	0.00086	0.00029
0.04	0.00514	0.00514	0.00343	0.00314
0.05	0.01714	0.01429	0.01543	0.01200
0.06	0.02657	0.02171	0.02600	0.02000
0.07	0.06171	0.04857	0.05971	0.04486
0.08	0.10800	0.08200	0.10543	0.08114
0.09	0.15886	0.12543	0.15743	0.12486

Табела 3.10: Експериментални резултати за PER без и со методите со враќање

p	BER_c	BER_{c_back}	$BER_{c\_back\_more}$	$BER_{c\_back\_2}$
0.02	0.00011	0.00011	0.00001	0.00001
0.03	0.00083	0.00077	0.00051	0.00024
0.04	0.00302	0.00165	0.00253	0.00142
0.05	0.01134	0.00510	0.01085	0.00507
0.06	0.01943	0.00892	0.01932	0.00869
0.07	0.04243	0.02194	0.04208	0.02017
0.08	0.07512	0.03795	0.07495	0.03459
0.09	0.11621	0.05383	0.11583	0.05378

Табела 3.11: Експериментални резултати за BER без и со методите со враќање

3.6 АДП алгоритам со подолги пораки

Во Поглавје 2.6 испитано е влијанието на должината на пораките на перформансите на стандардните кодовите кои поправаат грешки базирани на квазигрупи. Таму е покажано дека веројатностите за пакет-грешка и бит-грешка за

кодот (144, 576), со подолги пораки и кодни зборови, се приближно двапати поголеми отколку за кодот (72, 288). Исто така, забележано е дека во експериментите за кодот (72,288) се добиваат повеќе неуспешни декодирања со *грешка-повеќе-кандидати*. Додека, во експериментите за кодот (144,576) овие грешки не се појавуваат (освен за $p = 0.05$). Во ова поглавје, ќе биде испитано дали ова се случува и во експериментите со новиот АДП алгоритам.

Во Поглавје 3.3 дадени се експерименталните резултати добиени со АДП алгоритмот за кодот (72,288) со рата $R = 1/4$. За споредба направени се експерименти со истата квазигрупа, клучевите од 5 нибли и истата рата за кодот (144, 576) со двапати подолги пораки и кодни зборови. Најдобри резултати со АДП алгоритмот за код (144, 576) добиени се за патернот: 1100 1110 1100 1100 1110 1100 1100 1110 1100 1100 1110 1100 1100 1110 1100 1000 0000 0000. Во експериментите со код (144,576) забележано е дека не се појавуваат неуспешни декодирања со *грешка-повеќе-кандидати*. Но, исто како и со стандардниот алгоритам, бројот на неуспешни декодирања со *грешка-празно-множество* е поголем.

p	$PER_c(72, 288)$	$PER_c(144, 576)$
0.02	0.00171	0.00000
0.03	0.00257	0.00114
0.04	0.00514	0.00800
0.05	0.01714	0.02914
0.06	0.02657	0.05657
0.07	0.06171	0.11029
0.08	0.10800	0.17886
0.09	0.15886	0.28286

Табела 3.12: Експериментални резултати за веројатноста за пакет-грешка

Од резултатите во Табела 3.12 и Табела 3.13 може да се заклучи дека, исто како и со стандардниот алгоритам, веројатностите за пакет-грешка и бит-грешка за кодот со подолги пораки и кодни зборови се поголеми отколку за кодот со пократки. Ова е не точно само за $p = 0.02$ и $p = 0.03$, бидејќи

p	$BER_c(72, 288)$	$BER_c(144, 576)$
0.02	0.00011	0.00000
0.03	0.00083	0.00080
0.04	0.00302	0.00621
0.05	0.01134	0.02186
0.06	0.01943	0.04516
0.07	0.04243	0.08534
0.08	0.07512	0.13512
0.09	0.11621	0.22163

Табела 3.13: Експериментални резултати за веројатноста за бит-грешка

за овие вредности на p бројот на *грешки-празно-множество* е многу мал. Имено, за $p = 0.02$ и $p = 0.03$, бројот на *грешки-празно-множество* е ист за двата кода (за $p = 0.02$ бројот на овие грешки е 0), но за кодот (72,288) има и неколку неуспешни декодирања со *грешка-повеќе-кандидати*. Во експериментите со кодот (144,576), добиено е дека за овој код веројатноста за појавување на *грешка-повеќе-кандидати* е нула за сите вредности на p.

3.7 Експерименти со квазигрупи од ред 4 и ред 256

Во ова поглавје ќе бидат испитани перформансите на случајните кодови базирани на квазигрупи кога во процесите на кодирање и декодирање се користат квазигрупи од ред 4 и ред 256. Тогаш пораките и кодните зборови се стрингови од 2-битни симболи или 8-битни симболи (бајти), соодветно. Направени се неколку експерименти со стандардниот метод на кодирање/декодирање и со АДП алгоритмот дефиниран погоре.

Во Поглавје 2.4 испитани се перформансите за код (72,288) со стандардните алгоритми, за бинарен симетричен канал. Тие експерименти се направени со користење на азбуката $Q = \{0, 1, \ldots 9, a, b, c, d, e, f\}$ од нибли и различни квазигрупи од ред 16. Најдобри резултати се добиени за квазигрупата дадена во Табела 2.1, клуч од 10 нибли и патернот за редундантност: 1100 1100 1000 0000 1100 1000 1000 0000 1100 1100 1000 0000 1100 1000 1000

0000 0000 0000. Во Поглавје 3.3 овие резултати се споредени со најдобрите резултати добиени со АДП алгоритмот за азбука од нибли.

Сега, направени се експерименти со двата алгоритми (стандарден и АДП) во кои се користат квазигрупи од ред 4 и ред 256, наместо квазигрупи од ред 16. Направени се симулации за бинарен симетричен канал со различни патерни за додавање на редундансата, различни клучеви, различни должини на блоковите во процесот на декодирање и неколку квазигрупи од ред 4 и ред 256. Во експерименти разгледувани се веројатностите за пакет-грешка, веројатностите за бит-грешка и бројот на неуспешни декодирања со *грешка-празно-множество* и *грешка-повеќе-кандидати*.

3.7.1 Експерименти со квазигрупи од ред 4

Користејќи графички приказ (image pattern), Димитрова и Марковски, во [17], даваат класификација на квазигрупите од ред 4 на фрактални и нефрактални. Во [21] авторите даваат класификација на овие квазигрупи на линеарни и нелинеарни по Булова презентација. Ние ги испитавме перформансите на RCBQ со пораки од 2-битни симболи за квазигрупи од различни класи на овие класификации. Направени се експерименти со користење на фрактални квазигрупи; нефрактални и слабо нелинеарни; и нефрактални и чисто нелинеарни квазигрупи.

Во експериментите со квазигрупи од ред 4, најлоши резултати се добиваат кога се користат фрактални квазигрупи. Имено, ако во алгоритмите за криптирање/декриптирање (дадени на Слика 2.2) се користи фрактална квазигрупа, тогаш се добиваат многу неуспешни декодирања со *грешка-повеќе-кандидати* дури и за $B_{max} = 3$. Од друга страна, во експериментите со нефрактални и слабо нелинеарни и нефрактални и чисто нелинеарни квазигрупи, вредностите на веројатноста за пакет-грешка и веројатноста за бит-грешка се слични, но малку подобри за нефрактални и слабо нелинеарни квазигрупи.

Иако кодовите со нибли и кодовите со 2-битни симболи се два различни кода, во ова поглавје ќе бидат споредени резултатите добиени за код (72,288) со рата $1/4$ за двете азбуки. За $B_{max} = 3$, подобри резултати се добиваат во експериментите со 2-битни симболи споредено со експериментите со нибли. Но, за поголеми вредности на B_{max}, во сите експерименти со 2-битни симболи (со стандарден алгоритам и со АДП алгоритам) се добиваат многу неуспешни декодирања со *грешка-повеќе-кандидати*.

Најдобри резултати за код (72,288) со користење на азбуката од 2-битни симболи и стандардниот алгоритам добиени се за следните параметри:

– патернот за редундантност: 111000 111000 110000 000000 111000 110000 110000 000000 110000 110000 000000 000000 111000 111000 110000 000000 111000 110000 110000 000000 110000 000000 000000 000000,

– клучот $k = 012301230123213023103210$ од 24 симболи, и

– квазигрупата (3.1) (нефрактална и слабо нелинеарна квазигрупа).

$*$	0	1	2	3
0	0	2	1	3
1	1	3	2	0
2	2	0	3	1
3	3	1	0	2

(3.1)

Во Табела 3.14 и Табела 3.15 споредени се, за различни вредности на веројатноста p за бит-грешка на бинарен симетричен канал, најдобрите резултати на PER и BER за двата кода (72,288) (со 2-битни симболи и нибли) со користење на стандардниот алгоритам и $B_{max} = 3$. Во табелите PER_{s_2} и BER_{s_2} се веројатностите за пакет-грешка и бит-грешка за кодот со 2-битни симболи, а PER_s и BER_s за кодот со нибли (од Табела 2.10 за $B_{max} = 3$) .

p	PER_{s_2}	PER_s
0.02	0.00171	0.00475
0.03	0.00849	0.01843
0.04	0.02429	0.05559
0.05	0.05429	0.11758
0.06	0.09886	0.21314
0.07	0.15743	0.32971

Табела 3.14: Експериментални резултати за веројатноста за пакет-грешка за $B_{max} = 3$

p	BER_{s_2}	BER_s
0.02	0.00121	0.00209
0.03	0.00486	0.00849
0.04	0.01491	0.02629
0.05	0.03467	0.05488
0.06	0.05917	0.10655
0.07	0.09941	0.16738

Табела 3.15: Експериментални резултати за веројатноста за бит-грешка за $B_{max} = 3$

Од резултатите во Табела 3.14 може да се заклучи дека за $B_{max} = 3$ вредностите на PER_{s_2} се приближно двапати подобри од вредностите на PER_s. Истиот заклучок е точен и за вредностите на BER_{s_2} и BER_s, дадени во Табела 3.15. Подобрите резултати за бројот на *грешки-празно-множество* за кодот со 2-битни симболи следуваат и од формулата за теориската веројатност за пакет-грешка дадена во Теорема 2.1. Во оваа теорема се предвидени само овој тип на грешки. Но, во експериментите со кодот со 2-битни симболи се добиваат неколку неуспешни декодирања со *грешка-повеќе-кандидати*, дури и за $B_{max} = 3$. Затоа, експерименталните веројатности PER_{s_2} се поголеми од теориските PER_t дадени во Теорема 2.1.

Направени се експерименти и со користење на АДП алгоритмот за код (72,288) со 2-битни симболи. Во овие експерименти најдобри резултати се добиени со користење на следните параметри во дизајнот на кодот:

- патернот за редундантност: 111100 111100 111000 111000 111100 111000 111000 111100 111000 111000 110000 000000,
- два различни клучеви $k_1 = 012301230123213023103210$ и $k_2 = 321023102130012301230123$ од 24 симболи, и
- квазигрупата (3.1).

Резултатите добиени за веројатностите за пакет-грешка и бит-грешка за различни вредности на веројатноста p за бит-грешка на бинарен симетричен канал и $B_{max} = 3$ дадени се во Табела 3.16.

p	PER_{c_2}	BER_{c_2}
0.02	0.00114	0.00059
0.03	0.00714	0.00354
0.04	0.02057	0.01232
0.05	0.04886	0.02965
0.06	0.09200	0.05493
0.07	0.15000	0.09566

Табела 3.16: Експериментални резултати со АДП алгоритмот за $B_{max} = 3$

Со споредба на соодветните веројатности во Табела 3.14 (Табела 3.15) и Табела 3.16 може да се заклучи дека за кодот со 2-битни симболи АДП алгоритамот дава малку подобри резултати за PER и BER од стандардниот алгоритам за декодирање. Исто така, процесот на декодирање со АДП алгоритамот е двапати побрз отколку со стандардниот алгоритам.

3.7.2 Експерименти со квазигрупи од ред 256

Понатаму, направени се експерименти со азбуката од бајти (8-битни симболи) со користење на различни патерни, клучеви и квазигрупи од ред 256. Во овие експерименти, со двата алгоритми за декодирање (стандарден и АДП) се добиваат скоро исти вредности за PER и BER како за кодовите со азбука од нибли.

Во Табела 3.17, даден е еден пример. Овие резултати се добиени за кодови (72,288) со следните параметри:

- во стандарниот алгоритам - патернот за редундантност: 10 10 10 00 00 10 10 00 00 10 10 00 00 10 10 00 00 00 и клуч $k = 0123456789$ од 10 бајти,

- во АДП алгоритамот - патернот: 11 10 10 10 10 10 10 10 00 и два различни клучеви $k_1 = 0123456789$ и $k_2 = 5432897610$ од 10 бајти.

Во двата експерименти користена е иста квазигрупа од ред 256. Во Табела 3.17 споредени се веројатностите за пакет-грешка и бит-грешка до-

биени со стандардниот алгоритам и АДП алгоритмот за кодовите со 8-битни симболи (бајти) и нибли.

	PER		BER	
алгоритам	нибли	бајти	нибли	бајти
стандарден	0.1131	0.1100	0.0649	0.0749
АДП	0.1080	0.1029	0.0751	0.0838

Табела 3.17: Експериментални резултати за веројатностите за пакет-грешка и бит-грешка за $p = 0.08$, $B_{max} = 4$

Од резултатите во Табела 3.17 може да заклучи дека со двата алгоритми за декодирање се добиваат приближно исти вредности за PER и BER и за двете азбуки.

Заклучок

Во оваа глава е дефиниран нов алгоритам за кодирање/декодирање на случајните кодови базирани на квазигрупи со кој процесот на декодирање за кодови со рата 1/4 се забрзува за 4.5 пати. Дефиниран е нов метод за намалување на неуспешните декодирања кои завршуваат со повеќе кандидати и предложена е комбинација на двата методи за намалување на неуспешните декодирања, со која се добиваат подобри вредности за веројатноста за успешно декодирање. Исто така, испитани се перформансите на овие кодови кога во процесот на кодирање/декодирање се користат квазигрупи од ред 4 или ред 256, наместо квазигрупи од ред 16.

Глава 4

Примена на RCBQ за декодирање слики

Во оваа глава испитани се перформансите на случајните кодови базирани на квазигрупи за пренесување на слики низ бинарен симетричен канал. За таа цел направени се експерименти со користење на стандарниот алгоритам за кодирање/декодирање и новиот АДП алгоритам. Експерименталните резултати се споредени со соодветните резултати добиени со кодовите на Рид-Соломон.

4.1 Модификации на алгоритмите за нивна примена во декодирање слики

Во Глава 2, дефинирани се стандардните алгоритми за RCBQ, а во Глава 3 предложен е нов АДП алгоритам за овие кодови. Во двете верзии на правилото за декодирање има два типа на неуспешни декодирања: *грешка-празно-множество* и *грешка-повеќе-кандидати*. Кога ќе се појави *грешка-празно-множество*, тогаш процесот на декодирање завршува порано и само дел од пораката е декодиран. А во случај на *грешка-повеќе-кандидати* има повеќе од еден кандидат за декодираната порака. За примена на RCBQ за кодирање/декодирање на слики потребно е некако да се разрешат овие случаи на делумно и неединствено декодирана порака. Затоа, во експериментите со двата алгоритми користени се следните решенија за неуспешните декодирања

Неуспешно декодирање со *грешка-повеќе-кандидати*. Бидејќи во експериментите со АДП алгоритамот се јавуваа поголем број на неуспешни декодирања со *грешка-повеќе-кандидати*, предложивме хевристика за елими-

нирање на овој тип на грешки. Имено, ако се појави *грешка-повеќе-кандидати*, тогаш случајно се одбира една порака од редуцираните множества во последната итерација и таа порака се смета за декодирана порака. Сега, оваа хевристика се применува во експериментите со слики за двата алгоритми за RCBQ.

Неуспешно декодирање со *грешка-празно-множество*. Кога ќе се појави *грешка-празно-множество*, т.е., $S_i = \emptyset$ во стандардниот алгоритам (или во АДП алгоритамот двете редуцирани множества $S_i^{(1)}$ и $S_i^{(2)}$ се празни), се земаат стринговите без редундантните симболи од сите елементи во множеството S_{i-1} (или $S_{i-1}^{(1)}$ и $S_{i-1}^{(2)}$) и се определува нивниот максимален заеднички префикс. Ако овој подстринг има k симболи, тогаш за да се добие декодирана порака од l симболи се земаат овие k симболи и се додаваат $l-k$ нулти симболи на крајот од пораката. Во експериментите со слики забележавме дека овој тип на грешки прави највидливи промени во декодираните слики. Затоа, во експериментите со АДП алгоритамот користен е методот со враќање за намалување на бројот на неуспешни декодирања со *грешка-празно-множество*, дефиниран во Поглавје 3.4.

4.2 Експерименти

Направени се експерименти со неколку (во боја и црно-бели) слики. Тука, ќе бидат презентирани резултатите добиени со сликата дадена на Слика 4.1. Во експериментите користен е бинарен симетричен канал со следните вредности на веројатноста за бит-грешка: $p = 0.03$, $p = 0.06$, $p = 0.09$ и $p = 0.12$. Разгледани се само овие вредности на p, бидејќи за помали p нема видливи промени во декодираните слики. Од друга страна, за $p > 0.12$ кодирањето веќе нема смисла (веројатноста за бит-грешка е поголема од p). Исто така, помеѓу избраните вредности на p земен е чекор 0.03, за да се добијат видливи разлики во декодираните слики.

За секоја вредност на p, разгледувани се сликите добиени после пренос низ каналот во следните случаи:

а) без користење на код за поправање на грешки,

б) со користење на RCBQ со стандардниот алгоритам,

в) со користење на RCBQ со АДП алгоритамот (со враќање во случај на *грешка-празно-множество*),

г) со користење на Рид-Соломон код.

Слика 4.1: Оригинална слика

Сите експерименти се направени за кодот (72,288) со рата $R = 1/4$. Во експериментите користена е азбуката од нибли $Q = \{0, 1, \ldots, 9, a, b, c, d, e, f\}$ и квазигрупата дадена во Табела 2.1. Во двата алгоритми за RCBQ користени се патерните, клучевите и должините на блокови со кои се добиени најдобрите резултати. Експериментите се направени со користење на $B_{max} = 4$ во процесот на декодирање.

Во нашите експерименти со кодовите на Рид-Соломон (Reed-Solomon Codes - RSC) користена е скратената верзија на RSC(63,27) ([63]). Тоа е кодот RSC(48,12) над полето на Галоа GF(2^6) со примитивен полином $p(x) = 1 + X + X^6$ (кој ги има истите добри особини како и општите RSC). Овој скратен код има иста должина на кодните зборови (288 бита) и иста рата (1/4) како и разгледуваниот RCBQ.

4.2.1 Експериментални резултати за PER и BER

Со користење на Matlab, сликата дадена на Слика 4.1, прво е конвертирана во матрица од бајти, а потоа во низа од нибли. На тој начин добиена е листа од 16756 пораки од 18 нибли (72 бита). Во ова поглавје ќе бидат разгледани

вредностите добиени за веројатностите за пакет-грешка и бит-грешка, а потоа ќе бидат дадени добиените слики.

Прво, овие пораки беа пренесени низ бинарниот симетричен канал без користење на било кој код за поправање на грешки. Бидејќи, грешките во каналот се појавуваат случајно и независно, секој бит со иста веројатност може да биде неточно пренесен. Затоа, во овој случај во скоро сите пораки има по барем еден неточен симбол и вредностите на $PER = 1 - (1 - p)^{72}$ не се помали од 0.9 за сите вредности на веројатноста p за бит-грешка во бинарниот симетричен канал.

Потоа, направени се експерименти за истите вредности на p во каналот и истата слика, но со користење на RCBQ (со стандардниот алгоритам и со АДП алгоритамот со враќање во случај на *грешка-празно-множество*) како кодови за поправање на грешки. Во овие експерименти пораките прво се кодираат со соодветниот алгоритам за кодирање. После преносот низ бинарен симетричен канал, излезните пораки се декодираат со користење на соодветните алгоритми за декодирање (стандарден, АДП или Рид-Соломон). На овој начин, некои од грешките настанати при преносот, успешно се поправаат. Во Табела 4.1 и Табела 4.2 дадени се експериментални резултати за веројатностите за пакет-грешка (PER) и веројатностите за бит-грешка (BER), за различни вредности на веројатноста за бит-грешка p. Во овие табели PER_s и BER_s се веројатностите добиени со стандардниот алгоритам, PER_{c_back} и BER_{c_back} со АДП алгоритамот со враќање и PER_{rs} и BER_{rs} се веројатностите добиени со Рид-Соломон кодот.

p	PER_s	PER_{c_back}	PER_{rs}
0.03	0.0018	0.0032	0.0003
0.06	0.0338	0.0266	0.1157
0.09	0.1813	0.1282	0.7116
0.12	0.4719	0.3620	0.9748

Табела 4.1: Експериментални резултати за веројатноста за пакет-грешка

Од резултатите дадени во Табела 4.1 може да се види дека само за $p = 0.03$, PER добива помала вредности со кодот на Рид-Соломон. Додека за $p \geqslant 0.06$ подобри резултати дава RCBQ, особено со користење на нашиот АДП алгоритам.

p	BER_s	BER_{c_back}	BER_{rs}
0.03	0.0011	0.0010	0.0001
0.06	0.0251	0.0112	0.0241
0.09	0.1379	0.0570	0.1625
0.12	0.3715	0.1742	0.2565

Табела 4.2: Експериментални резултати за веројатноста за бит-грешка

Од Табела 4.2, може да се види дека разликите помеѓу резултатите за BER_s и BER_{rs} не се толку значајни. Причината за ова е во конструкцијата на RCBQ. Имено, кај овие кодови, кога некој бит е погрешно декодиран, тогаш скоро сите следни битови се неточно декодирани. Како последица на ова во некоја од итерациите се добива празно множество со кандидати за декодирање и само дел од пораката е декодирана. Исто така, може да се заклучи дека разликите помеѓу резултатите за BER_s и BER_{c_back} се позначајни. Имено, за $p \geqslant 0.06$, BER_{c_back} е приближно двапати помало од BER_s. Ова се случува бидејќи во експериментите со RCBQ, бит-грешката е најголема кога декодирањето завршува со *грешка-празно-множество*. Во експериментите со АДП алгоритамот користен е предложениот метод со враќање за намалување на овој тип на грешки. Како што е објаснето претходно, со овој метод бит-грешката се намалува дури и кога грешката нема да биде елиминирана, бидејќи со враќањето може да биде декодиран поголем дел од пораката. Да забележиме дека овие резултати и заклучоци се слични со соодветните резултати (дадени во претходните глави) добиени кога беа кодирани/декодирани пораки наместо слики.

Исто така, може да се забележи дека за RCBQ со стандардниот алгоритам и кодот на Рид-Соломон за $p \geqslant 0.09$ нема смисла да се користи кодирање/декодирање, бидејќи $BER > p$, но за $p = 0.09$, RCBQ со АДП алгоритамот сè уште дава $BER_{c\_back} < p$. За $p = 0.12$, со сите разгледувани алгоритми се добива BER полошо од p.

4.2.2 Визуелна илустрација на експериментите

Тука, ќе биде дадена визуелна илустрација на резултатите, т.е., сите трансформации на сликата, дадена на Слика 4.1, добиени во направените експери-

менти.

Сликите добиени за разгледуваните вредности на веројатноста за бит-грешка p во бинарен симетричен канал, претставени се на Слика 4.2 - Слика 4.5. Сликите под а) се добиени по преносот низ каналот без користење на код за поправање на грешки. Сликите добиени со користење на RCBQ со стандардниот и АДП алгоритамот претставени се под б) и в), соодветно. Под г) дадени се сликите кодирани/декодирани со Рид-Соломон кодот.

Сликите претставени на Слика 4.2 - Слика 4.5 визуелно ги претставуваат нашите претходно дадени заклучоци за вредностите на веројатноста за пакет-грешка. Имено, за $p = 0.03$ сликите добиени со двата алгоритми за RCBQ (Слика 4.2 б) и Слика 4.2 в)) се слични. Но, за $p \geqslant 0.06$ сликите добиени со користење на RCBQ со нашиот АДП алгоритам се почисти од сликите добиени со користење на стандардниот RCBQ и Рид-Соломон кодот.

Од сликите може да се види дека за сите разгледани вредности на p (освен за $p = 0.03$ и $p = 0.12$) сликите добиени со RCBQ со АДП алгоритамот се најчисти.

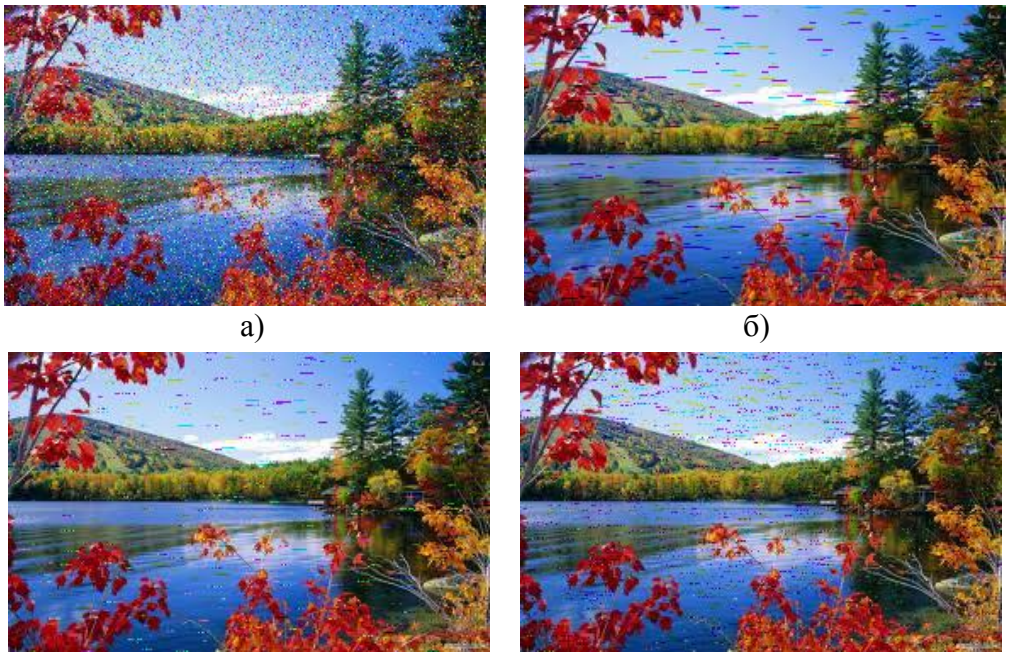
а)

б)

в)

г)

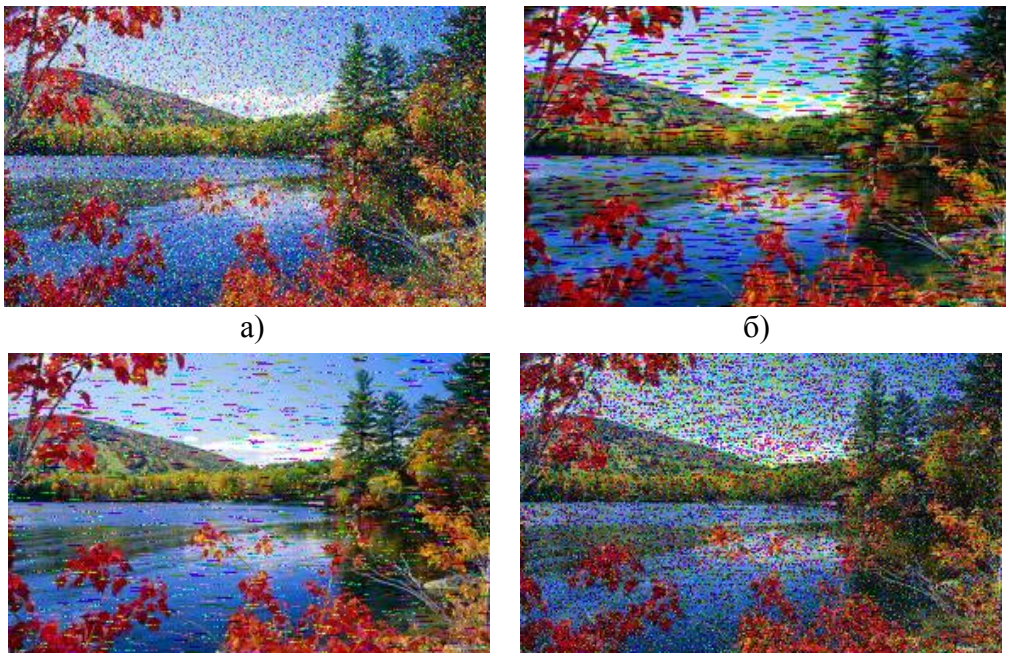
Слика 4.2: Слики за $p = 0.03$



а) б)

в) г)

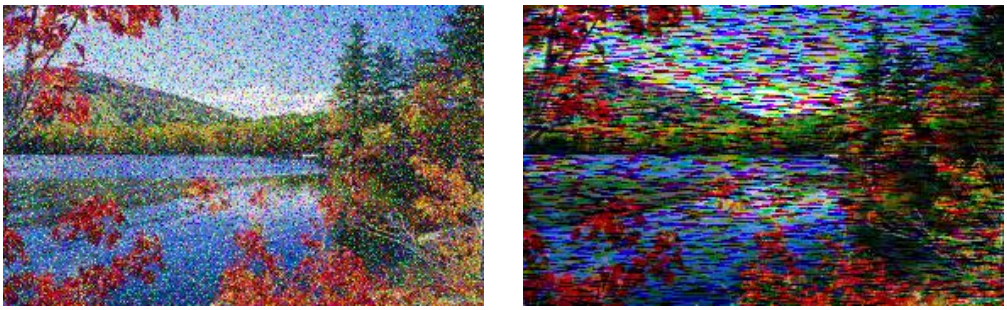
Слика 4.3: Слики за $p = 0.06$



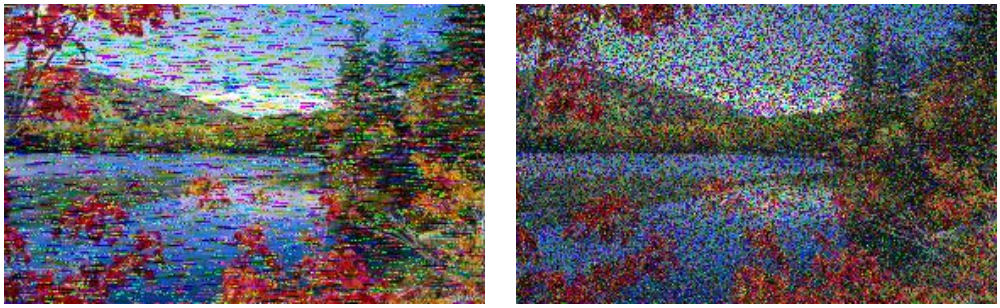
а) б)

в) г)

Слика 4.4: Слики за $p = 0.09$



а) б)



в) г)

Слика 4.5: Слики за $p = 0.12$

Исто така, како што е забележано и погоре, за $p = 0.12$ кодирањето/декодирањето на сликите нема смисла, бидејќи $BER > p$ и затоа сликата на Слика 4.5 а) е почиста од другите слики за истата вредност на p.

Од претходната дискусија може да се заклучи дека за декодирање на слики пренесени низ бинарен симетричен канал, RCBQ со АДП алгоритамот (со враќање) имаат подобри перформанси од другите разгледувани алгоритми.

4.2.3 Облик на недекодираните делови од пораките

Како што е објаснето погоре, во експериментите со двата алгоритми на RCBQ кога процесот на декодирање ќе заврши со *грешка-празно-множество* на местото од недекодираниот дел од пораката се стават нулти симболи. Всушност, овие нулти симболи се хоризонталните линии кои може да се забележат на Слика 4.2 б) – Слика 4.5 б) за стандардниот RCBQ и Слика 4.2 в) – Слика 4.5 в) за АДП алгоритамот. Додека, сликите добиени без користење на код за поправање на грешки (Слика 4.2 а) – Слика 4.5 а)), ги немаат овие

линии, но низ целата слика има точки кои всушност се неточно пренесените битови.

Бидејќи во сликите декодирани со RCBQ највидливите промени се во облик на хоризонтални линии, постои можност за понатамошно поправање на добиените слики. Имено, затоа што го знаеме обликот на оштетувањето на сликите можеме со анализирање на околните пиксели да ги поправиме неуспешно декодираните пиксели. Со користење на овие идеи може да се дефинира филтер за поправање на сликите декодирани со RCBQ со кој може да се добијат многу појасни слики. Ова е едно од отворените прашања за понатамошни истражувања кои произлегоа од оваа теза.

Заклучок

Во оваа глава се направени модификации во алгоритмите за декодирање на случајните кодови базирани на квазигрупи за нивна примена во кодирање/декодирање на слики и испитани се перформансите на овие кодови при кодирање/декодирање на слики пренесени низ бинарен симетричен канал. Исто така, добиените резултати се споредени со соодветните резултати добиени со Рид-Соломон кодовите. Од добиените резултати е заклучено дека за декодирање на слики пренесени низ бинарен симетричен канал, RCBQ со АДП алгоритамот (со враќање) има најдобри перформанси. Исто така, дадени се идеи за дефинирање на филтри за понатамошно поправање на декодираните слики.

Глава 5

Алгоритми-за-декодирање-со-4пресеци

Како што е претходно споменато, брзината на декодирање е најголем проблем за случајните кодови базирани на квазигрупи дефинирани во [30]. Со цел да ја подобриме брзината на декодирање на RCBQ, во Глава 3, дефиниравме нов АДП алгоритам, така што новиот процес на декодирање е 4.5 пати побрз од оригиналниот за код (72,288). Ова подобрување на брзината на декодирање ни даде идеја за користење на пресеци од повеќе множества S_i, со цел да добиеме поголемо зголемување на брзината на декодирање. Во АДП алгоритмот се користат две трансформации на редундантната порака со различни параметри, а кандидатите за декодираната порака се добиваат со користење на пресек на соодветните множества со кандидати за декодирање. Сега, направени се модификации на тој алгоритам каде што се користат четири трансформации на редундантната порака. Со овој нов алгоритам, наречен алгоритам-за-декодирање-со-4пресеци (4-Sets-Cut-Decoding или АД4П алгоритам) се добива поголемо подобрување на брзината на декодирање. Исто така, со цел да ги подобриме и веројатностите за пакет-грешка и бит-грешка ќе дефинираме неколку методи за генерирање на редуцираните множества со кандидати за декодирање.

Во оваа глава предложени се новите АД4П алгоритми и анализирани се перформансите на различни алгоритми за декодирање на RCBQ (стандардниот, АДП и АД4П алгоритмите) за код со рата $R = 1/8$. Исто така, разгледана е примената на методите за намалување на неуспешните декодирања во новите предложени алгоритми.

5.1 Кодирање со АД4П алгоритмите

Во оваа модификација на АДП алгоритамот наместо код (N_{block}, N) со рата R, се користат четири $(N_{block}, N/4)$ кодови со рата $4R$, кои декодираат иста порака од N_{block} битови.

Значи, во процесот на кодирање, алгоритмот за криптирање, даден на Слика 2.2, се применува четири пати на иста редундантна порака L со користење на различни параметри (различни клучеви или квазигрупи), а кодниот збор на пораката се добива со конкатенација на четирите кодни зборови од $N/4$ битови.

5.2 Прва верзија на АД4П алгоритамот (АД4П#1 алгоритам)

После преносот низ бинарен симетричен канал, излезната порака $D = D^{(1)}D^{(2)}...D^{(s)}$ се дели 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 овие пораки се декодираат паралелно со соодветните параметри. Слично како и во АДП алгоритмот со две множества, во секоја итерација од процесот на декодирање, се прави редукција на множествата со кандидати за декодирање добиени во четирите процеси на декодирање. Во оваа нова модификација на АДП алгоритмот, наречена АД4П алгоритам, множествата со кандидати за декодирање се генерираат на следниот начин.

Чекор 1. Нека $S_0^{(1)} = (k_1^{(1)} \dots k_n^{(1)}; \lambda)$, ..., $S_0^{(4)} = (k_1^{(4)} \dots k_n^{(4)}; \lambda)$, каде λ е празен стринг, а $k_1 = k_1^{(1)} \dots k_n^{(1)}$, ..., $k_4 = k_1^{(4)} \dots k_n^{(4)}$ се почетните клучеви кои се користени за добивање на четирите кодни зборови.

Чекор 2. Нека $S_{i-1}^{(1)}, \dots, S_{i-1}^{(4)}$ се дефинирани за $i \geqslant 1$.

Чекор 3. Нека четирите множества со кандидати за декодирање $S_i^{(1)}, \dots, S_i^{(4)}$ се добиени во четирите процеси на декодирање, на ист начин како и во стандарните RCBQ.

Чекор 4. Нека $V_1 = \{w_1 w_2 \dots w_{r \cdot a \cdot i} | (\delta, w_1 w_2 \dots w_{r \cdot a \cdot i}) \in S_i^{(1)}\}$, ..., $V_4 = \{w_1 w_2 \dots w_{r \cdot a \cdot i}\ | (\delta, w_1 w_2 \dots w_{r \cdot a \cdot i}) \in S_i^{(4)}\}$ и $V = V_1 \cap V_2 \cap V_3 \cap V_4$.

Чекор 5. За секое $j = 1, 2, 3, 4$ и за секој $(\delta, w_1 w_2 \dots w_{r \cdot a \cdot i}) \in S_i^{(j)}$, ако $w_1 w_2 \dots w_{r \cdot a \cdot i} \notin V$ тогаш $S_i^{(j)} \leftarrow S_i^{(j)} \setminus \{(\delta, w_1 w_2 \dots w_{r \cdot a \cdot i})\}$.
(* Всушност, за секое $j = 1, 2, 3, 4$, од множествата $S_i^{(j)}$ се елиминираат сите елементи чиј втор дел не се поклопува со вториот дел на некој елемент во сите останати три множества. Во следната итерација четирите процеси ги користат соодветните редуцирани множества $S_i^{(1)}$, $S_i^{(2)}$, $S_i^{(3)}$, $S_i^{(4)}$ *).

Чекор 6. Ако $i < s/4$ тогаш зголеми го i и врати се на Чекор 3.

Правилото за декодирање во новиот АД4П алгоритам е дефинирано на следниот начин.

- Ако, после последната итерација, сите редуцирани множества со кандидати за декодирање $S_{s/4}^{(1)}$, $S_{s/4}^{(2)}$, $S_{s/4}^{(3)}$, $S_{s/4}^{(4)}$ имаат само еден елемент со иста втора компонента $w_1 \dots w_{r \cdot a \cdot s/4}$, тогаш $L = w_1 \dots \ w_{r \cdot a \cdot s/4}$ е декодираната редундантна порака. Во овој случај велиме дека имаме *успешно декодирање*.

- Ако, после последната итерација, редуцираните множества $S_{s/4}^{(1)}$,..., $S_{s/4}^{(4)}$ имаат повеќе од еден елемент, тогаш имаме *грешка-повеќе-кандидати*. Во овој случај се применува истата хевристика како и во АДП алгоритамот (случајно се избира порака од редуцираните множества во последната итерација).

- Ако се добие само едно празно множество (или две празни множества) со кандидати за декодирање, тогаш декодирањето продолжува со трите (или двете) непразни множества (множеството V во Чекор 4 е пресек само на непразните множества).

- Ако, во некоја итерација се добие само едно непразно множество, тогаш декодирањето продолжува со непразното множество користејќи го стандардниот алгоритам за декодирање на RCBQ .

- Ако во некоја итерација се добие $S_i^{(1)} = S_i^{(2)} = S_i^{(3)} = S_i^{(4)} = \emptyset$, процесот се стопира и велиме дека се појавила *грешка-празно-множество*.

5.3 Втора верзија на АД4П алгоритамот (АД4П#2 алгоритам)

Во експериментите со АД4П#1 алгоритамот забележавме дека кога процесот на декодирање завршува со *грешка-празно-множество*, т.е., кога сите четири редуцирани множества со кандидати за декодирање се празни, многу често точната порака ја има во три од четирите нередуцирани множества. Затоа, во втората верзија на АД4П алгоритамот (АД4П#2 алгоритам) направена е следната модификација во Чекор 4 од процедурата за генерирање на множествата со кандидати за декодирање.

Чекор 4$^{\#2}$ Нека $V_1 = \{w_1 w_2 \dots w_{r \cdot a \cdot i} | (\delta, w_1 w_2 \dots w_{r \cdot a \cdot i}) \in S_i^{(1)}\}, \dots,$
$V_4 = \{w_1 w_2 \dots w_{r \cdot a \cdot i} | (\delta, w_1 w_2 \dots w_{r \cdot a \cdot i}) \in S_i^{(4)}\}$ и $V = V_1 \cap V_2 \cap V_3 \cap V_4$.

1. Ако $V = \emptyset$ тогаш $V' = V_1 \cap V_2 \cap V_3$ и $V = V'$.
2. Ако $V' = \emptyset$ тогаш $V'' = V_1 \cap V_2 \cap V_4$ и $V = V''$.
3. Ако $V'' = \emptyset$ тогаш $V''' = V_1 \cap V_3 \cap V_4$ и $V = V'''$.
4. Ако $V''' = \emptyset$ тогаш $V^{iv} = V_2 \cap V_3 \cap V_4$ и $V = V^{iv}$.

Всушност, во оваа модификација ако пресекот од сите четири множества V_1, V_2, V_3, V_4 е празно множество, тогаш се бара непразен пресек на три множества.

На овој начин се добива големо подобрување на веројатностите за пакет-грешка и бит-грешка без намалување на брзината на декодирање. Но, за поголемо подобрување на перформансите разгледани се уште две модификации на Чекор 4, кога пресекот на сите четири множества е празно множество.

5.4 Трета верзија на АД4П алгоритамот (АД4П#3 алгоритам)

Во експериментите со АД4П#2 алгоритмот, добиени се подобри резултати за PER и BER за сите вредности на веројатноста p за бит-грешка на бинарен

симетричен канал. Но, со анализа на експериментите кои со оваа верзија на алгоритмот завршиле со *грешка-празно-множество*, забележана е следната ситуација. Имено, во некои експерименти добиено е $V' \neq \emptyset$, но точната порака ја нема во множеството V', но ја има во некое од множествата V'' или V''' или V^{iv}. Слично, ако ($V' = \emptyset$ и $V'' \neq \emptyset$) или ($V' = \emptyset$, $V'' = \emptyset$ и $V''' \neq \emptyset$) и точната порака припаѓа во некој од следните пресеци, кои не се разлгедуваат ако претходниот пресек не е празен.

Затоа, направена е друга модификација на Чекор 4 во која ако пресекот на сите четири множества е празно множество, тогаш множеството $V = V' \cup V'' \cup V''' \cup V^{iv}$, т.е., новата модификација на Чекор 4 е следната.

Чекор 4$^{\#3}$ Нека $V_1 = \{w_1 w_2 \dots w_{r\cdot a\cdot i} | (\delta, w_1 w_2 \dots w_{r\cdot a\cdot i}) \in S_i^{(1)}\}, \dots,$
$V_4 = \{w_1 w_2 \dots w_{r\cdot a\cdot i} | (\delta, w_1 w_2 \dots w_{r\cdot a\cdot i}) \in S_i^{(4)}\}$ и $V = V_1 \cap V_2 \cap V_3 \cap V_4$.

Ако $V = \emptyset$ тогаш
$V = (V_1 \cap V_2 \cap V_3) \cup (V_1 \cap V_2 \cap V_4) \cup (V_1 \cap V_3 \cap V_4) \cup (V_2 \cap V_3 \cap V_4)$.

Со оваа модификација, ако новото множество $V \neq \emptyset$ се добиваат подобри вредности на веројатностите за пакет-грешка и бит-грешка отколку со АД4П#2 алгоритмот. Исто така, оваа модификација не ја намалува брзината на декодирање.

5.5 Четврта верзија на АД4П алгоритамот (АД4П#4 алгоритам)

Во третата верзија на АД4П алгоритамот, ако новото множество $V = \emptyset$, тогаш се јавува неуспешно декодирање со *грешка-празно-множество*. Со цел да се намали бројот на овие грешки, направена е нова модификација во алгоритмот. Всушност, ако после Чекор $4^{\#3}$ повторно се добие празно множество V тогаш множеството V ќе биде унија од сите пресеци од две множества, т.е., новиот Чекор 4 е следниот.

Чекор 4$^{\#4}$ Нека $V_1 = \{w_1 w_2 \dots w_{r\cdot a\cdot i} | (\delta, w_1 w_2 \dots w_{r\cdot a\cdot i}) \in S_i^{(1)}\}, \dots,$
$V_4 = \{w_1 w_2 \dots w_{r\cdot a\cdot i} | (\delta, w_1 w_2 \dots w_{r\cdot a\cdot i}) \in S_i^{(4)}\}$ и $V = V_1 \cap V_2 \cap V_3 \cap V_4$.
Ако $V = \emptyset$ тогаш
$V = (V_1 \cap V_2 \cap V_3) \cup (V_1 \cap V_2 \cap V_4) \cup (V_1 \cap V_3 \cap V_4) \cup (V_2 \cap V_3 \cap V_4)$.

Ако $V = \emptyset$ тогаш
$V = (V_1 \cap V_2) \cup (V_1 \cap V_3) \cup (V_1 \cap V_4) \cup (V_2 \cap V_3) \cup (V_2 \cap V_4) \cup (V_3 \cap V_4)$.

Со оваа нова модификација се добиваат подобри резултати само за $B_{max} = 4$. Кога $B_{max} = 5$ се добива добар процент на елиминирани неуспешни декодирања со *грешка-празно-множество*, но поголем број на неуспешни декодирања со *грешка-повеќе-кандидати*.

5.6 Споредба на алгоритмите за рата $R = 1/8$

Во ова поглавје дадени се експерименталните резултати за веројатностите за пакет-грешка (PER) и бит-грешка (BER) добиени со новите АД4П алгоритми и споредени се со резултатите добиени со стандардниот алгоритам за декодирање и со АДП алгоритмот. Исто така, анализирани се подобрувањата на перформансите на кодот добиени со сите предложени модификации на Чекор 4. Во ова поглавје ќе бидат споредени и брзините на декодирање на сите алгоритми.

Во претходните глави од оваа теза беа дадени експериментални резултати за кодот (72,288) со рата $1/4$ добиени со стандардниот и АДП алгоритмот. За да се добие код (N_{block}, N) со рата R во предложените АД4П алгоритми се користат четири кода $(N_{block}, N/4)$ со рата $4R$. Затоа, во овие алгоритми за код со рата $1/4$ нема редундантност (ако $R = 1/4$, тогаш $N/4 = N_{block}$, т.е., должината на кодниот збор е еднаква на должината на пораката). Така што, за споредба направени се експерименти за код (72,576) со рата $R = 1/8$.

Во сите експерименти користена е азбуката $Q = \{0, 1, \ldots 9, a, b, c, d, e, f\}$ од нибли со квазигрупните операции $*$ и $\backslash$ над Q дадени во Табела 2.1 и Табела 2.2 и блокови од 4 нибли во процесот на декодирање.

Најдобри резултати со стандардниот алгоритам за код (72,576) добиени се за следните параметри:

– патернот за редундантност: 1100 1000 0000 0000 0000 0000 1100 1000 0000 0000 0000 0000 1100 1000 0000 0000 0000 0000 1100 1000 0000 0000 0000 0000 1100 1000 0000 0000 0000 0000 1100 1000 0000 0000 0000 0000, и

– клучот $k = 0123456789$ од 10 нибли.

Со АДП алгоритмот, најдобри резултати се добиени со следните параметри:

– патернот 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 $k_1 = 0123456789$ и $k_2 = 5432897610$ од 10 нибли.

Во експериментите со новите АД4П алгоритми, најдобри резултати се добиени со користење на следните параметри:

– патернот за редундантност: 1100 1110 1100 1100 1110 1100 1100 1100 0000 за рата $1/2$, и

– четири различни клучеви $k_1 = 0123456789$, $k_2 = 5678901234$, $k_3 = abcdef0123$ и $k_4 = f0123abcde$ од 10 нибли.

Експерименталните резултати за веројатностите за пакет-грешка за $B_{max} = 4$ и различни вредности на веројатноста p за бит-грешка на бинарен симетричен канал дадени се во Табела 5.1 и претставени се на Слика 5.1. Во оваа табела PER_s се веројатностите за пакет-грешка добиени со стандардниот алгоритам, а PER_c веројатностите добиени со АДП алгоритмот. Во истата табела со $PER_{c4.1}$, $PER_{c4.2}$, $PER_{c4.3}$ и $PER_{c4.4}$ означени се веројатностите за пакет-грешка добиени со АД4П#1, АД4П#2, АД4П#3 и АД4П#4, соодветно.

За сите разгледани алгоритми направени се експерименти сè додека не се добие $BER > p$, бидејќи кога $BER > p$ користењето на кодовите нема смисла (тогаш бројот на неточно декодирани битови е поголем од бројот на неточно пренесени битови).

p	PER_s	PER_c	$PER_{c4.1}$	$PER_{c4.2}$	$PER_{c4.3}$	$PER_{c4.4}$
0.02	0.0011	0	0	0	0	0
0.03	0.0029	0.0017	0.0020	0.0006	0	0
0.04	0.0106	0.0074	0.0086	0.0031	0.0006	0.0006
0.05	0.0326	0.0134	0.0254	0.0074	0.0026	0.0026
0.06	0.0663	0.0371	0.0600	0.0106	0.0034	0.0029
0.07	0.1300	0.0643	0.1129	0.0237	0.0094	0.0063
0.08	0.2200	0.1257	0.1906	0.0383	0.0209	0.0091
0.09	/	0.1791	/	0.0791	0.0437	0.0189
0.10	/	/	/	0.1329	0.0906	0.0277
0.11	/	/	/	0.2166	0.1649	0.0534
0.12	/	/	/	0.3140	0.2666	0.0977
0.13	/	/	/	/	/	0.1537
0.14	/	/	/	/	/	0.2429
0.15	/	/	/	/	/	0.3631

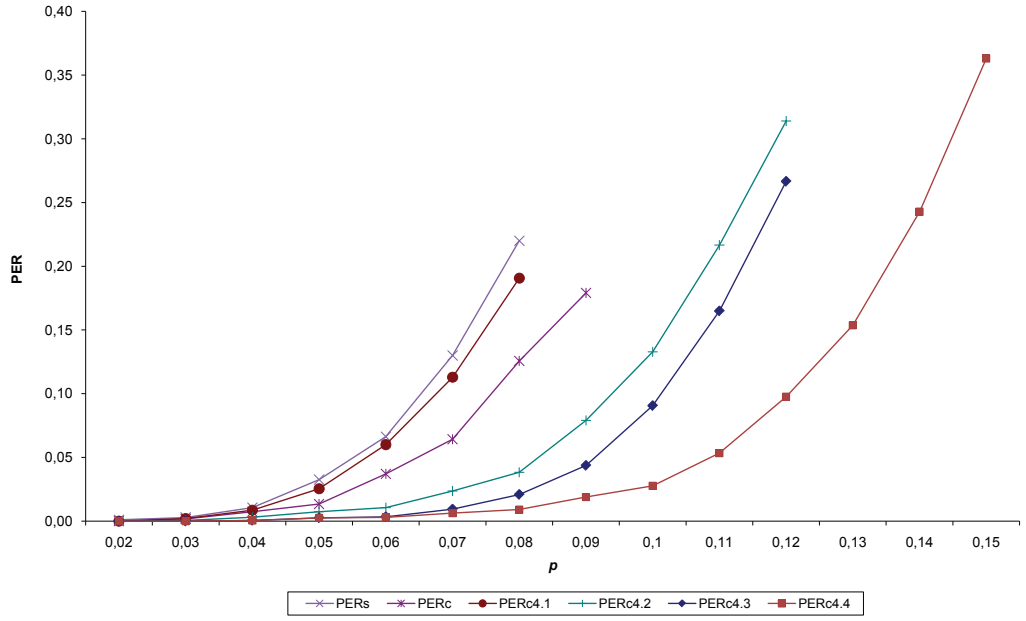
Табела 5.1: Експериментални резултати за веројатностите за пакет-грешка за $B_{max} = 4$

Од добиените резултати за PER, дадени во Табела 5.1, може да се изведат следните заклучоци. Со користење на АДП алгоритмот наместо стандардниот алгоритам се добива големо подобрување на веројатностите за пакет-грешка (за $p > 0.04$, PER_c се приближно двапати помали од PER_s). Исто така, АДП алгоритмот е повеќе од двапати побрз од стандардниот алгоритам.

Со АД4П#1 алгоритмот се добиваат полоши резултати за PER од вредностите добиени со АДП алгоритамот. Но, овој алгоритам е повеќе од 3 пати побрз од стандардниот и околу 1.4 пати побрз од АДП алгоритамот.

Од вредностите добиени за $PER_{c4.2}$ во Табела 5.1 може да се види дека со оваа модификација за сите вредности на p се добиваат подобри резултати

за PER споредено со вредностите добиени со стандардниот и АДП алгоритамот ($PER_{c4.2}$ се од 2 до 3.5 пати помали од PER_c). Исто така, со АД4П#2 алгоритамот брзината на декодирање е скоро иста како и со АД4П#1 алгоритамот (повеќе од 3 пати побрз од стандардниот алгоритам).



Слика 5.1: Споредба на PER за $B_{max} = 4$

Од резултатите за $PER_{c4.3}$ може да се види дека со АД4П#3 алгоритамот успешно се елиминирани голем број на неуспешни декодирања со *грешка-празно-множество* (за $p \geqslant 0.04$, $PER_{c4.3}$ се од 1.2 до 5.2 пати помали од $PER_{c4.2}$), повторно без да се намали брзината на декодирање. И со последната модификација, т.е., АД4П#4, за $p > 0.05$ добиени се подобри резултати за веројатностите за пакет-грешка со скоро иста брзина на декодирање. Освен тоа, за $p > 0.09$, вредностите на $PER_{c4.4}$ се приближно 3 пати помали од $PER_{c4.3}$.

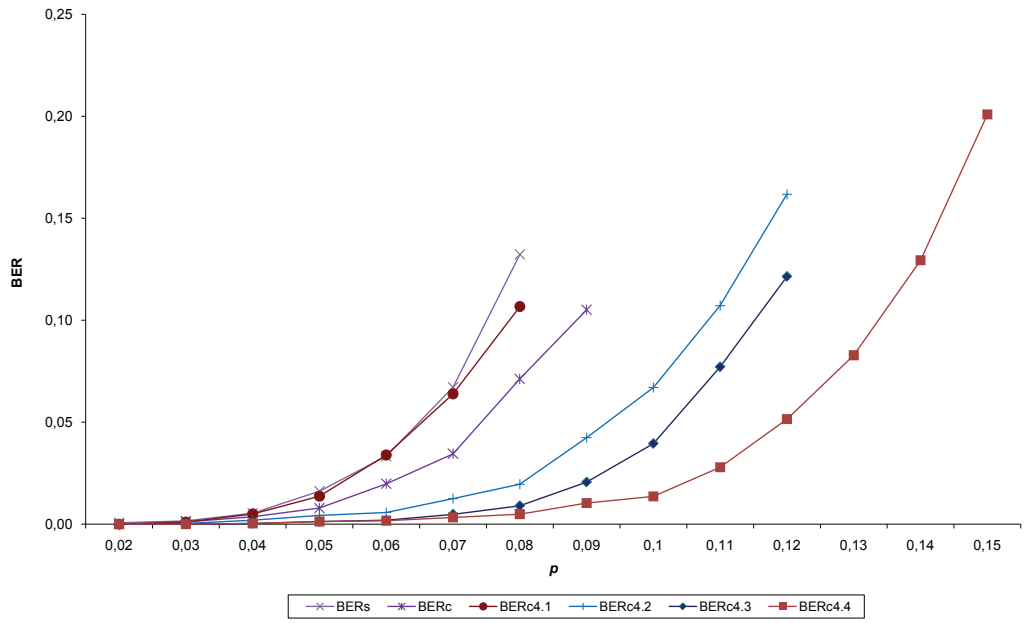
Во Табела 5.2 и Слика 5.2 дадени се вредностите за веројатноста за бит-грешка (BER) добиени во истите експерименти и за вредностите добиени со различни алгоритми користени се истите ознаки како и во Табела 5.1.

p	BER_s	BER_c	$BER_{c4.1}$	$BER_{c4.2}$	$BER_{c4.3}$	$BER_{c4.4}$
0.02	0.0007	0	0	0	0	0
0.03	0.0016	0.0011	0.0010	0.0003	0	0
0.04	0.0054	0.0036	0.0050	0.0019	0.0003	0.0003
0.05	0.0162	0.0079	0.0137	0.0043	0.0013	0.0013
0.06	0.0333	0.0198	0.0339	0.0057	0.0019	0.0017
0.07	0.0672	0.0345	0.0639	0.0125	0.0048	0.0033
0.08	0.1323	0.0712	0.1067	0.0196	0.0090	0.0049
0.09	/	0.1051	/	0.0424	0.0206	0.0103
0.10	/	/	/	0.0670	0.0395	0.0136
0.11	/	/	/	0.1072	0.0771	0.0279
0.12	/	/	/	0.1618	0.1214	0.0515
0.13	/	/	/	/	/	0.0828
0.14	/	/	/	/	/	0.1294
0.15	/	/	/	/	/	0.2009

Табела 5.2: Експериментални резултати за веројатностите за бит-грешка за $B_{max} = 4$

Од резултатите за BER во Табела 5.2 може да се изведат истите заклучоци како и за PER (за сите алгоритми и за сите p, BER е приближно $PER/2$).

Исто така, за истите кодови направени се експерименти и со користење на $B_{max} = 5$ во процесот на декодирање. Во Табела 5.3 (Слика 5.3) и Табела 5.4 (Слика 5.4), претставени се резултатите добиени за веројатностите за пакет-грешка и бит-грешка за различни вредности на веројатноста p за бит-грешка на бинарен симетричен канал.



Слика 5.2: Споредба на BER за $B_{max} = 4$

Од резултатите за PER_s и PER_c во Табела 5.3, може да се заклучи дека за овој код со користење на АДП алгоритмот се добиваат подобри резултати отколку со стандарниот алгоритам. Од друга страна, во однос на брзината на декодирање, за $p < 0.06$ експериментите со АДП алгоритмот се 5.2 пати побрзи од експериментите со стандардниот алгоритам. Но, за $p \geqslant 0.06$, во некои експерименти со АДП алгоритамот, се добива голема кардиналност на множествата со кандидати за декодирање (после некоја итерација), така што брзината на декодирање се намалува (но сè уште е подобра од брзината добиена со стандардниот алгоритам).

Слично како и за $B_{max} = 4$, сега за $p > 0.04$, со користење на новиот АД4П#1 алгоритам се добиваат полоши резултати за PER и BER од резултатите со АДП алгоритмот. Но, од времетраењето на нашите експерименти, може да се заклучи дека овој алгоритам е 6.3 пати побрз од стандардниот алгоритам и од 1.2 до 6.2 пати побрз од АДП алгоритмот.

p	PER_s	PER_c	$PER_{c4.1}$	$PER_{c4.2}$	$PER_{c4.3}$	$PER_{c4.4}$
0.03	0.00571	0.00007	0.00007	0	0	0
0.04	0.00714	0.00086	0.00057	0.00057	0.00029	0.00029
0.05	0.01000	0.00257	0.00286	0.00200	0.00114	0.00086
0.06	0.01086	0.00571	0.00743	0.00371	0.00229	0.00229
0.07	0.02457	0.01429	0.01571	0.00657	0.00286	0.00286
0.08	0.04829	0.03057	0.03171	0.02171	0.01114	0.01114
0.09	0.07171	0.05086	0.05914	0.03629	0.01514	0.01486
0.10	0.12057	0.08800	0.09514	0.05543	0.02429	0.02429
0.11	0.18057	0.13629	0.15114	0.09686	0.04543	0.04514
0.12	/	0.18886	0.22257	0.15029	0.07914	0.07829
0.13	/	0.28948	/	0.21514	0.10943	0.10629
0.14	/	/	/	0.28943	0.17200	0.16029
0.15	/	/	/	/	0.22857	0.22600
0.16	/	/	/	/	0.32314	0.30457
0.17	/	/	/	/	/	0.39143

Табела 5.3: Експериментални резултати за веројатностите за пакет-грешка за $B_{max} = 5$

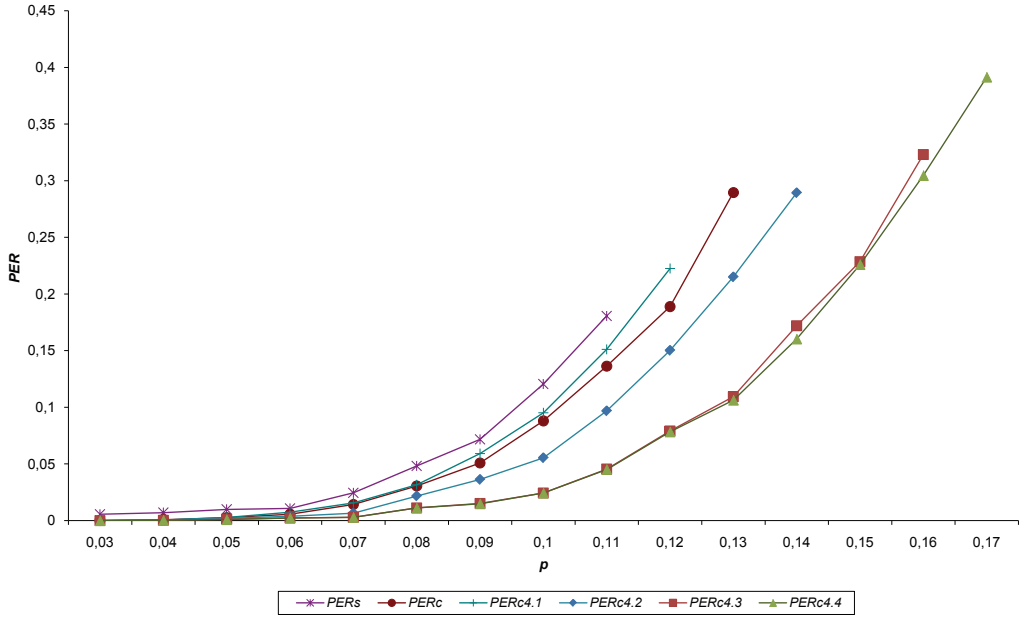
Од резултатите добиени со АД4П#2 алгоритмот, може да се види дека се добива подобрување на вредностите за PER и BER со незначително мало намалување на брзината на декодирање (најголемата разлика е 0.84 сек. по порака). Исто така, со АД4П#3 алгоритмот, се добиваат скоро двапати подобри вредности за PER и BER споредено со АД4П#2 алгоритмот. Повторно брзината е скоро непроменета.

Резултатите за $PER_{c4.3}$ и $PER_{c4.4}$ се скоро идентични, т.е., за двете модификации на алгоритмот, вкупниот број на неуспешни декодирања е скоро ист. Но, соодносите на бројот на *грешки-празно-множество* и *грешки-повеќе-кандидати* добиени со овие две модификации на алгоритмот се многу различни. Имено, со АД4П#3 алгоритамот се добиваат само неколку неуспешни декодирања со *грешка-повеќе-кандидати*. Од друга страна, со предложената

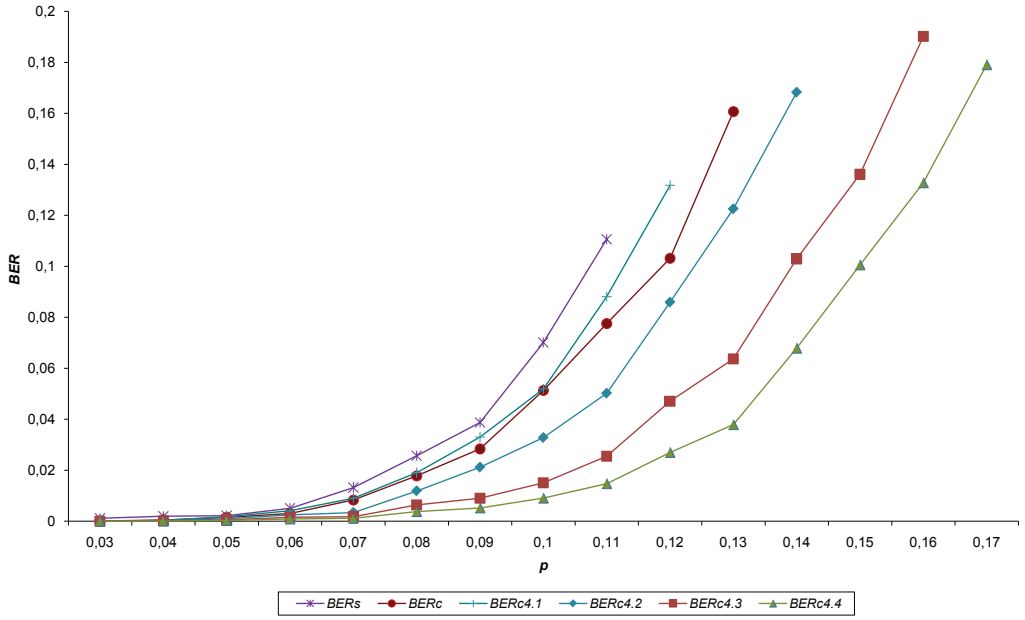
модификација во АД4П#4 алгоритамот за намалување на бројот на *грешки-празно-множество*, многу од овие грешки се елиминираат, но се добива многу поголем број на *грешки-повеќе-кандидати* (посебно за $p \geqslant 0.08$). Затоа, резултатите за $BER_{c4.4}$ се помали од $BER_{c4.3}$. Исто така, во некои експерименти со АД4П#4 алгоритамот за $p \geqslant 0.08$ после некоја итерација се добива многу голем кардинален број на множествата со кандидати за декодирање и во некои случаи дури мора и да се стопира декодирањето на пораката (заради недоволно меморија за процесирање на елементите). Затоа, со последната модификација на АД4П алгоритамот, со зголемување на p брзината на декодирање континуирано опаѓа. На пример, за $p = 0.13$, АД4П#4 алгоритамот е 6.4 пати поспор од АД4П#3 алгоритамот.

p	BER_s	BER_c	$BER_{c4.1}$	$BER_{c4.2}$	$BER_{c4.3}$	$BER_{c4.4}$
0.03	0.00117	0.00005	0.00004	0	0	0
0.04	0.00195	0.00047	0.00037	0.00021	0.00011	0.00007
0.05	0.00218	0.00148	0.00179	0.00096	0.00049	0.00025
0.06	0.00509	0.00302	0.00413	0.00251	0.00152	0.00087
0.07	0.01323	0.00831	0.00902	0.00339	0.00178	0.00113
0.08	0.02569	0.01778	0.01909	0.01186	0.00636	0.00378
0.09	0.03876	0.02836	0.03303	0.02115	0.00899	0.00521
0.10	0.07013	0.05131	0.05193	0.03279	0.01505	0.00906
0.11	0.11062	0.07755	0.08812	0.05017	0.02546	0.01472
0.12	/	0.10314	0.13179	0.08590	0.04704	0.02698
0.13	/	0.16065	/	0.12251	0.06365	0.03789
0.14	/	/	/	0.16827	0.10289	0.06789
0.15	/	/	/	/	0.13609	0.10059
0.16	/	/	/	/	0.19017	0.13277
0.17	/	/	/	/	/	0.17911

Табела 5.4: Експериментални резултати за веројатностите за бит-грешка за $B_{max} = 5$



Слика 5.3: Споредба на PER за $B_{max} = 5$



Слика 5.4: Споредба на BER за $B_{max} = 5$

Од сите презентирани резултати добиени во експериментите со различни алгоритми за декодирање за код (72,576) може да се заклучи дека за овој код со новите АД4П алгоритми се добива големо подобрување на брзината на декодирање и многу подобри резултати за веројатностите за пакет-грешка и бит-грешка.

Исто така, од експериментите може да се заклучи дека АД4П#4 алгоритамот дава најдобри резултати за вредностите на PER и BER, посебно за $B_{max} = 4$. Но, за $B_{max} = 5$, брзината на декодирање на овој алгоритам е помала од брзината на декодирање на АД4П#3 алгоритамот и притоа се добива голем број на *грешки-повеќе-кандидати*.

Може да се заклучи дека АД4П#4 алгоритамот е најдобар за кодот (72,576) за $B_{max} = 4$ во процесот на декодирање. Од друга страна, за $B_{max} = 5$, како компромис помеѓу брзината и точното декодирање, најдобар алгоритам е АД4П#3 алгоритамот.

5.7 Експерименти со методите за намалување на бројот на грешки

Како што е веќе споменато во Глава 2, кај RCBQ неуспешно декодирање со *грешка-празно-множество* се јавува кога при преносот во некој од блоковите на кодираната порака ќе се појават повеќе од предвидените B_{max} бит грешки. Некои од овие грешки може да бидат елиминирани ако се поништат неколку итерации од процесот на декодирање и потоа дел од нив или сите се извршат со поголема вредност на B_{max}. Затоа, во Поглавје 2.5 предложен е метод со враќање за намалување на бројот на неуспешни декодирања со *грешка-празно-множество*. Во овој метод ако во некоја итерација (на пример i^{th}-тата) од декодирањето се добие празно множество, тогаш k претходни итерации ($(i-1)^{th}$, $(i-2)^{th}$, ..., $(i-k)^{th}$) се поништуваат. Потоа, првата од поништените итерации ($(i-k)^{th}$) се извршува со $B_{max} = B_{max}+1$ или $B_{max}+2$, а следните итерации продолжуваат со старата вредност на B_{max}. Со користење на овој метод за намалување на бројот на *грешки-празно-множество* добиени се подобрувања на веројатностите за пакет-грешка и бит-грешка во стандардниот алгоритам (Поглавје 2.5) и АДП алгоритамот (Поглавје 3.4).

Овие подобрувања ни дадоа идеја за користење на сличен метод со враќање и за неуспешните декодирања со *грешка-повеќе-кандидати*, кој е дефиниран во Поглавје 3.5. Во овој метод, ако процесот на декодирање заврши со повеќе елементи во редуцираните множества со кандидати за декодирање во последната итерација, тогаш се поништуваат и првата од поништените итерации се извршува со користење на помала вредност на B_{max} (следните итерации ја користат претходната вредност на B_{max}).

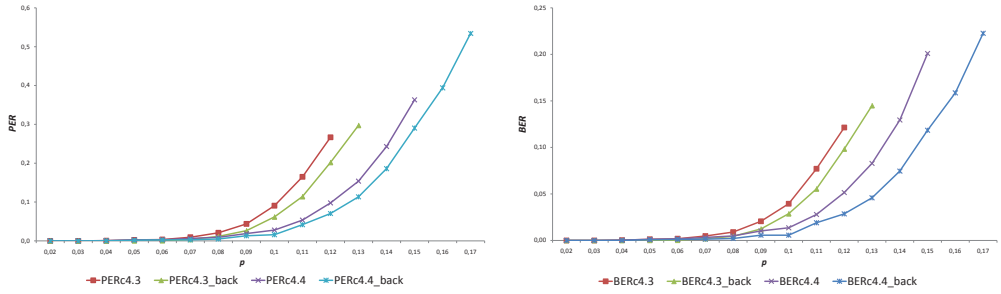
Направени се експерименти со користење на следната комбинација на овие два методи со враќање во новите АД4П алгоритми. Ако се добие *грешка-празно-множество*, т.е., празни редуцирани множества, во некоја итерација од процесот на декодирање, тогаш се поништуваат две итерации (или една итерација) и првата од поништените итерации се извршува со користење на $B_{max} + 2$. Ако процесот на декодирање заврши со повеќе елементи во множествата после последната итерација, тогаш процесот се враќа две итерации назад и предпоследната итерација се извршува со $B_{max} - 1$. Исто така, ако после враќањето за *грешка-празно-множество* се добијат повеќе кандидати во последната итерација, тогаш се прави уште едно враќање за *грешка-повеќе-кандидати*. Со користење на оваа комбинација на методите за намалување на бројот на неуспешни декодирања, повторени се експериментите со третата и четвртата верзија на АД4П алгоритамот (со кои беа добиени најдобри резултати).

За $B_{max} = 5$, со двете верзии на алгоритмот, најдобри резултати се добиваат ако во случај на *грешка-празно-множество* се поништат две итерации. Но, за $B_{max} = 4$ со АД4П#3 алгоритмот подобри резултати се добиваат ако во враќањето за *грешка-празно-множество* се поништи една итерација.

Во Табела 5.5, Слика 5.5 а) (за $B_{max} = 4$) и Табела 5.7, Слика 5.6 а) (за $B_{max} = 5$) споредени се веројатностите за пакет-грешка ($PER_{c4.3\_back}$, $PER_{c4.4\_back}$) добиени со користење на дефинираната комбинација на двата методи со враќање и веројатностите ($PER_{c4.3}$, $PER_{c4.4}$) добиени со третата и четвртата верзија на АД4П алгоритамот без враќање (од табелите дадени во претходното поглавје). Исто така, во Табела 5.6, Слика 5.5 б) ($B_{max} = 4$) и Табела 5.8, Слика 5.6 б) ($B_{max} = 5$) споредени се соодветните веројатности за бит-грешка ($BER_{c4.3\_back}$, $BER_{c4.4\_back}$, $BER_{c4.3}$, $BER_{c4.4}$), добиени во истите експерименти. Исто како и претходно, со предложените методи со враќање, направени се експерименти сè додека не се добие $BER > p$.

p	$PER_{c4.3}$	$PER_{c4.3\_back}$	$PER_{c4.4}$	$PER_{c4.4\_back}$
0.02	0	0	0	0
0.03	0	0	0	0
0.04	0.0006	0	0.0006	0.0003
0.05	0.0026	0.0003	0.0026	0.0017
0.06	0.0034	0.0006	0.0029	0.0026
0.07	0.0094	0.0037	0.0063	0.0029
0.08	0.0209	0.0114	0.0091	0.0049
0.09	0.0437	0.0263	0.0189	0.0131
0.10	0.0906	0.0617	0.0277	0.0160
0.11	0.1649	0.1140	0.0534	0.0417
0.12	0.2666	0.2020	0.0977	0.0703
0.13	/	0.2974	0.1537	0.1134
0.14	/	/	0.2429	0.1860
0.15	/	/	0.3631	0.2903
0.16	/	/	/	0.3940
0.17	/	/	/	0.5340

Табела 5.5: Експериментални резултати за PER за $B_{max} = 4$ со и без враќање



а) б)

Слика 5.5: PER и BER без и со враќање за $B_{max} = 4$

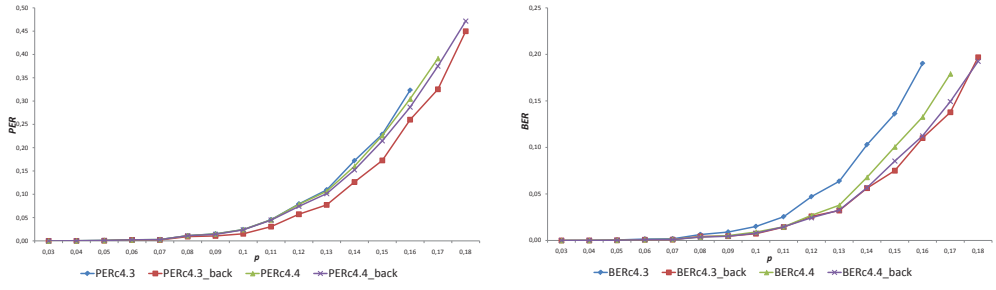
p	$BER_{c4.3}$	$BER_{c4.3\_back}$	$BER_{c4.4}$	$BER_{c4.4\_back}$
0.02	0	0	0	0
0.03	0	0	0	0
0.04	0.0003	0	0.0003	0.0001
0.05	0.0013	0.0002	0.0013	0.0008
0.06	0.0019	0.0003	0.0017	0.0012
0.07	0.0048	0.0020	0.0033	0.0014
0.08	0.0090	0.0044	0.0049	0.0021
0.09	0.0206	0.0124	0.0103	0.0056
0.10	0.0395	0.0287	0.0136	0.0057
0.11	0.0771	0.0554	0.0279	0.0191
0.12	0.1214	0.0983	0.0515	0.0285
0.13	/	0.1449	0.0828	0.0459
0.14	/	/	0.1294	0.0745
0.15	/	/	0.2009	0.1184
0.16	/	/	/	0.1586
0.17	/	/	/	0.2225

Табела 5.6: Експериментални резултати за BER за $B_{max} = 4$ со и без враќање

Од резултатите во Табела 5.5 и Табела 5.6 може да се заклучи дека за $B_{max} = 4$ за двата алгоритми со предложеното враќање се добива подобрување на PER и BER за сите вредности на p. Вредностите за BER добиени со АД4П#4 алгоритамот со враќање се приближно двапати помали од соодветните вредности добиени без методот со враќање.

p	$PER_{c4.3}$	$PER_{c4.3\_back}$	$PER_{c4.4}$	$PER_{c4.4\_back}$
0.03	0	0	0	0
0.04	0.00029	0	0.00029	0.00029
0.05	0.00114	0.00029	0.00086	0.00086
0.06	0.00229	0.00200	0.00229	0.00229
0.07	0.00286	0.00200	0.00286	0.00257
0.08	0.01114	0.00914	0.01114	0.01114
0.09	0.01514	0.01057	0.01486	0.01400
0.10	0.02429	0.01514	0.02429	0.02371
0.11	0.04543	0.03057	0.04514	0.04457
0.12	0.07914	0.05714	0.07829	0.07400
0.13	0.10943	0.07714	0.10629	0.10143
0.14	0.17200	0.12629	0.16029	0.15229
0.15	0.22857	0.17257	0.22600	0.21457
0.16	0.32314	0.26000	0.30457	0.28686
0.17	/	0.32514	0.39143	0.37486
0.18	/	0.44971	/	0.47143

Табела 5.7: Експериментални резултати за PER за $B_{max} = 5$ со и без враќање



а) б)

Слика 5.6: PER и BER без и со враќање за $B_{max} = 5$

p	$BER_{c4.3}$	$BER_{c4.3\_back}$	$BER_{c4.4}$	$BER_{c4.4\_back}$
0.03	0	0	0	0
0.04	0.00011	0	0.00007	0.00007
0.05	0.00049	0.00015	0.00025	0.00025
0.06	0.00152	0.00080	0.00087	0.00080
0.07	0.00178	0.00087	0.00113	0.00097
0.08	0.00636	0.00437	0.00378	0.00361
0.09	0.00899	0.00463	0.00521	0.00453
0.10	0.01505	0.00724	0.00906	0.00778
0.11	0.02546	0.01437	0.01472	0.01465
0.12	0.04704	0.02592	0.02698	0.02438
0.13	0.06365	0.03212	0.03789	0.03256
0.14	0.10289	0.05622	0.06789	0.05675
0.15	0.13609	0.07500	0.10059	0.08532
0.16	0.19017	0.11002	0.13277	0.11241
0.17	/	0.13777	0.17911	0.14929
0.18	/	0.19689	/	0.19246

Табела 5.8: Експериментални резултати за BER за $B_{max} = 5$ со и без враќање

Исто така, од резултатите во Табела 5.7 и Табела 5.8 може да се види дека за $B_{max} = 5$, со предложеното враќање, подобро подобрување за PER и BER се добива со користење на АД4П#3 алгоритмот. Уште повеќе, за сите вредности на p, $PER_{c4.3\_back}$ се помали од $PER_{c4.4\_back}$, иако во експериментите без враќање овие веројатности се скоро идентични. Причина за тоа е зголемениот кардинален број на множествата со кандидати за декодирање во АД4П#4 алгоритмот. Затоа, неуспешните декодирања со *грешка-повеќе-кандидати* завршуваат со голем број на елементи во множествата добиени во последната итерација. Оттука, со користење на $B_{max}-1$ во првата поништена итерација, кардиналниот број на овие множества не се намалува на 1. Но, ако се користи $B_{max} - 2$, тогаш точната порака се исфрла од множествата S.

5.8 Визуелна илустрација на експериментите

Во ова поглавје испитани се перформансите на ново предложениот АД4П#3 алгоритам (со предложената комбинација на двата методи со враќање со кој се добиени најдобри резултати) при декодирање на слики пренесени низ бинарен симетричен канал. За таа цел, направени се експерименти со Слика 4.1. Во овие експерименти користен е бинарен симетричен канал и сликата е кодирана/декодирана со користење на кодот (72,576) со рата $R = 1/8$ и параметрите со кои се добиени најдобрите резултати за АД4П алгоритмите. Експериментите се направени со користење на $B_{max} = 5$ во процесот на декодирање и следните вредности на веројатноста за бит-грешка во каналот: $p = 0.05$, $p = 0.08$, $p = 0.11$, $p = 0.14$ и $p = 0.17$ (за $p > 0.17$ вредностите на $BER_{c4.3\_back}$ се поголеми од p). Во експериментите користени се истите решенија за неуспешните декодирања како и во Глава 4.

Исто така, за споредба направени се експерименти и со Рид-Соломон код. Во експериментите со Рид-Соломон кодовите (RSC) користена е скратена верзија на кодот RSC(127,57) [63]. Тоа е кодот RSC(80,10) дефиниран над полето на Галоа GF(2^7) со примитивен полином $p(x) = 1 + X + X^7$. Овој скратен RSC има иста рата 1/8 и приближно иста должина на пораки и кодни зборови ((70, 560)) како и разгледуваниот RCBQ.

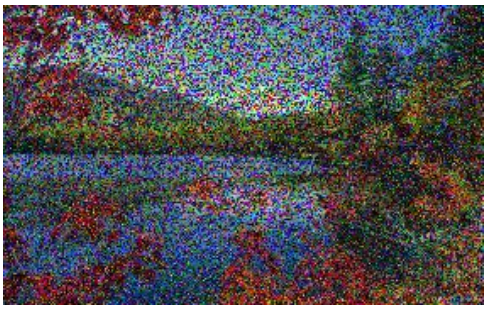
Сликите добиени за разгледуваните вредности на веројатноста p за бит-грешка во бинарен симетричен канал дадени се на Слики 5.7 - 5.11. На овие слики под а) дадени се сликите добиени со користење на RCBQ со АД4П#3 алгоритмот со предложената комбинација на двата методи со враќање. Под б) претставени се сликите кодирани/декодирани со Рид-Соломон кодот.

а) б)

Слика 5.7: Слики за $p = 0.05$

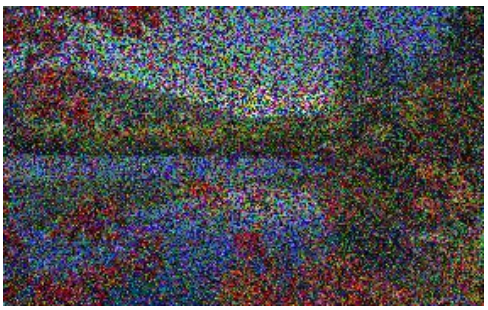

а) б)

Слика 5.8: Слики за $p = 0.08$

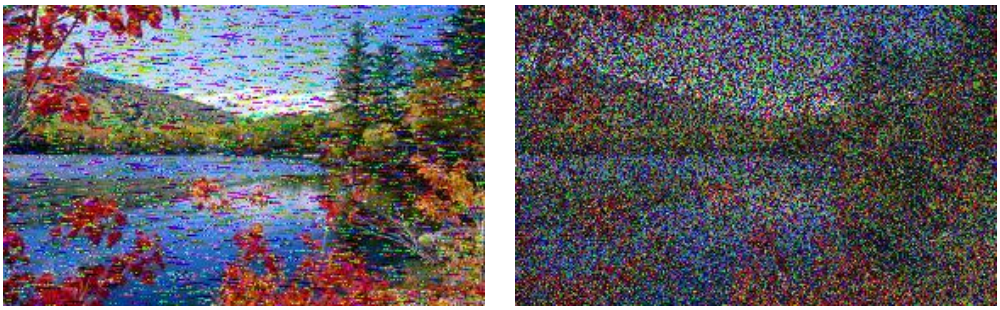
а) б)

Слика 5.9: Слики за $p = 0.11$

а) б)

Слика 5.10: Слики за $p = 0.14$



а) б)

Слика 5.11: Слики за $p = 0.17$

Од сликите, може да се заклучи дека за сите разгледани вредности на p, сликите добиени со користење на RCBQ со АД4П#3 (со враќање) се почисти од соодветните слики добиени со користење на Рид-Соломон кодот. Исто така, за $p \geqslant 0.08$ кодирањето/декодирањето на сликата со Рид-Соломон кодот нема смисла ($BER > p$). Но, со RCBQ со АД4П#3 алгоритамот за сите $p \leqslant 0.17$ се добива $BER_{c\_back} < p$ (како и во Табела 5.8).

Заклучок

Во оваа глава дефиниран е нов алгоритам за кодирање/декодирање за кодовите базирани на квазигрупи, наречен алгоритам-за-декодирање-со-4пресеци (АД4П алгоритам или 4-Sets-Cut-Decoding, со четири различни методи за генерирање на множествата со кандидати за декодирање). Со користење на овој алгоритам за код со рата 1/8, декодирањето е 6.3 пати побрзо отколку со стандардниот алгоритам, а за некои вредности на p се добиваат 24 пати помали вредности за веројатноста за пакет-грешка. Исто така, детално се испитани перформансите на различните алгоритми за декодирање (стандардниот, АДП и АД4П алгоритмите) за код со рата $R = 1/8$ и разгледано е подобрувањето на перформансите на кодот со примена на предложените методи за намалување на неуспешните декодирања во новиот алгоритам. На крајот од оваа глава испитани се перформансите на случајните кодови базирани на квазигрупи со АД4П#3 алгоритмот при кодирање/декодирање на слики пренесени низ бинарен симетричен канал. Исто така, овие резултати се споредени со соодветни резултати добиени со Рид-Соломон код.

Глава 6

Теориски резултати за новите алгоритми на случајните кодови базирани на квазигрупи

Во оваа глава ќе бидат дадени некои теориски резултати за новите алгоритми (АДП и АД4П алгоритамот) на RCBQ кои се предложени во оваа теза.

Ќе биде изведена теориска горна граница за веројатностите за пакет-грешка (без неуспешните декодирања со *грешка-повеќе-кандидати*) за АДП и АД4П алгоритмите. Ќе биде покажано дека оваа горна граница е всушност еднаква на теориската веројатност за пакет-грешка PER_t за стандардниот алгоритам на RCBQ. Прво, теориски ќе биде докажана горната граница за PER, а потоа ќе биде и експериментално потврдена.

Исто така, на крајот од оваа глава ќе бидат дадени приближни формули за кардиналниот број на редуцираноте множества со кандидати за декодирање добиени во процесот на декодирање со новите алгоритми.

6.1 Теориска горна граница за веројатноста за пакет-грешка кај новите алгоритми

Формула за теориската веројатност за пакет-грешка за стандардниот алгоритам дадена е во Теорема 2.1 ([30]). Но, од доказот на оваа теорема, кој е даден во [30], јасно е дека во оваа формула не се предвидени неуспешните декодирања со *грешка-повеќе-кандидати*. Бројот на овие грешки многу зависи од избраната квазигрупа, од патернот за додавање на редундантноста и од должината на почетниот клуч. Затоа, многу е тешко за овие кодови да се

предвиди бројот на *грешки-повеќе-кандидати*.

Во ова поглавје ќе биде докажано дека формулата за PER_t во Теорема 2.1 (за иста рата на кодот, иста должина на блоковите и вредност на B_{max} во процесот на декодирање) ја дава горната граница за PER добиена со АДП и АД4П алгоритмите. Од истите причини како и претходно, во следната теорема и нејзиниот доказ ќе биде разгледувана веројатноста само за *грешка-празно-множество* и *некоригирана-грешка* (декодирање кое успешно завршило, но декодираната порака не е точна).

Теорема 6.1. *Веројатноста за пакет-грешка* (*без грешки-повеќе-кандидати*) *добиена со стандардниот алгоритам за декодирање на RCBQ е горна граница за веројатноста за пакет-грешка добиена со АДП и АД4П алгоритмите* (*за иста рата на код, иста должина на блокови и вредност на* B_{max} *во процесот на декодирање*).

Доказ. Веројатноста дека максимум t битови во еден блок $D^{(i)}$ од r симболи не се точно пренесени е

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

каде p е веројатноста за бит-грешка во бинарен симетричен канал.

Настаните A_i : "ќе се појават максимум B_{max} грешки во блок $D^{(i)}$", за секое i, се независни, а веројатноста дека точниот блок $C^{(i)}$ ќе се содржи во множеството H_i е $P(p; B_{max}) = 1 - q_B$.

Бидејќи, во АДП алгоритамот, за да се добие код со рата R, се користат два кода со рата $2R$, бројот на итерации во процесот на декодирање е двапати помал. Затоа, ако за даден код во процесот на декодирање со стандардниот алгоритам има s итерации, тогаш за истиот код бројот на итерации во двата паралелни процеси на декодирање во АДП алгоритмот е $s/2$.

Кога се користи АДП алгоритамот, точниот блок $L^{(i)}$, $i = 1, 2, ..., s/2$ ќе припаѓа (како втор дел од некој елемент) во редуцираните множества $S_i^{(1)}$ и $S_i^{(2)}$ ако и во двата процеси на декодирање точните блокови $C^{(i)}$ и $C^{(s/2+i)}$ (од двата кодни збора) припаѓаат во соодветните множества $H_i^{(1)}$ и $H_i^{(2)}$. Веројатноста за тој настан е $(1-q_B)(1-q_B)$ за секое i, бидејќи двата процеси на декодирање се независни. Затоа, веројатноста секој точен блок $L^{(i)}$ да припаѓа во редуцираните множества $S_i^{(1)}$ и $S_i^{(2)}$ за $i = 1, 2, ..., s/2$ е $((1-q_B)(1-q_B))^{s/2} = (1-q_B)^s$. Оттука следува дека веројатноста за пакет-грешка е $1 - (1-q_B)^s = PER_t$, каде PER_t е теориската веројатност за

пакет-грешка, ако за истиот код (иста рата, должина на блокови и B_{max}) се користи стандардниот алгоритам за декодирање.

Но, според правилото за декодирање на АДП алгоритамот, кога едно од множествата со кандидати за декодирање е празно, декодирањето на пораката продолжува со користење на стандардниот алгоритам. Во тој случај процесот на декодирање ќе заврши успешно, ако точниот блок се содржи во непразното множество. Затоа, веројатноста $1-(1-q_B)^s$ (која е еднаква на веројатноста PER_t во Теорема 2.1) претставува горна граница за веројатноста за пакет-грешка (без *грешки-повеќе-кандидати*) добиена со АДП алгоритамот.

На сличен начин, се добива истата горна граница и за АД4П алгоритамот. Во овој алгоритам има четири процеси на декодирање и за код со рата R се користат четири кодови со рата $4R$. Затоа, во АД4П алгоритамот имаме четири пати помал број на итерации ($s/4$). Во сите итерации, точниот блок $L^{(i)}$, $i = 1, 2, ..., s/4$ се содржи (како втор дел од некој елемент) во редуцираните множества $S_i^{(1)}$, $S_i^{(2)}$, $S_i^{(3)}$ и $S_i^{(4)}$, ако точните блокови $C^{(i)}$, $C^{(s/4+i)}$, $C^{(s/2+i)}$ и $C^{(3s/4+i)}$ (од четирите кодни зборови) припаѓаат во соодветните множества $H_i^{(1)}$, $H_i^{(2)}$, $H_i^{(3)}$ и $H_i^{(4)}$. Исто како и претходно, горната граница за веројатноста за пакет-грешка е

$$1 - ((1 - q_B)(1 - q_B)(1 - q_B)(1 - q_B))^{s/4} = 1 - (1 - q_B)^s = PER_t.$$

Бидејќи процесот на декодирање на пораката продолжува и кога барем едно од множествата со кандидати за декодирање не е празно, оваа веројатност повторно е горна граница за PER (без *грешки-повеќе-кандидати*) .

Со оваа теорема докажавме дека со новите алгоритми за декодирање има подобрување не само во брзината на декодирање, туку исто така намален е и бројот на неуспешни декодирања (кога горната граница не е достигната).

6.2 Експериментална потврда на теориската горна граница за веројатноста за пакет-грешка

За експериментална потврда на Теорема 6.1, која е докажана во претходното поглавје, ќе ги споредиме вредностите на PER_t со експерименталните резултати за PER без неуспешните декодирања со *грешка-повеќе-кандидати* добиени во експериментите со АДП и АД4П алгоритмите.

6.2.1 Споредба за рата $R = 1/4$

Прво, ќе бидат споредени вредностите на PER_t со експериментални резултати за веројатностите за пакет-грешка без *грешки-повеќе-кандидати* PER_{c_1} добиени со АДП алгоритмот и PER_{s_1} добиени со стандардниот алгоритам за код (72,288) со рата $R = 1/4$. Овие резултати се за кодовите чии параметри се дадени во Поглавје 3.3 и Поглавје 2.4.2, соодветно. Во овие експерименти во процесот на декодирање користени се блокови од 4 нибли (16 бита). Затоа, во формулата за теориската веројатност за пакет-грешка PER_t, веројатностите q_B се пресметуваат со користење на следната формула:

$$q_B = 1 - P(p; B_{max}) = 1 - \sum_{k=0}^{B_{max}} \binom{16}{k} p^k (1-p)^{16-k}.$$

Бројот на блокови во кодните зборови, т.е., бројот на итерации во процесот на декодирање е 18.

Во Табела 6.1, дадени се експерименталните резултати за PER_{c_1} и PER_{s_1} заедно со вредностите на PER_t за различни вредности на веројатноста за бит-грешка p на бинарен симетричен канал.

p	PER_t	PER_{c_1}	PER_{s_1}
0.02	0.000209	0	0.000313
0.03	0.001447	0.000857	0.002188
0.04	0.005541	0.003429	0.005313
0.05	0.015319	0.014286	0.014688
0.06	0.034361	0.025714	0.035938
0.07	0.066467	0.057714	0.065000
0.08	0.114889	0.104000	0.112500
0.09	0.181424	0.155429	0.186875

Табела 6.1: Споредба на теориската горна граница со експерименталните резултати за код (72,288) со рата $R = 1/4$

Од резултати дадени во Табела 6.1 може да се види дека за сите вредности на p, $PER_{c\_1} < PER_t$, т.е., теориската веројатност за пакет-грешка е горна граница за PER (без *грешки-повеќе-кандидати*) добиена со АДП алгоритмот. Исто така, со споредба на експериментални резултати добиени со стандардниот алгоритам (PER_{s_1}) и АДП алгоритамот (PER_{c_1}) може да се заклучи дека со користење на АДП алгоритмот се добиваат помал број на неуспешни декодирања со *грешка-празно-множество* и *некоригирана-грешка*. Имено, за сите вредности на p, $PER_{s\_1} > PER_{c\_1}$.

6.2.2 Споредба за рата $R = 1/8$

Тука ќе бидат споредени вредностита на PER_t со експерименталните резултати за веројатноста за пакет-грешка (без *грешки-повеќе-кандидати*) добиени со стандардниот алгоритам, АДП алгоритамот и АД4П алгоритамот за код (72, 576) со рата $R = 1/8$. Во Поглавје 5.6, дадени се резултатите за овие кодови и користените параметри. Во формулата за PER_t, за веројатностите q_B се користи истата формула, како и претходно, бидејќи во експериментите за рата $R = 1/8$ во процесот на декодирање повторно се користат блокови од 4 нибли (16 бита). Но, сега бројот на блокови во кодните зборови, т.е., бројот на итерации е 36.

Вредностите за PER_t заедно со експерименталните резултати за PER (без *грешки-повеќе-кандидати*) кои се добиени со стандардниот алгоритам (PER_{s_1}), АДП алгоритамот (PER_{c_1}) и АД4П#1 алгоритамот ($PER_{c4.1\_1}$) претставени се во Табела 6.2 (за $B_{max} = 4$) и Табела 6.3 (за $B_{max} = 5$). Горната граница за PER е споредена само со резултатите добиени со АД4П#1 алгоритамот, бидејќи во другите верзии на овој алгоритам има дополнителни методи за намалување на бројот на неуспешни декодирања со *грешка-празно-множество* и нивните веројатности се помали од $PER_{c4.1\_1}$.

Со споредба на вредностите за PER_t со експерименталните резултати за PER_{c_1} и $PER_{c4.1\_1}$ може да се заклучи дека Теорема 6.1 е точна и за рата $R = 1/8$. Имено, за сите вредности на p и двете разгледувани вредности на B_{max} со двата нови алгоритми (АДП и АД4П алгоритамот) добиено е $PER_{c\_1} < PER_t$ и $PER_{c4.1\_1} < PER_t$.

p	PER_t	PER_{c_1}	$PER_{c4.1\_1}$	PER_{s_1}
0.02	0.0004	0.0000	0.0000	0.0009
0.03	0.0029	0.0017	0.0020	0.0026
0.04	0.0111	0.0074	0.0086	0.0103
0.05	0.0304	0.0134	0.0254	0.0320
0.06	0.0675	0.0371	0.0600	0.0660
0.07	0.1285	0.0643	0.1129	0.1294
0.08	0.2166	0.1257	0.1906	0.2200

Табела 6.2: Споредба на теориската горна граница со експерименталните резултати за $B_{max} = 4$ за код (72,576) со рата $R = 1/8$

p	PER_t	PER_{c_1}	$PER_{c4.1\_1}$	PER_{s_1}
0.03	0.00016	0.00007	0.00007	0.00029
0.04	0.00083	0.00057	0.00057	0.00086
0.05	0.00291	0.00229	0.00286	0.00314
0.06	0.00793	0.00514	0.00714	0.00743
0.07	0.01818	0.01286	0.01571	0.02086
0.08	0.03667	0.02829	0.03171	0.04171
0.09	0.06685	0.04400	0.05914	0.06257
0.10	0.11209	0.08029	0.09486	0.11457
0.11	0.17491	0.12171	0.15086	0.17857
0.12	0.25605	0.14314	0.22229	/

Табела 6.3: Споредба на теориската горна граница со експерименталните резултати за $B_{max} = 5$ за код (72,576) со рата $R = 1/8$

Исто така, од резултатите во Табела 6.2 и Табела 6.3, може да се види дека за сите вредности на веројатноста p за бит-грешка во бинарен симетри-

чен канал, $PER_{s\_1} > PER_{c\_1}$ и $PER_{s\_1} > PER_{c4.1\_1}$. Ова потврдува дека и за рата $R = 1/8$, со новите алгоритми се намалува бројот на неуспешни декодирања со *грешка-празно-множество* и *некоригирана-грешка*. Овие разлики се поголеми за поголеми вредности на p. Освен тоа, со третата и четвртата верзија на АД4П алгоритамот се добива уште поголемо подобрување на перформансите во однос на бројот на неупешни декодирања со *грешка-празно-множество*.

6.3 Приближни формули за кардиналниот број на редуцираните множества со кандидати за декодирање

Во [30], авторите даваат приближна формула за пресметување на кардиналиот број $|S_i|$ на множествата со кандидати за декодирање добиени со стандардниот алгоритам на RCBQ. Приближната формула за $|S_i|$ тие ја добиваат на следниот начин.

Се разгледува код (N_{block}, N) со рата R и нека $L = L^{(1)}L^{(2)}...L^{(s)}$ е редундантната порака. Нека N_i е бројот на информациски нибли во подблокот $L^{(i)}$ од r нибли. Нека B_{checks} е кардиналниот број на множествата H_i, за $i \geqslant 1$. Ако се претпостави дека стринговите добиени како излез од инверзниот алгоритам на алгоритмот за кодирање се (скоро) случајни, тогаш веројатноста дека ќе се добие стринг со $r - N_i$ нулти нибли (како излез) е $\frac{1}{2^{4(r-N_i)}}$. Оттука, приближната формула за (очекуваната) кардиналност на првото множество S_1 е:

$$|S_1| \approx \frac{B_{checks}}{2^{4(r-N_1)}}.$$

Бидејќи во i-тата итерација (за $i > 1$), инверзниот алгоритам на алгоритмот за кодирање се применува со клучот од секој елемент во множеството S_{i-1} и секој елемент од множеството H_i, кардиналниот број на множествата S_i може приближно да биде пресметан со користење на следната формула

$$|S_i| \approx |S_{i-1}|\frac{B_{checks}}{2^{4(r-N_1)}}.$$

Овие приближни формули за кардиналниот број на множествата со кандидати за декодирање даваат подобра апроксимација, ако за пресметување на $|S_i|$ се користат експериментално добиените вредности за $|S_{i-1}|$ наместо приближните (добиени во претходниот чекор од апроксимацијата).

Тука ќе бидат изведени слични приближни формули за кардиналниот број на редуцираните множества со кандидати за декодирање добиени со АДП и АД4П алгоритмите. Овие приближни формули се добиени со користење на истата претпоставка, како и претходно, дека процесот на формирање на множествата со кандидати за декодирање е скоро случаен.

Нека $S_1^{(1)}$ и $S_1^{(2)}$ се множествата добиени во првата итерација од АДП алгоритамот пред предложената редукција во Чекор 4 и Чекор 5 и нека $\left|S_1^{(1)}\right| = n_1^{(1)}$ и $\left|S_1^{(2)}\right| = n_1^{(2)}$. Како што напоменавме погоре, се претпоставува дека множествата со кандидати за декодирање се случајни. Ако N_1 е бројот на информациски нибли во подблокот $L^{(1)}$ од r нибли, тогаш стринговите (вториот дел од елементите) во множествата $S_1^{(j)}, j = 1, 2$ може да бидат разгледувани како стрингови од $4N_1$ бита. За секој стринг k од $4N_1$ бита, веројатноста p_1 дека k е стрингот во вториот дел од некој елемент од множеството $S_1^{(1)}$, т.е., $k \in V_1$ (алгоритмот даден во Поглавје 3.2) е:

$$p_1 = P\{k \in V_1\} = 1 - \left(1 - \frac{1}{2^{4N_1}}\right)^{n_1^{(1)}}.$$

Слично, веројатноста p_2 дека стрингот k е втор дел од некој елемент во множеството $S_1^{(2)}$ е:

$$p_2 = P\{k \in V_2\} = 1 - \left(1 - \frac{1}{2^{4N_1}}\right)^{n_1^{(2)}}.$$

Оттука, веројатноста дека $k \in V = V_1 \cap V_2$, т.е., дека елементите кои во вториот дел го имаат стрингот k ќе припаѓаат во редуцираните множества е:

$$P\{k \in V_1 \cap V_2\} = p_1 p_2 = \left(1 - q^{n_1^{(1)}}\right)\left(1 - q^{n_1^{(2)}}\right),$$

каде што $q = 1 - \dfrac{1}{2^{4N_1}}$.

Според тоа, приближната (заради претпоставката за случајност) формула за (очекуваниот) кардинален број на множеството V во Чекор 4 од АДП алгоритмот, т.е., кардиналниот број на множествата со кандидати за декодирање $S_1^{(1)}$ и $S_1^{(2)}$ после редуцирањето во Чекор 5 е:

$$\left|S_1^{(j)}\right| \approx \sum_k P\{k \in V_1 \cap V_2\} = 2^{4N_1}\left(1 - q^{n_1^{(1)}}\right)\left(1 - q^{n_1^{(2)}}\right), \;\; j = 1, 2.$$

Сега, нека $S_i^{(1)}$ и $S_i^{(2)}$ се множествата добиени во Чекор 3 (пред редукцијата) од i-тата итерација на процесот на декодирање. Претпоставуваме дека бројот на елементи во $S_i^{(1)}$ $\left(\text{и } S_i^{(2)}\right)$ кои се добиени од секој елемент во соодветното множество со кандидати за декодирање од претходната итерација е еднаков. Ова значи дека во множеството $S_i^{(1)}$ $\left(\text{и } S_i^{(2)}\right)$ имаме приближно еднаков број на елементи со стрингови кои имаат еднаков префикс. Експерименталните резултати покажаа дека ова е (приближно) точно. Нека m_{i-1} е бројот на елементи во редуцираните множества со кандидати за декодирање кои се добиени во $(i-1)$-та итерација и $\left|S_i^{(1)}\right| = n_i^{(1)}$ и $\left|S_i^{(2)}\right| = n_i^{(2)}$. Тогаш, со користење на горната претпоставка, се добива дека во множествата $S_i^{(j)}$, т.е., множествата V_j (во Чекор 4), за $j = 1, 2$, има m_{i-1} класи со $\dfrac{n_i^{(j)}}{m_{i-1}}$ елементи кои во вториот дел имаат стрингови со ист префикс од $r(i-1)$ нибли. Значи, елементите во секоја од овие класи се разликуваат само во последните r нибли, т.е., попрецизно во N_i нибли, каде што N_i е бројот на информациски нибли во подблокот $L^{(i)}$ од редундантната порака. Пресекот $V = V_1 \cap V_2$ (во Чекор 4) ќе биде непразен само за елементите од соодветните класи (од елементи со ист префикс) во множествата V_1 и V_2. Користејќи го истиот метод, како за множествата во првата итерација, се добива дека кардиналниот број на пресекот на соодветните класи од множествата V_1 и V_2 приближно е:

$$2^{4N_i}\left(1 - q^{\frac{n_i^{(1)}}{m_{i-1}}}\right)\left(1 - q^{\frac{n_i^{(2)}}{m_{i-1}}}\right),$$

каде што $q = 1 - \dfrac{1}{2^{4N_i}}$. Бидејќи има m_{i-1} класи од елементи со ист префикс од $r(i-1)$ нибли, приближната формула за кардиналниот број на редуцираните множества со кандидати за декодирање во i-тата итерација е:

$$\left|S_i^{(j)}\right| \approx m_{i-1}\left(2^{4N_i}\left(1 - q^{\frac{n_i^{(1)}}{m_{i-1}}}\right)\left(1 - q^{\frac{n_i^{(2)}}{m_{i-1}}}\right)\right), \quad j = 1, 2.$$

На сличен начин се изведуваат и приближни формули за кардиналноста на редуцираните множества добиени со АД4П#1 алгоритамот. Единствената разлика е бројот на множествата во овој алгоритам. Во него има 4 множества наместо 2. Со користење на истите ознаки како погоре и во Поглавје 5.2 се добиваат следните приближни формули:

– за кардиналниот број на редуцираните множества со кандидати за декодирање во првата итерација

$$\begin{aligned}\left|S_1^{(j)}\right| &\approx \sum_k P\{k \in V_1 \cap V_2 \cap V_3 \cap V_4\} \\ &= 2^{4N_1}\left(1-q^{n_1^{(1)}}\right)\left(1-q^{n_1^{(2)}}\right)\left(1-q^{n_1^{(3)}}\right)\left(1-q^{n_1^{(4)}}\right),\end{aligned}$$

за $j = 1, 2, 3, 4$;

– за кардиналниот број на редуцираните множества со кандидати за декодирање во i итерација, $i > 1$

$$\left|S_i^{(j)}\right| \approx m_{i-1}\left(2^{4N_i}\left(1-q^{\frac{n_i^{(1)}}{m_{i-1}}}\right)\left(1-q^{\frac{n_i^{(2)}}{m_{i-1}}}\right)\left(1-q^{\frac{n_i^{(3)}}{m_{i-1}}}\right)\left(1-q^{\frac{n_i^{(4)}}{m_{i-1}}}\right)\right),$$

$j = 1, 2, 3, 4.$

Со овие приближни формули се добива добра апроксимација за кардиналниот број на множествата со кандидати за декодирање после редуцирањето во новите алгоритми за RCBQ. Исто како и кај формулите за стандардниот алгоритам, изведените формули даваат подобра апроксимација ако за пресметување на $|S_i|$ се користат експериментално добиените вредности за $|S_{i-1}|$ наместо приближните. Прецизноста е скоро иста како и кај формулите за стандардниот алгоритам дадени во [30] (најголемото добиено отстапување е 10).

Со користење на овие апроксимации за кардиналниот број на множествата со кандидати за декодирање, може да се провери дали некој патерн за додавање на редундансата е добар. Освен тоа, брзината на декодирање и бројот на неуспешни декодирања со *грешка-повеќе-кандидати*, исто така, зависат од бројот на елементи во множествата S. Од формулите, изведени во ова поглавје, јасно е дека предложеното редуцирање на множествата со кандидати за декодирање, во новите алгоритми, значително го намалува бројот на елементи во множествата.

Заклучок

Во оваа глава е докажана Теорема 6.1 за теориската веројатност за пакет-грешка добиена со новите алгоритми за кодирање/декодирање дефинирани

во претходните глави и се изведени приближни формули за кардиналниот број на множествата со кандидати за декодирање после редукцијата на нивните елементи предложена во новите алгоритми. Со тоа дадени се теориски (математички) методи за избор на добри параметри во новите предложени алгоритми за RCBQ. Исто така, со изведените формули е докажано дека со новите алгоритми е добиено подобрување на перформансите на овие кодови.

Глава 7

Парастрофна квазигрупна трансформација и нејзина примена во криптографија

Квазигрупите и квазигрупните трансформации се многу погодни за конструкција на криптографски примитиви, кодови кои поправаат грешки и кодови кои откриваат грешки. Причина за тоа е нивната структура, нивниот голем број, својствата на квазигрупните трансформации и.т.н. Квазигрупните стринг транформации и нивните својства се разгледувани во неколку трудови.

Во Глава 1, опишана е E-трансформацијата и разгледани се нејзините својства корисни за примена во криптографија. Користејќи ги парастрофите на квазигрупите, во [35], Крапеж дава идеја за нова квазигрупна стринг трансформација која исто така може да биде применета во криптографија. Во оваа глава ние ќе дефинираме модификација на оваа квазигрупна трансформација. Прво ќе биде опишана новата квазигрупна трансформација наречена парастрофна квазигрупна трансформација, а потоа ќе бидат разгледани нејзините криптографски својства.

7.1 Парастрофи

Секоја квазигрупата $(Q, *)$ има множество од пет квазигрупи, наречени парастрофи. Парастрофите на операцијата $*$ кои се означуваат со $/, \backslash, \cdot, //, \backslash\backslash$, дефинирани се во Табела 7.1 .

За секое $f \in \{*, \backslash, /, \cdot, //, \backslash\backslash\}$, (Q, f) претставува квазигрупа.

Парастрофни операции				
$x\backslash y = z$	$\iff$	$x * z = y$		
$x/y = z$	$\iff$	$z * y = x$		
$x \cdot y = z$	$\iff$	$y * x = z$		
$x//y = z$	$\iff$	$y/x = z$	$\iff$	$z * x = y$
$x\backslash\backslash y = z$	$\iff$	$y\backslash x = z$	$\iff$	$y * z = x$

Табела 7.1: Парастрофите на квазигрупна операција $*$

Пример 7.1. Нека $Q = \{1, 2, 3, 4\}$. Во (7.1) даден е пример на квазигрупа $(Q, *)$ од ред 4 и нејзините парастрофи.

$*$	1	2	3	4
1	1	2	3	4
2	2	1	4	3
3	4	3	1	2
4	3	4	2	1

$/$	1	2	3	4
1	1	2	3	4
2	2	1	4	3
3	4	3	1	2
4	3	4	2	1

$\backslash$	1	2	3	4
1	1	2	3	4
2	2	1	4	3
3	3	4	2	1
4	4	3	1	2

(7.1)

$\cdot$	1	2	3	4
1	1	2	4	3
2	2	1	3	4
3	3	4	1	2
4	4	3	2	1

$//$	1	2	3	4
1	1	2	4	3
2	2	1	3	4
3	3	4	1	2
4	4	3	2	1

$\backslash\backslash$	1	2	3	4
1	1	2	3	4
2	2	1	4	3
3	3	4	2	1
4	4	3	1	2

Може да забележиме дека не мора сите парастрофи на една квазигрупа да бидат различни. Може да видиме дека од 6-те квазигрупи дадени во Пример 7.1, само три се различни. Користејќи го бројот на различни парастрофи за секоја квазигрупа, во Поглавје 7.3.1 ќе дадеме класификација на квазигрупите од ред 4.

Во оваа теза ќе ги користиме следните ознаки за квазигрупната операција $*$ и нејзините парастрофи:

$$f_1(x, y) = x * y, \quad f_2(x, y) = x\backslash y, \quad f_3(x, y) = x/y,$$
$$f_4(x, y) = x \cdot y, \quad f_5(x, y) = x//y, \quad f_6(x, y) = x\backslash\backslash y.$$

7.2 Парастрофна квазигрупна трансформација

Нека $A = \{1, \ldots, a\}$ е азбука од природни броеви ($a \geqslant 2$) и со $A^+ = \{x_1 \ldots x_k | \; x_i \in A, \; k \geqslant 1\}$ е означено множеството од сите конечни стрингови над A. Тогаш $A^+ = \bigcup\limits_{k \geqslant 1} A^k$, каде $A^k = \{x_1 \ldots x_k | \; x_i \in A\}$

Нека $M = x_1 x_2 \ldots x_k$ е влезната порака. Нека d_1 е случајно избран природен број така што $2 \leqslant d_1 < k$ и нека l е случајно избран елемент (лидер) од A. Исто така, нека $(A, *)$ е квазигрупа и $f_1, \ldots, f_6$ се сите нејзини парастрофни операции.

Со користење на трансформацијата E, за избрани l, d_1 и квазигрупа $(A, *)$ ја дефинираме парастрофната квазигрупна трансформација $PE = PE_{l,d_1} : A^+ \to A^+$ на следниот начин.

На почеток, нека $q_1 = d_1$ биде должината на првиот блок, т.е., $M_1 = x_1 x_2 \ldots x_{q_1 - 1} x_{q_1}$. Нека, $s_1 = (d_1 \bmod 6) + 1$. Со примена на трансформацијата E на блокот M_1 со лидер l и квазигрупна операција f_{s_1}, се добива криптираниот блок

$$C_1 = y_1 y_2 \ldots y_{q_1 - 1} y_{q_1} = E_{f_{s_1}, l}(x_1 x_2 \ldots x_{q_1 - 1} x_{q_1}).$$

Понатаму, користејќи ги последните два симболи во C_1 се пресметува бројот $d_2 = 4y_{q_1 - 1} + y_{q_1}$ кој ја одредува должината на следниот блок. Нека, $q_2 = q_1 + d_2$, $s_2 = (d_2 \bmod 6) + 1$ и $M_2 = x_{q_1 + 1} \ldots x_{q_2 - 1} x_{q_2}$. Со примена на трансформацијата $E_{f_{s_2}, y_{q_1}}$, криптираниот блок C_2 е

$$C_2 = y_{q_1 + 1} \ldots y_{q_2 - 1} y_{q_2} = E_{f_{s_2}, y_{q_1}}(x_{q_1 + 1} \ldots x_{q_2 - 1} x_{q_2}).$$

Во општ случај, нека за дадено i, криптираните блокови $C_1, \ldots, C_{i-1}$ се добиени и нека d_i е пресметан со користење на последните два симболи во C_{i-1}, т.е., $d_i = 4y_{q_{i-1} - 1} + y_{q_{i-1}}$. Нека $q_i = q_{i-1} + d_i$, $s_i = (d_i \bmod 6) + 1$ и $M_i = x_{q_{i-1} + 1} \ldots x_{q_i - 1} x_{q_i}$. Со примена на трансформацијата $E_{f_{s_i}, y_{q_{i-1}}}$ на блокот M_i се добива криптираниот блок

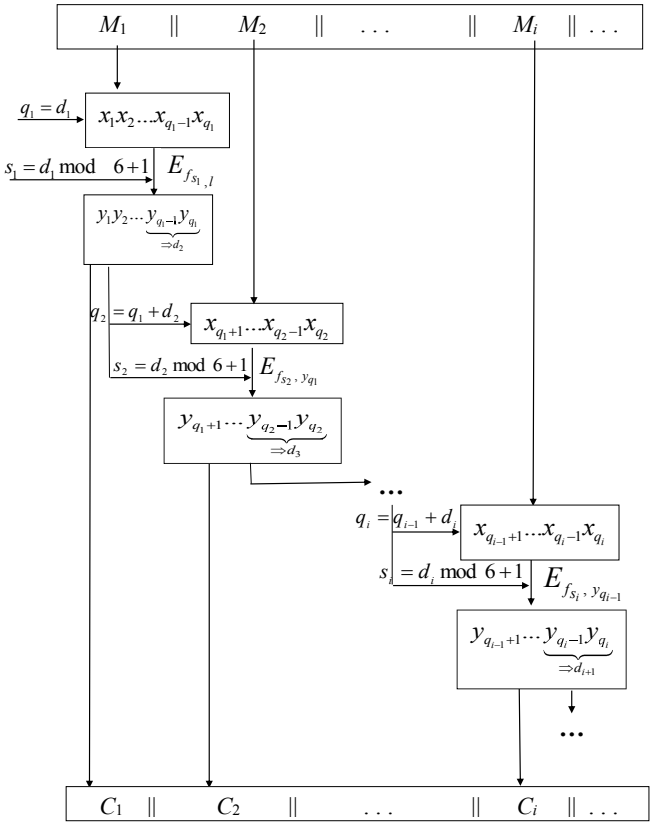
$$C_i = E_{f_{s_i}, y_{q_{i-1}}}(x_{q_{i-1} + 1} \ldots x_{q_i - 1} x_{q_i}).$$

Сега, парастрофната квазигрупна трансформација се дефинира на следниот начин:

$$PE_{l,d_1}(M) = PE_{l,d_1}(x_1 x_2 \ldots x_n) = C_1 || C_2 || \ldots || C_r, \tag{7.2}$$

каде $||$ е конкатенација на блокови. Да забележиме дека должината на последниот блок M_r може да биде помала од d_r. Тоа зависи од бројот на букви

во влезната порака. Шематскиот приказ на трансформацијата PE е даден на Слика 7.1.



Слика 7.1: Парастрофна трансформација PE

За произволна квазигрупа на множеството A, случајно избрани лидери $l_1, \ldots l_n$ и должини $d_1^{(1)}, \ldots, d_1^{(n)}$, дефинираме пресликувања PE_1, PE_2, …, PE_n како во (7.2) така што PE_i одговара на должината $d_1^{(i)}$ и лидерот l_i. Користејќи ги овие трансформации, ја дефинираме трансформацијата $PE^{(n)}$ на следниот начин:

$$PE^{(n)} = PE^{(n)}_{(l_n,d_1^{(n)}),\ldots,(l_1,d_1^{(1)})} = PE_n \circ PE_{n-1} \circ \cdots \circ PE_1,$$

каде $\circ$ е композиција на пресликувања.

Направени се многу експерименти со PE-трансформацијата користејќи различни начини за одредување на должината на следниот блок и квазигрупната операција во секоја итерација. Со анализа на добиените резултати заклучивме дека најдобри резултати се добиваат доколку за пресметување на должината на следниот блок и соодветната квазигрупна операција се користат последните два симболи од претходниот криптиран блок. Имено, доколку земеме само еден симбол, тогаш може да се добијат само 4 вредности за избор на квазигрупната операција, а за секоја квазигрупа има 6 парастрофи. Од друга страна, доколку се земат повеќе од два симболи за пресметување на должината на следниот блок и соодветната квазигрупна операција, заклучивме дека во тој случај парастрофните операции не се менуваат доволно често. Затоа, во овие случаи добивме резултати кои беа полоши во однос на фракталноста, т.е., не добивме зголемување на бројот на квазигрупи погодни за примена во криптографија (Поглавје 7.3).

7.3 Класификации на квазигрупите од ред 4 корисни во криптографија

Со користење на графички приказ на квазигрупно процесираните низи добиени со примена на квазигрупните стринг трансформации, Димитрова и Марковски во [17] даваат класификација на квазигрупите од ред 4 на фрактални и нефрактални. Бројот на фрактални квазигрупи од ред 4 е 192, а бројот на нефрактални квазигрупи е 384. Фракталните квазигрупи не се погодни за конструкција на криптографски примитиви, бидејќи тие даваат регуларни (правилни) структури.

Со цел да се зголеми бројот на квазигрупи погодни за примена во криптографија, во ова поглавје дадени се нови класификации на квазигрупите од ред 4, и тоа:

- класификација според бројот на различни парастрофи;
- класификација според парастрофна фракталност.

7.3.1 Класификација според бројот на различни парастрофи

Во ова поглавје разгледани се сите 576 квазигрупи од ред 4 и за секоја квазигрупа определено е множеството од сите нејзини парастрофи. Бројот на

елементи во секое од овие множества е најмногу 6, т.е., за некои квазигрупи не се сите парастрофи различни.

Определувајќи го бројот на различни парастрофи за секоја квазигрупа, множеството од сите квазигрупи од ред 4 го поделивме на 4 класи. Бројот на квазигрупи во секоја од овие класи е даден во Табела 7.2.

Бр. на парастрофи	Бр. на квазигрупи
1	16
2	2
3	240
6	318
Вкупно	576

Табела 7.2: Број на елементи во класите според број на различни парастрофи

Во Табела 7.3 дадена е поткласификација на претходната. Имено, во оваа поткласификација во зависност од бројот на различни парастрофи класифицирани се посебно фракталните и нефракталните квазигрупи од ред 4.

Бр. на парастрофи	Бр. на фрактални квазигрупи	Бр. на нефрактални квазигрупи
1	16	0
2	2	0
3	96	144
6	78	240
Вкупно	192	384

Табела 7.3: Број на елементи во поткласите според број на различни парастрофи

Од Табела 7.3, може да се види дека класата на фрактални квазигрупи од ред 4 е поделена во 4 поткласи, а класата на нефрактални квазигрупи е поделена само во 2 поткласи. Со споредба на Табела 7.2 и Табела 7.3 може да се заклучи дека сите квазигрупи со една или две различни парастрофи се

фрактални, додека квазигрупите со 3 или 6 парастрофи може да бидат фрактални или нефрактални.

Следното својство е докажано со детална проверка на сите квазигрупи.

Својство 7.1. *Парастрофите на фракталните квазигрупи од ред 4 се исто така фрактални.*

Оттука, сите нефрактални квазигрупи од ред 4 имаат нефрактални парастрофи.

7.3.2 Класификација според парастрофна фракталност

Со користење на истиот графички патерн, како и во [17], направивме слична класификација на квазигрупите од ред 4 со користење на новата PE- трансформација наместо E-трансформацијата. Новата $PE^{(n)}$ трансформација ја применивме на низата $123412341234...$ и ја разгледувавме фракталната структура на добиената слика. Во зависност од таа структура, дефинираме нови видови на фрактални квазигрупи.

Дефиниција 7.1. *Квазигрупите со фрактална структура добиена по примената на PE-трансформацијата се нарекуваат* ***парастрофно фрактални квазигрупи****. Во спротивно, квазигрупата се нарекува* ***парастрофно нефрактална квазигрупа****.*

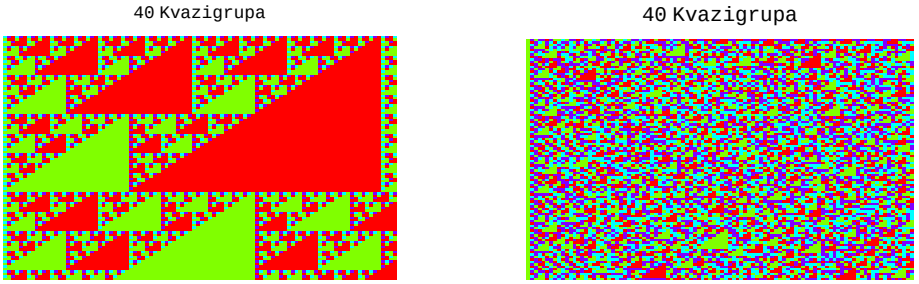
Направени се експерименти со графички патерн за сите 576 квазигрупи од ред 4 и определено е дека 88 се парастрофно фрактални и 488 се парастрофно нефрактални. Од експериментите заклучивме дека сите парастрофно фрактални квазигрупи се во класата на фрактални квазигрупи, но не сите фрактални квазигрупи се парастрофно фрактални. Ова значи дека класата од сите 192 фрактални квазигрупи е поделена во 2 потткласи:

1) парастрофно фрактални (88 квазигрупи) и

2) фрактални, но парастрофно нефрактални (104 квазигрупи), наречени ***фрактални парастрофно-нефрактални квазигрупи***.

На Слика 7.2, претставени се графичките патерни добиени со квазигрупата со лексикографски број 40 која е фрактална (а), но не е парастрофно нефрактална (б).

Од друга страна, класата од сите 384 нефрактални квазигрупи целосно е содржана во класата на парастрофно нефракталните квазигрупи.



(а) (б)

Слика 7.2: Фрактална, но парастрофно нефрактална квазигрупа

Во согласност со горе-наведеното, направена е следната нова класификација на квазигрупите од ред 4:

1) класа на парастрофно-фрактални квазигрупи (88 квазигрупи);

2) класа на фрактални парастрофно-нефрактални квазигрупи (104 квазигрупи);

3) класа на нефрактални квазигрупи (384 квазигрупи).

Со оваа класификација зголемен е бројот на квазигрупи погодни за примена во криптографија. Имено, за криптографски цели може да се користат не само квазигрупите од класа 3, туку и квазигрупите од класа 2.

Понатаму, во зависност од бројот на различни парастрофи, определен е бројот на елементи во поткласите од парастрофно-фрактални и фрактални парастрофно-нефрактални квазигрупи. Добиените резултатите дадени се во Табела 7.4.

Со споредба на Табела 7.3 и Табела 7.4 може да се заклучи дека сите фрактални квазигрупи со една парастрофа се парастрофно фрактални, а сите фрактални квазигрупи со 2 или 6 различни парастрофи се фрактални парастрофно - нефрактални квазигрупи. Само поткласата од фрактални квазигрупи со 3 различни парастрофи содржи и парастрофно фрактални и фрактални парастрофно - нефрактални квазигрупи.

Бр. на парастрофи	Бр. на парастрофно-фрактални	Бр. на фрактални парастрофно-нефрактални
1	16	0
2	0	2
3	72	24
6	0	78
Вкупно	88	104

Табела 7.4: Број на елементи во поткласите од фрактални квазигрупи според бројот на различни парастрофи

7.4 Алгебарски особини на парастрофно фракталните квазигрупи

Во [45], авторите даваат математички модел на фракталноста на квазигрупите со користење на идентитети. Со цел да најдеме сличен модел, но сега за парастрофната фракталност на квазигрупите, ги испитавме алгебарските својства на парастрофно фракталните квазигрупи од ред 4. За да најдеме соодветни идентитети со кои ќе се издвојат парастрофно-фракталните квазигрупи, разгледавме многу идентитети, а посебно идентитетите за симетрија. Суштински идентитети за нашиот модел се следните:

- комутативност ($x * y = y * x$),
- коса симетрија ($(x * y) * (y * x) = const$)
- лева лупа ($e * x = x$),
- десна лупа ($x * e = x$),
- десна симетрија ($(x * y) * y = x$),
- лева симетрија ($y * (y * x) = x$),
- тотална симетрија (комутативност и лева симетрија).

Користејќи детална проверка најдовме дека секоја парастрофно - фрактална квазигрупа од ред 4 припаѓа во множеството I од квазигрупи кои го

задоволуваат идентитетот $x * (x * (x * (x * y))) = y$ и во едно од следните множества од квазигрупи:

- Лупи (Loops - L)
- Тотално симетрични квазигрупи (Totally symmetric quasigroups - TS)
- Леви Лупи (Left Loops - LL) и Десно симетрични квазигрупи (Right symmetric quasigroups - RS)
- Десни Лупи (Right Loops - RL) и Лево симетрични квазигрупи (Left symmetric quasigroups - LS)
- Леви Лупи (Left Loops - LL) и Коso симетрични квазигрупи (Skew symmetric quasigroups - SS)
- Десни Лупи (Right Loops - RL) и Коso симетрични квазигрупи (Skew symmetric quasigroups - SS)
- Комутативни квазигрупи (Commutative quasigroups - C) и Коso симетрични квазигрупи (Skew symmetric quasigroups - SS).

Со користење на оваа нотација, со директна проверка докажано е следното својство:

Својство 7.2. *Множеството од сите парастрофно-фрактални квазигрупи PFQ од ред 4 може да биде претставено на следниот начин:*

$$PFQ = I \cap [L \cup TS \cup (LL \cap RS) \cup (RL \cap LS) \cup (SS \cap (LL \cup RL \cup C))].$$

На овој начин, добиен е математички модел за парастрофната фракталност и без користење на графички патерн може да се провери дали дадена квазигрупа е парастрофно-фрактална или не е.

7.5 Теоретски доказ за отпорност од статистички напади

За да се комплетира доказот за подобноста на парастрофната квазигрупна трансформација за примена во криптографија, потребно е да се докаже дека Теоремата 1.2 важи и за оваа трансформација. Имено, за примена на една трансформација во криптографија многу важно својство е рамномерната распределба на подстринговите во излезната порака. Ова својство гарантира

дека пораката криптирана со парастрофната квазигрупна трансформација е отпорна на статистички напади. Затоа, во ова поглавје ќе ја испитаме распределбата на потстринговите во излезната порака добиена со користење на PE- трансформацијата.

Нека азбуката A е дефинирана како погоре. Случајно избран елемент од множеството A^k може да биде разгледуван како случаен вектор $(X_1, X_2, \ldots, X_k)$, каде A е рангот на X_i, $i = 1, \ldots, k$. Овој вектор ќе го рагледуваме како влезна порака. Сега, трансформацијата $PE = PE_{l,d_1} : A^+ \to A^+$ може да биде дефинирана на следниот начин:

$$PE_{l,d_1}(X_1, \ldots, X_k) = (Y_1, \ldots, Y_k) \Leftrightarrow$$

$$\Leftrightarrow \begin{cases} Y_1 = f_{s_1}(l, X_1), \;\; Y_j = f_{s_1}(Y_{j-1}, X_j), \; j = 2, \ldots, d_1, \\ \\ Y_{q_i+j} = f_{s_{i+1}}(Y_{q_i+j-1}, X_{q_i+j}), i = 1, \ldots, r-1, \; j = 1, \ldots, d_{i+1} \end{cases} \tag{7.3}$$

Нека $(p_1, p_2, ..., p_a)$ е законот на распределба на буквите $1, ..., a$ во влезната порака. Тоа значи дека $p_i > 0$ за секое $i = 1, 2, ..., a$ и $\sum_{i=1}^{a} p_i = 1$.

Прво ќе докажеме дека после примената на трансформацијата $PE^{(1)}$ на влезна порака α, буквите во трансформираната порака имаат рамномерна распределба.

Теорема 7.1. *Буквата Y_t има рамномерна распределба на множеството $A = \{1, \ldots, a\}$, т.е., $Y_t \sim U(\{1, \ldots, a\})$ за секое t $(t = 1, 2, \ldots, k)$.*

Доказ. Во овој доказ користени се истите ознаки како и во конструкцијата на парастрофната квазигрупна трансформација дадена во претходното поглавје.

Прво, да забележиме дека лидерот l може да го разгледуваме како случајна променлива која има рамномерна распределба на множеството A, бидејќи тој е случајно избран елемент од множеството A. Затоа, $l \sim U(\{1, \ldots, a\})$, т.е.,

$$P\{l = i\} = \frac{1}{a}, \qquad \text{за секое } i \in A.$$

Исто така, лидерот l е случајна променлива која е независна од буквите X_i $(i = 1, \ldots, k)$ во влезната порака.

Нека $t = 1$. Со користење на равенството (7.3) и теоремата за тотална веројатност, за распределбата на Y_1, добиваме

$$\begin{aligned} P\{Y_1 = j\} &= P\{f_{s_1}(l, X_1) = j\} \\ &= \sum_{i=1}^{a} P\{l = i\} P\{f_{s_1}(l, X_1) = j | l = i\} \\ &= \sum_{i=1}^{a} \frac{1}{a} P\{f_{s_1}(l, X_1) = j | l = i\} \\ &= \sum_{i=1}^{a} \frac{1}{a} P\{f_{s_1}(i, X_1) = j\} \\ &= \frac{1}{a} \sum_{i=1}^{a} P\{X_1 = f'_{s_1}(i, j)\} \end{aligned}$$

Тука, f'_{s_1} е инверзната квазигрупна трансформација на f_{s_1}, т.е. ако $f_{s_1}(u, x) = v$, тогаш $f'_{s_1}(u, v) = x$. Да забележиме дека ако i се менува над сите вредности во A тогаш за дадено j, со изразот $X_1 = f'_{s_1}(i, j)$ ќе се добијат сите вредности од А. Оттука,

$$P\{Y_1 = j\} = \frac{1}{a} \sum_{i=1}^{a} P\{X_1 = f'_{s_1}(i, j)\} = \frac{1}{a} \sum_{i=1}^{a} p_i = \frac{1}{a},$$

т.е., $Y_1 \sim U(\{1, \ldots, a\})$.

Понатаму доказот ќе го изведеме со користење на индукција. Нека $Y_r \sim U(\{1, 2, \ldots, a\})$. Слично како и претходно, користејќи дека $f_{s_{r+1}}$ е парастрофната трансформација користена во $(r+1)^{th}$ чекор, распределбата на Y_{r+1} ја определуваме на следниот начин.

$$\begin{aligned} P\{Y_{r+1} = j\} &= P\{f_{s_{r+1}}(Y_r, X_{r+1}) = j\} \\ &= \sum_{i=1}^{a} P\{Y_r = i\} P\{f_{s_{r+1}}(Y_r, X_{r+1}) = j | Y_r = i\} \\ &= \sum_{i=1}^{a} \frac{1}{a} P\{f_{s_{r+1}}(i, X_{r+1}) = j | Y_r = i\} \end{aligned}$$

Во согласност со дефиницијата на парастрофната трансформација дадена со (7.3), може да заклучиме дека случајните променливи X_{r+1} и Y_r се независни.

Користејќи го ова во претходното равенство, добиваме

$$\begin{aligned} P\{Y_{r+1} = j\} &= \sum_{i=1}^{a} \frac{1}{a} P\{f_{s_{r+1}}(i, X_{r+1}) = j\} \\ &= \frac{1}{a} \sum_{i=1}^{a} P\{X_{r+1} = f'_{s_{r+1}}(i, j)\} \\ &= \frac{1}{a}. \end{aligned}$$

Како и претходно, $f'_{s_{r+1}}$ е инверзната квазигрупна трансформација на $f_{s_{r+1}}$. Во последното равенство, го користиме фактот дека $X_{r+1} = f'_{s_{r+1}}(i, j)$ ќе ги прими сите вредности од A кога j е фиксно, а i се менува над сите вредности од A, т.е.

$$\sum_{i=1}^{a} P\{X_{r+1} = f'_{s_{r+1}}(i, j)\} = \sum_{i=1}^{a} p_i = 1.$$

На овој начин, докажавме дека Y_t има рамномерна распределба на множеството A, за секое $t \geqslant 1$.

Од Теорема 7.1 може да го изведеме следниот заклучок. Ако $M \in A^k$ и $C = PE_{l,d_1}(M)$, тогаш буквите во пораката C имаат рамномерна распределба, т.е., веројатноста буквата i да се појави на било кое место во стрингот C е $\frac{1}{a}$, за секое $i \in A$.

Теорема 7.2. *Нека $M \in A^+$ е произволен стринг и $C = PE^{(n)}(M)$. Тогаш m-торките од букви во C имаат рамномерна распределба за $m \leqslant n$.*

Доказ. Нека $(Y_1^{(n)}, Y_2^{(n)}, \ldots, Y_k^{(n)}) = PE^{(n)}(X_1, X_2, \ldots, X_k)$. Теоремата ќе ја докажеме со користење на индукција. Да претпоставиме дека тврдењето е точно за $n = r$, т.е., $(Y_{t+1}^{(r)}, Y_{t+2}^{(r)} \ldots Y_{t+l}^{(r)}) \sim U(\{1, 2, \ldots, a\}^l)$ за секое $1 \leqslant l \leqslant r$ и секое $t \geqslant 0$. Сега, нека $n = r + 1$. Ќе ја разгледаме распределбата на $(Y_{t+1}^{(r+1)}, Y_{t+2}^{(r+1)} \ldots Y_{t+l}^{(r+1)})$ за секое $1 \leqslant l \leqslant r + 1$ и произволно t.

$$\begin{aligned} &P\{Y_{t+1}^{(r+1)} = y_{t+1}^{(r+1)}, Y_{t+2}^{(r+1)} = y_{t+2}^{(r+1)}, \ldots, Y_{t+l}^{(r+1)} = y_{t+l}^{(r+1)}\} \\ &\quad = P\{Y_{t+1}^{(r+1)} = y_{t+1}^{(r+1)}, f_{s_{t+2}}(Y_{t+1}^{(r+1)}, Y_{t+2}^{(r)}) = y_{t+2}^{(r+1)}, \ldots \\ &\qquad\qquad \ldots, f_{s_{t+l}}(Y_{t+l-1}^{(r+1)}, Y_{t+l}^{(r)}) = y_{t+l}^{(r+1)}\}, \end{aligned}$$

каде f_{s_j} е парастрофната операција користена во чекорот j и f'_{s_j} е нејзината инверзна трансформација, $j = t+2, \ldots, t+l$. Понатаму,

$$\begin{aligned}
&P\{Y_{t+1}^{(r+1)} = y_{t+1}^{(r+1)}, Y_{t+2}^{(r+1)} = y_{t+2}^{(r+1)}, \ldots, Y_{t+l}^{(r+1)} = y_{t+l}^{(r+1)}\} \\
&\quad = P\{Y_{t+1}^{(r+1)} = y_{t+1}^{(r+1)}, f_{s_{t+2}}(y_{t+1}^{(r+1)}, Y_{t+2}^{(r)}) = y_{t+2}^{(r+1)}, \ldots \\
&\qquad\qquad \ldots, f_{s_{t+l}}(y_{t+l-1}^{(r+1)}, Y_{t+l}^{(r)}) = y_{t+l}^{(r+1)}\} \\
&\quad = P\{Y_{t+1}^{(r+1)} = y_{t+1}^{(r+1)}, Y_{t+2}^{(r)} = f'_{s_{t+2}}(y_{t+1}^{(r+1)}, y_{t+2}^{(r+1)}), \ldots \\
&\qquad\qquad \ldots, Y_{t+l}^{(r)} = f'_{s_{t+l}}(y_{t+l-1}^{(r+1)}, y_{t+l}^{(r+1)})\} \\
&\quad = P\{Y_{t+1}^{(r+1)} = y_{t+1}^{(r+1)}\} P\{Y_{t+2}^{(r)} = f'_{s_{t+2}}(y_{t+1}^{(r+1)}, y_{t+2}^{(r+1)}), \ldots \\
&\qquad\qquad \ldots, Y_{t+l}^{(r)} = f'_{s_{t+l}}(y_{t+l-1}^{(r+1)}, y_{t+l}^{(r+1)})\}.
\end{aligned}$$

Последното равенство се добива со користење на фактот дека $Y_{t+1}^{(r+1)}$ е независна од векторот $(Y_{t+2}^{(r)}, \ldots, Y_{t+l}^{(r)})$, бидејќи $Y_{t+2}^{(r)}, \ldots, Y_{t+l}^{(r)}$ не се користат за добивање на $Y_{t+1}^{(r+1)}$.

Со користење на индуктивната претпоставка $Y_{t+1}^{(r+1)} \sim U(\{1, 2, \ldots, a\})$, $(Y_{t+2}^{(r)}, \ldots, Y_{t+l}^{(r)}) \sim U(\{1, 2, \ldots, a\}^{l-1})$ од претходниот израз добиваме дека

$$P\{Y_{t+1}^{(r+1)} = y_{t+1}^{(r+1)}, Y_{t+2}^{(r+1)} = y_{t+2}^{(r+1)}, \ldots, Y_{t+l}^{(r+1)} = y_{t+l}^{(r+1)}\} = \frac{1}{a} \cdot \frac{1}{a^{l-1}} = \frac{1}{a^l}.$$

Со ова докажавме дека $(Y_{t+1}^{(n)}, Y_{t+2}^{(n)} \ldots Y_{t+l}^{(n)}) \sim U(\{1, 2, \ldots, a\}^l)$ за секое $l \leqslant n$ и секое $t \geqslant 0$.

Со Теорема 7.2 покажавме дека пораката криптирана со парастрофната квазигрупна трансформација е отпорна на статистички напади.

7.6 Експериментални резултати

Направивме многу експерименти со цел да ги илустрираме нашите теоретски резултати. Во ова поглавје даден е еден пример. Случајно е избрана порака M со 1,000,000 букви од азбуката $A = \{1, 2, 3, 4\}$ со распределба на буквите дадена во Табела 7.5 .

1	2	3	4
0.70	0.15	0.10	0.05

Табела 7.5: Распределба на буквите во влезната порака

Користена е квазигрупата (7.4) и нејзините парастрофи.

$*$	1	2	3	4
1	3	2	1	4
2	1	4	2	3
3	4	1	3	2
4	2	3	4	1

(7.4)

Со примена на трансформацијата $PE^{(3)}$ на пораката M, се добива криптираната порака $C = PE^{(3)}(M)$. Во секоја PE-трансформација, должината на првиот блок е $d_1 = 3$, а почетниот лидер е $l_1 = 1$.

Распределбата на буквите во излезната порака C е дадена во Табела 7.6.

1	2	3	4
0.2505	0.2498	0.2502	0.2494

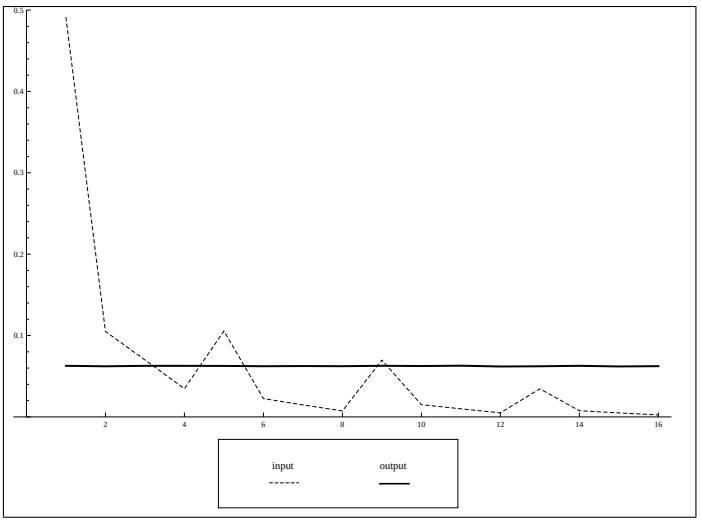
Табела 7.6: Распределба на буквите во излезната порака

Од веројатностите во Табела 7.6, јасно е дека распределбата на буквите во излезната порака C е рамномерна.

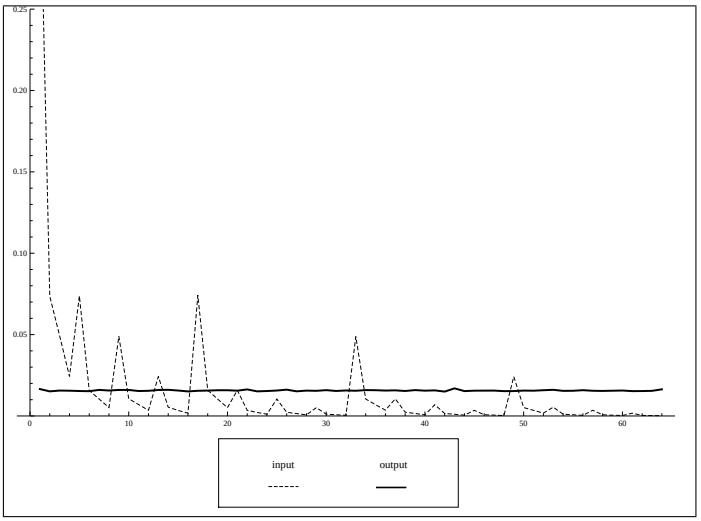
Распределбите на паровите, тројките и четворките од букви во C претставени се на Слика 7.3, Слика 7.4 и Слика 7.5. На Слика 7.3, паровите се претставени на x-оската во лексикографски редослед ($'11' \to 1, '12' \to 2, \ldots,$ $'44' \to 16$). На сличен начин, на Слика 7.4 и Слика 7.5 прикажани се тројките и четворките од букви.

Од Слика 7.3 и Слика 7.4 може да се заклучи дека после три примени на PE-трансформацијата, паровите и тројките од букви во излезната порака имаат рамномерна распределба, како што е и докажано во Теорема 7.2. Исто така, на Слика 7.5 може да се види дека распределбата на четворките во C не

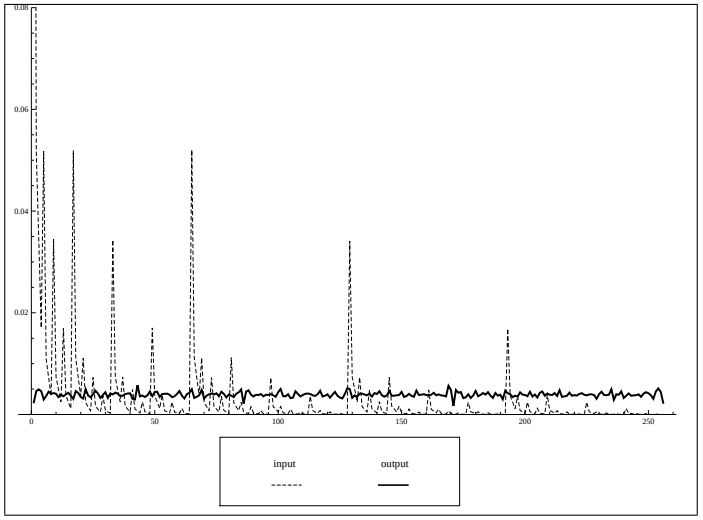
е рамномерна, но сепак оваа распределба е поблиска до рамномерна распределба споредено со распределбата на четворките во влезната порака.



Слика 7.3: Распределба на паровите од букви во влезната и излезната порака



Слика 7.4: Распределба на тројките во влезната и излезната порака

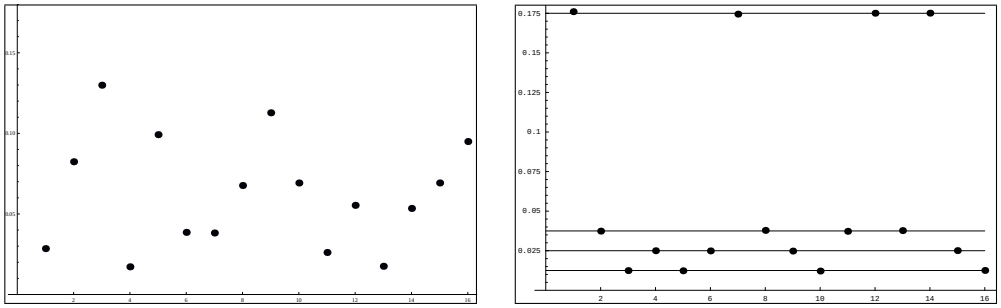


Слика 7.5: Распределба на четворките во влезната и излезната порака

Во Теорема 1.4 (која е докажана во [2]), Бакева и Димитрова докажуваат дека веројатностите на $(n+1)$-торките од букви во $\beta = E^{(n)}(\alpha)$ се поделени во a класи каде $a = |A|$, ако $(p_1, p_2, \ldots, p_a)$ е законот на распределба на буквите во влезниот стринг и притоа $p_1, p_2, \ldots, p_a$ се различни веројатности, т.е., $p_i \neq p_j$ за $i \neq j$. Притоа, секоја класа содржи a^n елементи со исти веројатности и веројатноста на секоја $(n+1)$-торка во i-тата класа е $\frac{1}{a^n}p_i$, за $i = 1, 2, \ldots, a$. Ако $p_{i_1} = p_{i_2} = \cdots = p_{i_\nu}$ за некои $1 \leqslant i_1 < \cdots < i_\nu \leqslant a$, тогаш класите со веројатности $\frac{1}{a^n}p_{i_1} = \frac{1}{a^n}p_{i_2} = \cdots = \frac{1}{a^n}p_{i_\nu}$ ќе бидат споени во една класа со νa^n елементи. Користејќи ги овие резултати, авторите предлагаат и алгоритам за криптоанализа.

Експериментално проверивме дали оваа теорема важи кога се применува PE-трансформацијата. Распределбата на паровите после една примена на PE-трансформацијата е прикажана на Слика 7.6 а). На Слика 7.6 б), претставена е распределбата на паровите после една примена на E - трансформацијата. Може да се види дека на Слика 7.6 б) веројатностите на паровите се поделени во 4 класи како што е и покажано во [2]. Но на Слика 7.6 а) не може да се одвојат некакви класи на веројатностите. Ова значи дека алгоритмот за криптоанализа предложен во [2] не може да се примени кога влезната порака е криптирана со користење на PE-трансформацијата. Оттука, може да заклучиме дека пораката криптирана со PE-трансформација е поотпорна

на статистички напади.



а) б)

Слика 7.6: Распределбата на паровите од букви во излезните пораки добиени со PE- и E-трансформација

Да забележиме дека за релевантни статистички анализи, мора да имаме доволно долга влезна порака. Имено, во експериментите, веројатностите на n-торките се пресметуваат како релативни фрекфенции. Но, релативната фрекфенција на еден настан тежи кон веројатноста на настанот само ако имаме доволно голем примерок. Затоа, релевантни статистички анализи не може да бидат направени за кратки пораки. Оттука, невозможно е да се направи статистички напад на недоволно долга влезна порака. Да забележиме дека ако еден напаѓач фати и конкатенира голем број на кратки пораки криптирани со иста $PE^{(n)}$-трансформација, тој ќе добие долга порака и може да примени статистички напад. Но, нападот ќе биде невозможен ако почесто се менуваат квазигрупите користени во криптирањето со $PE^{(n)}$- трансформацијата.

Во [47], авторите заклучуваат дека E-трансформацијата може да се применува во криптографија како функција за криптирање, бидејќи бројот на квазигрупи е огромен (постојат повеќе од 10^{58000} квазигрупи кога $|A| = 256$) и затоа нападот со груба сила (brute force) е неприфатлив.

Ако PE-трансформацијата се користи во алгоритам за криптирање, тогаш тајниот клуч ќе биде тројката $(*, l, d_1)$. Во тој случај, нападот "со груба сила" е исто така невозможен, бидејќи освен квазигрупната операција $*$ и лидерот l, клучот ја содржи и должината на првиот блок d_1 која влијае и на динамиката со која се менуваат парастрофите.

Заклучок

Во оваа глава е дефинирана нова квазигрупна трансформација (PE-трансформација) и направени се две нови класификации на квазигрупите од ред 4 корисни во криптографија. Исто така, даден е математички модел за парастрофната фракталност на квазигрупите и се докажани Теорема 7.1 и Теорема 7.2 за рамномерната распределба на подстринговите во пораката добиена со новата квазигрупна трансформација.

Со сумирање на добиените резултати може да заклучиме дека:

– парастрофно фракталните квазигрупи не треба да се користат за дизајн на криптографски примитиви, бидејќи тие имаат фрактални структура и својства на симетрија;

– дефиниран е математички модел за одделување на парастрофно фракталните квазигрупи;

– парастрофната трансформација е попогодна за дизајн на криптографски примитиви, бидејќи таа го зголемува бројот на квазигрупи корисни во криптографија.

На крај, од сите изнесени резултати во оваа глава може да се заклучи дека PE- трансформацијата е подобра функција за криптирање од E - трансформацијата.

Заклучок

Квазигрупите и квазигрупните трансформации се многу погодни алгебарски структури за конструкција на криптографски примитиви, кодови за откривање на грешки и кодови кои поправаат грешки. Причини за тоа се: структурата на квазигрупите, нивниот голем број, својствата на квазигрупните трансформации и друго. Досегашните истражувања за нивната примена како во областа на кодирањето, така и во областа на криптографијата дадоа голем придонес во новите достигнувања во овие области. Оваа докторска дисертација е уште еден прилог кон примената на квазигрупите и квазигрупните трансформации во овие две области.

Најголем дел од дисертацијата е посветен на случајните кодови базирани на квазигрупи предложени од Данило Глигороски, Смиле Марковски и Љупчо Коцарев. Овие кодови претставуваат крипто-кодови кои поправаат грешки и притоа користат алгоритми за криптирање/декриптирање во самиот процес на кодирање и декодирање. Затоа, тие овозможуваат не само корекција на одреден број на грешки во пратените податоци, туку исто така обезбедуваат и безбедност на информациите. Ако информацијата е кодирана со користење на овие кодови, тогаш примателот може да ја декодира и да ја добие оригиналната информација само ако тој/таа знае точно кои параметри биле користени во процесот на кодирање, дури и ако комуникацискиот канал е без пречки. Заради се поголемата потреба од пренос на податоците кој во исто време ќе биде и ефикасен и безбеден, случајните кодови базирани на квазигрупи се предизвик за нови истражувања.

Направени се многу експерименти со цел да се испитаат перформансите на овие кодови и истражувано е како изборот на параметрите во нивниот дизајн влијае на нивните перформанси. Од добиените резултати заклучивме дека перфомансите на овие кодови многу зависат од изборот на квазигрупата и другите параметри во нивниот дизајн. Од експерименталната споредба на перформансите на овие кодови со кодови, познати по својата практична примена за складирање и пренос на податоци (како што се кодовите на Рид-

Милер и Рид-Соломон), произлезе заклучокот дека и кодовите базирани на квазигрупи можат да најдат широка практична примена особено заради нивните криптографски својства. Но, отворено остана прашањето за забрзување на процесот на декодирање на овие случајни кодови, што се покажа како голема слабост при изведувањето на експериментите.

Со цел да се реши најголемиот проблем на случајните кодовите базирани на квазигрупи, т.е., да се забрза процесот на декодирање, предложени се два нови алгоритми за кодирање/декодирање. Со првиот предложен алгоритам, наречен алгоритам-за-декодирање-со-пресеци (Cut-Decoding или АДП алгоритам), постигнато е големо подобрување на брзината на декодирање. Притоа не се намалени добрите перформанси на овие кодови во однос на веројатностите за пакет-грешка и бит-грешка. За код со рата 1/4, декодирањето на пораките со користење на АДП алгоритамот е 4.5 пати побрзо отколку со користење на стандардниот алгоритам за декодирање.

Добиеното забрзување на процесот на декодирање со користење на новиот АДП алгоритам, отвори ново прашање: Дали со нова модификација на овој алгоритам може да се постигне поголемо намалување на кандидатите за декодирање во секоја од итерациите и со тоа да се постигне поголемо забрзување на декодирањето? Оттука, произлезе идејата за нова модификација на АДП алгоритамот, која е наречена алгоритам-за-декодирање-со-4пресеци (4-Sets-Cut -Decoding или АД4П алгоритам). Во првичните експерименти со овој алгоритам добивме поголемо забрзување на процесот на декодирање, но полоши резултати во однос на веројатностите за пакет-грешка и бит-грешка. Затоа, направени се дополнителни три верзии на овој алгоритам (АД4П#2, АД4П#3 и АД4П#4 алгоритми) со цел да се добие подобрување на перформансите и во однос на веројатностите за точно декодирање, а притоа да се задржи подобрувањето на брзината. Со анализа на резултатите добиени од голем број на експерименти направени со различните алгоритми за декодирање, заклучено е дека за код со рата 1/8 со новите алгоритми (АДП и АД4П алгоритмите) се добива големо подобрување на брзината на декодирање и многу подобри резултати за веројатностите за пакет-грешка и бит-грешка. Имено, за овој код АДП алгоритамот, за одредени вредности на веројатност p за бит-грешка во бинарен симетричен канал, е 5.2 пати побрз од стандардниот. Додека со АД4П алгоритмите декодирањето е 6.3 пати побрзо отколку со стандардниот алгоритам, а за некои вредности на p се добиваат и 24 пати помали веројатности за пакет-грешка.

За дополнително подобрување на веројатностите за успешно декодирање дефинирани се неколку нови методи за намалување на неуспешните декоди-

рања со двата типа на грешки (*грешка-празно-множество* и *грешка-повеќе-кандидати*). Со примена на комбинација од овие предложените методи во новите алгоритми за кодирање/декодирање елиминирани се дел од неуспешните декодирања и добиени се многу подобри резултати за веројатностите за пакет-грешка и бит-грешка. Притоа, брзината на декодирањето е скоро непроменета.

Како заокружување на испитувањето на перформансите на овие кодови, направени се експерименти со различни должини на пораките, при иста рата на кодот, со цел да се прикаже како промената на должината на пораките влијанието на перформансите. Во дизајнот и претходните истражувања со кодовите базирани на квазигрупи користена е само азбуката од нибли и квазигрупи од ред 16. Во оваа дисертација испитани се и перформансите на кодовите кои во процесите на кодирање и декодирање користат квазигрупи од ред 4 или ред 256 (во стандардниот и АДП алгоритамот). Исто така, направени се експерименти во кои овие кодови се користат за кодирање/декодирање на слики пренесени низ бинарен симетричен канал. Притоа, користени се неколку различни (во боја и црно-бели) слики и споредени се перформансите на стандарниот алгоритам за кодирање/декодирање, АДП алгоритамот и кодовите на Рид-Соломон за кодови со рата 1/4. Од добиените резултати заклучивме дека за оваа рата со користење на новиот АДП алгоритам се добиваат појасни декодирани слики отколку со стандарниот алгоритам и Рид-Соломон кодовите. Исто така, направена е споредба на перформансите на случајните кодови базирани на квазигрупи со АД4П#3 алгоритмот и Рид-Соломон код со рата 1/8 при декодирање на слики пренесени низ бинарен симетричен канал. Од добиените резултати заклучивме дека и за оваа рата случајниот код со новиот АД4П#3 алгоритам дава појасни слики (за сите разгледани веројатности за бит-грешка во бинарен симетричен канал).

Како отворено прашање за понатомошни истражувања останува испитувањето на перформансите на случајните кодови базирани на квазигрупи за пораки пренесувани и низ други канали со пречки, како на пример за пренос низ Гаусов канал. Од сликите добиени по декодирањето со овие кодови произлегоа некои идеи за дефинирање на филтри за прочистување на добиените слики. Исто така, останува отворено прашањето и за примена на други квазигрупни трансформации или други алгоритми за криптирање/декриптирање во алгоритмите за кодирање и декодирање на овие криптокодови базирани на квазигрупи.

Освен експерименталните резултати за новите алгоритми кои се предложени во оваа теза, дадени се и некои теориски резултати со кои се пот-

врдува подобрувањето на перформансите на случајните кодови базирани на квазигрупи. Изведена е формула за горната граница за веројатноста за пакет-грешка кај новите алгоритми. Притоа, покажано е дека оваа горна граница е еднаква на теориската веројатност за пакет-грешка за стандардниот алгоритам. Оваа теориски изведена горна граница е и експериментално потврдена. Исто така, определени се приближни формули за кардиналниот број на множествата со кандидати за декодирање после нивното редуцирање предложено во новите алгоритми. Со тоа е покажано дека со новите алгоритми се добива значително намалување на бројот на елементи во множествата со кандидати за декодирање, од што најмногу зависи брзината на декодирање. Овие теориски резултати може да се искоритат и како математички метод за избор на добри параметри во дизајнот на кодовите.

Како прилог кон примената на квазигрупите и во областа на криптографијата, во оваа дисертација дефинирана е нова квазигрупна стринг трансформација наречена парастрофна квазигрупна трансформација (или PE- трансформација). Со користење на оваа трансформација направена е нова класификација на квазигрупите во зависност од нивната парастрофна фракталност. Направени се две нови класификации на квазигрупите од ред 4: 1) класификација според бројот на различни парастрофи; и 2) класификација според парастрофна фракталност. На овој начин, зголемен е бројот на квазигрупи од ред 4 кои се погодни за дизајн на криптографски примитиви. Исто така, испитани се алгебарските својства на парастрофно фракталните квазигрупи и предложен е математички модел за парастрофната фракталност. Дадени се неколку својства кои се корисни за примена на PE- трансформацијата во криптографија. Даден е и теоретски доказ дека после n примени на PE- трансформацијата на произволна порака, распределбата на m-торките од букви ($m = 1, \ldots, n$) е рамномерна. Класи на веројатностите во распределбата на $(n+1)$-торките не се издвојуваат како во случај на E-трансформацијата. Но, во овој случај можеби постои некоја зависност од повисок ред, што е отворено прашање на кое ќе се бара одговор во иднина. Ова значи дека ако PE- трансформацијата се користи како функција за криптирање, тогаш добиените шифрирани пораки се отпорни на статистички напади кога PE- трансформацијата е применета доволен број пати. Од сите изнесени резултати може да се заклучи дека PE- трансформацијата е подобра функција за криптирање од претходно дефинираната E-трансформацијата. Затоа, понатаму може да се размислува за дизајн на шифрувач во кој за криптирање на пораките ќе се користи новата PE- трансформација.

На крајот може да се заклучи дека анализите од направените истражувања

дадени во оваа дисертација даваат нови и оригинални резултати во насока на примената на квазигрупите како во областа на кодирањето, така и во областа на криптографијата. Резултатите добиени од овие истражувања се компатибилни со најсовремените истражувања во оваа област и даваат свој научен придонес во решавањето на проблемите поврзани со примената на квазигрупите во теоријата на кодирање и во криптографијата, како едни од водечките трендови во информатиката. Темата е актуелна и е во склад со истражувањата кои интензивно се одвиваат, како во решавањето на многуте отворени теориски предизвици, така и во дизајнирањето на практични апликации за точен и безбеден пренос. Исто така, добиените резултати отворија многу прашања за понатамошни истражувања, чие решавање би имало практична вредност во складирањето и коректниот пренос на податоците, како и во зголемувањето на нивото на заштита на истите.

Литература

[1] Bakeva V.: *Probabilistic model in cryptography and coding theory*, Pliska Studia Mathematica Bulgarica, Vol.16, (2004), pp.13-22.

[2] Bakeva V., Dimitrova V.: *Some Probabilistic Properties of Quasigroup Processed Strings useful in Cryptanalysis*, Gusev, M., Mitrevski, P. (eds.) ICT-Innovations 2010, Springer (2010), pp. 61-70.

[3] Bakeva V., Dimitrova V., Popovska-Mitrovikj A.: *Parastrophic Quasigroup String Processing*, Proc. of the 8^{th} Conference on Informatics and Information Technology with International Participants, Macedonia (2011) pp. 19-21.

[4] Bakeva V., Ilievska, N.: *A probabilistic model of error-detecting codes based on quasigroups*, Quasigroups and Related Systems, Vol. 17 (2009), pp. 135-148.

[5] Bakeva V., Popovska-Mitrovikj A., Dimitrova V.: *Resistance to statistical attacks of Parastrophic Quasigroup Transformation*, submitted to Serdica Journal of Computing

[6] Belousov V. D.: n*-arnie Kvazigruppi* (*n-ary Quasigroups*), Stiinca, Kisiniev, 1972.

[7] Bossert M.: *Channel Coding for Telecommunications*, John Wiley and Sons, Ltd, 1999.

[8] Bowman J. C.: *Coding Theory*, University of Alberta, Edmonton, Canada, 2003.

[9] Cooke B.: *Reed-Muller Error Correcting Codes*, MIT Undergraduate Journal of Mathematics, pp. 21-26.

[10] Cox D.R., Miller H.D.: *The Theory of Stochastic Processes*, Chapman and Hall, 1994.

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

[12] Denes J., Keedwell A. D: *Latin squares: New developments in the theory and applications*, Elsevier science publishers, 1991.

[13] Denes J., Keedwell A. D: *Some applications of non-associative algebraic systems in cryptology*, Pure Mathematics and Applications 12(2), 2001, 147-195.

[14] 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.

[15] Dimitrova V., Markovski J.: *On Quasigroup Pseudo Random Sequence Generators*, Proceedings of the 1st Balkan Conference in Informatics, Thessaloniki, Greece, (2003) pp. 393–401.

[16] Dimitrova V., Markovski J.: *Implementation of pseudo random sequence generator using quasigroup processing*, Proc. of the VI National Conference ETAI 2003, 17-20 Sep. 2003, Ohrid, Macedonia, pp. I-86–I-90.

[17] 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.

[18] Dimitrova V., Markovski S.: *Construction of n-ary qusigroups*, III Congress of Mathematicians of Macedonia, Macedonia, 2005.

[19] Dimitrova V., Markovski S., Mileva A.: *Periodic quasigroups string transformations*, The Journal Quasigroups and Related Systems vol.17 No. 2, 2009, pp. 191-204.

[20] Gligoroski D.: *Stream cipher based on quasigroup string transformations in* $\mathbb{Z}_p^*$, Contributions, Sec. Math. Tech. Sci., MANU, 2004.

[21] 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.

[22] Gligoroski D., Knapskog S.: *Edon-$\mathcal{R}(256, 384, 512)$-an Efficient Implementation of Edon-$\mathcal{R}$ Family of Cryptographic Hash Functions*, Cryptology ePrint Archive, Report 2007/154.

[23] Gligoroski D., Knapskog S.: *Adding MAC Functionality to Edon80*, International Journal of Computer Science and Network Security 7(1), 2004, 194-204.

[24] Gligoroski D., Markovski S., Bakeva V.: *On infinite class of strongly collision resistant hash functions "EDON-F" with variable length of output*, Proceedings of the 1st MII 2003 Conference, Thessaloniki, April 14-16,2003,pp. 302-308.

[25] Gligoroski D., Markovski S., Knapskog S. J.: *The Stream Cipher Edon80*, New Stream Cipher Designs: The eSTREAM Finalists, 152-169, 2008, Springer-Verlag.

[26] Gligoroski D., Markovski S., Kocarev Lj.: *Edon-$\mathcal{R}$, an Infinite Family of Cryptographic Hash Functions*, The Second NIST Cryptographic Hash Workshop, UCSB, Santa Barbara, CA, 2006, pp. 275–285.

[27] 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.

[28] Gligoroski D., Markovski S., Kocarev Lj.: *A Synchronous Stream Cipher + Redundancy = Cryptcoding - Extended Abstract*, The 2007 Intern. Confer. on Security and Management SAM'07, June 2007, Las Vegas, pp. 25-28.

[29] Gligoroski D., Markovski S., Kocarev Lj.: *New Directions in Coding: From statistical Physics to Quasigroup String Transformations*, 2004 Inter. Symp. on Nonlinear Theory and Applc. (NOLAT2004), Fukuoka, Japan, Nov. 29- Dec. 3, (2004), pp. 545-548.

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

[31] Gligoroski D., Markovski S., Kocarev L., Gusev M.: *Edon80 - Hardware Synchronous stream cipher*, SKEW 2005 - Symmetric Key Encryption Workshop, May 2005, Aarhus Denmark.

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

[33] Hoffman D. G., Leonard D. A., Lindner C. C., Phelps K.T., Rodger C. A., Wall J. R.: *Coding Theory, The Essentials*, Auburn University, Auburn, Alabama, 1992.

[34] Huffman W. C., Pless V.: *Fundamentals of Error Correcting Codes*, Cambridge University Press, 2003.

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

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

[37] Lin S., Costello D. J.: *Error Control Coding: Fundamentals and Applications*, Prentice-Hall, Inc. Englewood Cliffs, New Jersey, 1983.

[38] MacKay D. J. C.: *Information Theory, Inference, and Learning Algorithms*, Cambridge University Press, 2003.

[39] Markovski J., Dimitrova V.: *Improving existing PRSG using QSP*, Fourth International for Informatics and Information Technology, 11-14 Dec. 2003, Bitola, Macedonia.

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

[41] Markovski S., Bakeva V.: *On a Stream Error Correcting Codes*, Proceedings of the 2nd CIIT, Bitola, Macedonia (2001), pp. 14-19.

[42] Markovski S., Bakeva V.: *On Error-Detecting Codes Based on Quasigroup Operations*, CIIT, Molika, (2003), pp. 400-405.

[43] Markovski S., Bakeva V.: *Error-Detecting Codes with Cyclically Defined Redundancy*, Зборник на трудови од III Конгрес на математичарите и информатичарите на Македонија, Струга, (2006), pp. 485-492.

[44] Markovski S., Bakeva V.:*Quasigroup String Processing: Part* 4, Contributions, Sec. Math. Tech.l Sci., MANU, XX 1-2, Skopje, Macedonia (2006-2007), pp.41-53.

[45] Markovski S., Dimitrova V., Samardziska S.: *Identities sieves for quasigroups*, Quasigroups and Related Systems, Vol. 18, No. 2, (2010) pp. 149-164.

[46] 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.

[47] 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.

[48] Markovski S., Gligoroski D., Bakeva V.: *Random walk tests for pseudo random number generators*, Mathematical Communications, Vol. 6, No.2, Osijek, Croatia (2001), pp. 135-143.

[49] Markovski S., Gligoroski D., Bakeva V.: *Quasigroup Hash Functions*, Proceedings of Workshop of Education of Informaticians and Industrial Mathematicians: New Challenges and Needs, Ohrid, Macedonia, 2001, pp. 123-131.

[50] 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.

[51] 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.

[52] 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.

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

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

[55] Markovski S., Mileva A.: *Generating huge quasigroups from small nonlinear bijections via extended Feistel function*, Quasigroups and Related Systems 17, 91-106, 2009.

[56] 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.

[57] Mathur C. N., Narayan K., Subbalakshmi i K.P: *High Diffusion Codes: A Class of Maximum Distance Separable Codes for Error Resilient Block Ciphers* 2nd IEEE International Workshop on Adaptive Wireless Networks (AWiN), Globecom, (2005).

[58] McEliece R.J.: *A Public-Key Cryptosystem Based on Algebraic Coding Theory*, DSN Progress Report, Jet Propulsion Laboratory, Calif., (1978), pp. 114–116.

[59] McKay B. D.: *Latin squares*, A Web Resourse $http : //cs.anu.edu.au /bdm/data/latin.html$.

[60] McKay B., Meynert A., Myrvold W: *Small Latin Squares, Quasigroups and Loops*, J. Combinatorial Designs 15, 2007, 98-119.

[61] Menezes A. J., P. C. Van Orchort P. C, Vanstone S. A.: *Handbook of Applied Cryptography*, CRC Press, 1997.

[62] Mileva A., Markovski S.: *Quasigroups String Transformations and Hash Function Design. A Case Study: The NaSHA Hash Function*, ICT Innovations conference 2009, Ohrid, pp 367-376.

[63] Moreira J. C., Farrell P. G.: *Essentials of Error-Control Coding*, John Wiley and Sons, Ltd, 2006.

[64] Nanjunda C., Haleem M., Chandramouli R.: *Robust Encryption for Secure Image Transmission over Wireless Channels*, IEEE International Conference on Communications (ICC 2005), 2005, Seoul, Korea, pp. 1287-1291.

[65] Pless V. S. and Huffman W. C. (editors): *Handbook of Coding Theory*, Elsevier Science B.V., Amsterdam, The Netherlands, 1998.

[66] Papoulis A.: *Probability, Random Variables and Stochastic Processes*, McGraw-Hill, Inc., New York, 1965.

[67] 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.

[68] 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.

[69] 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.

[70] 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 Particiipants, Faculty of Computer Science and Engineering, University ”Ss.Cyril and Methodius” (Macedonia), Bitola, Macedonia, April 2012, pp. 214-217.

[71] 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, pp. 178-181.

[72] Popovska-Mitrovikj A., Markovski S., Bakeva V.: *Error-correcting codes with cryptographic algorithms*, Proceedings of 21st Telecommunications Forum (TELFOR), 2013, Belgrade, Serbia, pp. 327-330, $http$: $//ieeexplore.ieee.org/xpl/articleDetails.jsp?tp = \&arnumber = 6716236\&queryText\%3DPopovska - Mitrovikj.$

[73] Popovska-Mitrovikj A., Markovski S., Bakeva V.: *4-Sets-Cut-Decoding algorithms for random codes based on quasigroups*, submitted to Advances in Mathematics of Communications (AMC)

[74] Popovska-Mitrovikj A., Mechkaroska D., Bakeva V.:, *Applying error-correcting codes based on quasigroups for image coding*, Proceedings of the 11th International Conference on Informatics and Information Technology, Faculty of Computer Science and Engineering, University "Ss.Cyril and Methodius" (Macedonia), Bitola, Macedonia, April 2014, (in print).

[75] Purser M.: *Introduction of Error-Correcting Codes*, Artech House, Boston, 1995.

[76] Raaphorst S.: *Reed-Muller Codes*, Carleton University, 2003.

[77] 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.

[78] Sade A.: *Quasigroupes parastrophiques. Expressions et identites*, Math. Nachr. 20, 1959, 73 - 106.

[79] Satti M.: *A Quasigroup Based Cryptographic System*, arXiv:cs/0610017v1 [cs.CR], 2006.

[80] Schneier B.: *Applied Cryptography*, John Wiley and Sons, Inc, 1996.

[81] Smith J. D. H.: *An introduction to quasigroups and their representations*, Academic Press, Inc., 1974.

[82] Stein S. K.: *On the foundations of quasigroups*, Trans. Amer. Math. Soc. 85, 1957, 228-256.

[83] Stinson D.R.: *Cryptography: Theory and Practice*, CRC Press, Inc., 1995.

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

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

[86] Бакева В.: *Прилог кон примената на веројатносни модели во системите за масовно опслужување, криптографијата и кодирањето*, Докторска дисертација, Универзитет ”Св. Кирил и Методиј”, Скопје, 2002.

[87] Димитров Б.: Вериги на Марков, Издателство ”Наука и искуство”, Софија, 1974.

[88] Димитрова В.: *Квазигрупно процесирани низи, нивна булова презентација и примена во криптографијата и кодирањето*, Докторска дисертација, Универзитет ”Св. Кирил и Методиј”, Скопје, 2010.

[89] Поповска-Митровиќ А.: *Споредба на перформансите на случајните кодови базирани на квазигрупи и кодовите на Рид-Милер и Рид-Соломон*, магистерски труд, Универзитет ”Св. Кирил и Методиј”, Скопје, 2009.

Прилози

Contents

List of Tables

List of Figures

Chapter 1

Quasigroups and Quasigroups string transformation

Quasigroups and quasigroup transformations are very useful for construction of cryptographic primitives, error detecting and error correcting codes. The reasons for that are: the structure of quasigroups, their large number, the properties of quasigroup transformations and others. In this thesis we consider some applications of quasigroups in cryptography and coding theory. Therefore, in this chapter, we will give the definitions and properties of the quasigroups and quasigroups string transformation that we will use in the next chapters. More for quasigroups and their mathematical background are given in [6, 11, 36, 81].

1.1 Quasigroups

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 $u, v \in Q$.

In the sequel we assume that the set Q is a finite set. The cardinality $|Q|$ of this set is called an order of the quasigroup. The main body of the multiplication table of a quasigroup is a Latin square over the set Q.

Given a quasigroup $(Q, *)$ a new operation $\backslash$ called a parastrophe, can be derived from the operation $*$ as follows:

$$x * y = z \Leftrightarrow y = x \backslash z. \tag{1.2}$$

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

$$x \backslash (x * y) = y \quad \text{and} \quad x * (x \backslash y) = y, \tag{1.3}$$

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*

$$x * y = y * x. \tag{1.4}$$

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

$$(x * y) * (y * x) = const. \tag{1.5}$$

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

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

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

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

Definition 1.5. *A quasigroup $(Q, *)$ is a loop if it has an identity element $e \in Q$ such that*

$$(\forall\, x \in Q)\ (x * e = x = e * x) \tag{1.8}$$

Let note that $(Q, *)$ is a loop iff it is a left loop and right loop.

Definition 1.6. *A quasigroup $(Q, *)$ is a right-symmetric quasigroup if it satisfies the identity*

$$(x * y) * y = x. \tag{1.9}$$

Definition 1.7. *A quasigroup $(Q, *)$ is a left-symmetric quasigroup if it satisfies the identity*

$$y * (y * x) = x. \tag{1.10}$$

Definition 1.8. *A quasigroup $(Q, *)$ is a totally symmetric if it is a commutative and a left-symmetric quasigroup (or a commutative and a right-symmetric quasigroup) .*

1.2 Quasigroups string transformation

Quasigroup string transformations are defined on a finite set Q (i.e., an alphabet Q) endowed with a quasigroup operation $*$, and they are mappings from Q^+ to Q^+, where Q^+ is the set of all nonempty words on Q. Two quasigroup string transformations are defined ([46]). Let $l \in Q$ be a fixed element, called a leader. For every $a_i, b_i \in Q$, E_l- and D_l- transformations are defined as follows:

$$E_l(a_1a_2\dots a_n) = (b_1b_2\dots b_n) \Leftrightarrow \left\{ \begin{array}{lcllll} b_1 & = & l & * & a_1 & \\ b_i & = & b_{i-1} & * & a_i, & 2 \leqslant i \leqslant n \end{array} \right. \tag{1.11}$$

$$D_l(a_1a_2\dots a_n) = (c_1c_2\dots c_n) \Leftrightarrow \left\{ \begin{array}{lcllll} c_1 & = & l & * & a_1 & \\ c_i & = & a_{i-1} & * & a_i, & 2 \leqslant i \leqslant n \end{array} \right. \tag{1.12}$$

By using the identities (1.3) we have that:

Theorem 1.1. *For each $M \in Q^+$ and each leader $l \in Q$ $D_{l,\backslash}(E_{l,*}(M)) = M = E_{l,*}(D_{l,\backslash}(M))$, i.e, $E_{l,*}$ and $D_{l,\backslash}$ are permutations on Q^+, mutually inverse.*

For given quasigroup operations $*_1, *_2, \dots, *_n$ on the set Q we can define mappings $E_1, E_2, \dots, E_n$, in the same manner as previous by choosing fixed elements $l_1, l_2, \dots, l_n \in Q$ (such that E_i is corresponding to $*_i$ and l_i). Then

$$E^{(n)} = E^{(n)}_{l_n,\dots,l_1} = E_n \circ E_{n-1} \circ \dots \circ E_1,$$

where $\circ$ is the usual composition of mappings ($n \geqslant 1$).

Similarly, we can define composition $D^{(n)}$ of quasigroup transformations D_1, $D_2, \dots, D_n$ with leaders $l_1, l_2, \dots, l_n$, respectively:

$$D^{(n)} = D^{(n)}_{l_n,\dots,l_1} = D_n \circ D_{n-1} \circ \dots \circ D_1.$$

Example 1.1. Let $Q = \{0, 1, 2, 3\}$ and $(Q, *)$ and its parastrophe $(Q, \backslash)$ be the quasigroups given on (1.13).

$*$	0	1	2	3
0	0	1	2	3
1	1	2	3	0
2	2	3	0	1
3	3	0	1	2

$\backslash$	0	1	2	3
0	0	1	2	3
1	3	0	1	2
2	2	3	0	1
3	1	2	3	0

(1.13)

	$1\,0\,2\,1\,0\,0\,0\,0\,0\,0\,0\,1\,1\,1\,1\,1\,2\,1\,0\,2\,2\,0\,1\,0\,1\,3\,0 = \alpha$
0	$1\,1\,3\,0\,0\,0\,0\,0\,0\,0\,0\,1\,2\,3\,0\,1\,3\,0\,0\,2\,0\,0\,1\,1\,2\,1\,1 = E_{0,*}(\alpha)$
0	$1\,2\,1\,1\,1\,1\,1\,1\,1\,1\,1\,2\,0\,3\,3\,0\,3\,3\,3\,1\,1\,1\,2\,3\,1\,2\,3 = E^{(2)}_{0,*}(\alpha)$
0	$1\,3\,0\,1\,2\,3\,0\,1\,2\,3\,0\,2\,2\,1\,0\,0\,3\,2\,1\,2\,3\,0\,2\,1\,2\,0\,3 = E^{(3)}_{0,*}(\alpha)$
0	$1\,0\,0\,1\,3\,2\,2\,3\,1\,0\,0\,2\,0\,1\,1\,1\,0\,2\,3\,1\,0\,0\,2\,3\,1\,1\,0 = E^{(4)}_{0,*}(\alpha)$

Table 1.1: Consecutive E-transformations

Let $\alpha = 1\,0\,2\,1\,0\,0\,0\,0\,0\,0\,0\,1\,1\,1\,1\,1\,2\,1\,0\,2\,2\,0\,1\,0\,1\,3\,0$ and $l = 0$ is a leader. Then four successive applications of the E-transformation are given in Table 1.1.

Strings obtained by four consecutive applications of D-transformations, $D_{0,\backslash}$, on the last string in Table 1.1 $\beta = E^{(4)}_{0,*}(\alpha)$, are given in Table 1.2.

	$1\,0\,0\,1\,3\,2\,2\,3\,1\,0\,0\,2\,0\,1\,1\,1\,0\,2\,3\,1\,0\,0\,2\,3\,1\,1\,0 = \beta$
0	$1\,3\,0\,1\,2\,3\,0\,1\,2\,3\,0\,2\,2\,1\,0\,0\,3\,2\,1\,2\,3\,0\,2\,1\,2\,0\,3 = D_{0,\backslash}(\beta)$
0	$1\,2\,1\,1\,1\,1\,1\,1\,1\,1\,1\,2\,0\,3\,3\,0\,3\,3\,3\,1\,1\,1\,2\,3\,1\,2\,3 = D^{(2)}_{0,\backslash}(\beta)$
0	$1\,1\,3\,0\,0\,0\,0\,0\,0\,0\,0\,1\,2\,3\,0\,1\,3\,0\,0\,2\,0\,0\,1\,1\,2\,1\,1 = D^{(3)}_{0,\backslash}(\beta)$
0	$1\,0\,2\,1\,0\,0\,0\,0\,0\,0\,0\,1\,1\,1\,1\,1\,2\,1\,0\,2\,2\,0\,1\,0\,1\,3\,0 = D^{(4)}_{0,\backslash}(\beta)$

Table 1.2: Consecutive D-transformations

From Table 1.2 we can conclude that $\alpha = D^{(4)}_{0,\backslash}(\beta) = D^{(4)}_{0,\backslash}(E^{(4)}_{0,*}(\alpha))$.

1.3 Some properties of quasigroup string transformations

In [47] authors proposed a transformation $E^{(n)}$ as an encryption function and proved the following theorem.

Theorem 1.2. *Let $\alpha = a_1a_2...a_k \in Q^+$ be an arbitrary string and $\beta = E^{(n)}(\alpha)$. If k is a large enough then m-tuples in β are uniformly distributed for $m \leqslant n$.*

Theorem 1.2 states that after n applications of E-transformation on an arbitrary message, the distribution of the substrings in the obtained message is uniform. This property is very important for application of some transformation in cryptography. Namely, the uniform distribution of the substrings in the encrypted message guarantees the resistance to statistical attacks. In the same paper, the authors remarked that the distribution of the substrings of length m is not uniform for $m > n$. Since, the distribution of these substrings is not uniform it can be used for attack on the encrypted message in order to discover the original message.

In [2] authors found the distribution of $(n+1)$-tuples after application of $E^{(n)}$-transformation and proved the following theorem and corollary.

Theorem 1.3. *Let $(p_1, p_2, \ldots, p_s)$ be the distribution of letters in an input string α and let $p_1, p_2, \ldots, p_s$ be distinct probabilities, i.e., $p_i \neq p_j$ for $i \neq j$. The probabilities of $(n+1)$-tuples in output message $\beta = E^{(n)}(\alpha)$ are divided in s classes. Each class contains s^n elements with the same probabilities and the probability of each $(n+1)$-tuple in i-th class is $\frac{1}{s^n}p_i$, for $i = 1, 2, \ldots, s$.*

Corrolary 1.1. *If $p_{i_1} = p_{i_2} = \cdots = p_{i_\nu}$ for some $1 \leqslant i_1 < \cdots < i_\nu \leqslant s$ in Theorem 1.3, then the classes with probabilities $\frac{1}{s^n}p_{i_1}$, $\frac{1}{s^n}p_{i_2}$,..., $\frac{1}{s^n}p_{i_\nu}$ will be merged in one class with νs^n elements.*

In the same paper ([2]) authors gave a proof for a more general form of these results, i.e., they found the distribution of m-tuples ($m > n$) after application of $E^{(n)}$-transformation. This generalization of the previous results is given in the following theorems.

Theorem 1.4. *Let $(p_1, p_2, \ldots, p_s)$ be the distribution of the letters in an input string α and let K be the number of all distinct products $p_{i_1} \ldots p_{i_{m-n}}$ for $i_1, i_2, \ldots, i_{m-n} \in A$. Then the probabilities of m-tuples $(m > n)$ in output message $\beta = E^{(n)}(\alpha)$ are divided in K classes. The m-tuples in a class have the same probabilities.*

Corrolary 1.2. *Let r be a number of different probabilities $p_1, p_2, \ldots, p_s$ of letters in the alphabet A. If all products $p_{i_1} \ldots p_{i_{m-n}}$ are distinct then the number of classes is $\overline{C}_r^{m-n}$, where $\overline{C}_r^k$ is the total number of k-combinations with repetitions of r-element set.*

From the proof of the previous theorem (given in [2]) follows that the probability of m-tuples is $\frac{1}{s^n}p_1^{n_1}p_2^{n_2}\ldots p_r^{n_r}$ where $n_1 + n_2 + \cdots + n_r = m - n$. Also,

theorems for number of elements in each class are proven. Using these theoretical results, authors in the same paper gave the following algorithm for statistical attack.

Let the message M be encrypted using encryption algorithm based on $E^{(n)}$-quasigroup transformation for fixed n and let the obtained message $C = E^{(n)}(M)$ be send. Let an intruder catch the encrypted message C and s/he knows that encryption algorithm based on $E^{(n)}$-quasigroup transformation is used. If we suppose that the distribution of the letters $(p_1, p_2, \ldots, p_s)$ in language used in original message is known, then the intruder can do a statistical attack using the algorithm given in Table 1.3.

The first two letters from the original message M can not be find in this way, but they can be easily found by questing (if it is important for the message).

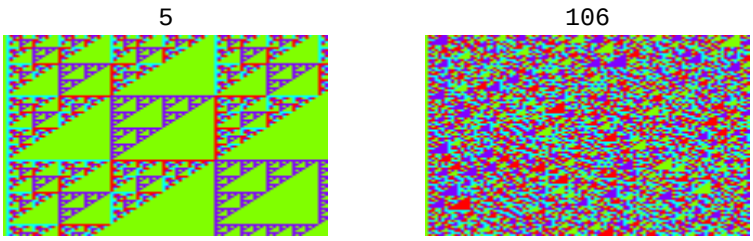
For application of this algorithm the message C must be long enough in order to obtain relative frequencies of $(n+1)$-tuples as much as closer to their probabilities in the language. The real statistical analysis cannot be done if C is a short message. Then the decrypted message M_1 will not correspond to the original message M.

Algorithm of statistical attack	
Input:	An encrypt message C and the distribution $(p_1, p_2, \ldots, p_s)$ of the letters in language.
Output:	The original message M.
Step 1.	Find the relative frequencies (statistical probabilities) of letters, pairs, 3-tuples and so on, until you obtain distribution which is not uniform. Let say that n is the smallest number such that $(n+1)$-tuples are not uniformly distributed.
Step 2.	Process whole message C in the following way. Take the first $(n+1)$-tuple and find its relative frequency f.
	2.1. Find i such that $\lvert f - \frac{1}{s^n} p_i \rvert = \min\limits_{1 \leqslant j \leqslant s} \lvert f - \frac{1}{s^n} p_j \rvert$.
	2.2. Decrypt this $(n+1)$-tuple with the letter i.
	2.3. Take the next $(n+1)$-tuple and find its relative frequency f and go to step 2.1.

Table 1.3: Algorithm for statistical attack

1.4 Classification of quasigroups of order 4 by fractality and linearity

Using an image pattern, Dimitrova and Markovski give a classification of quasigroups of order 4 in [17] as fractal and non-fractal. This classification is made in the following way. The authors start with a periodical sequence 123412341... of length 100 and apply 100 times E-transformation given in (1.11) with given leaders. They present the transformed sequences visually using different color for each symbol 1,2,3,4. In this way, they obtain an image pattern for each quasigroup and then analyze the structure of these patterns. If the pattern has a fractal structure, the related quasigroup is called fractal. In the opposite case, the quasigroup is called non-fractal. The number of fractal quasigroups of order 4 is 192 and the number of non-fractal quasigroups is 384. In Figure 1.1 we present an example for image pattern of a fractal (a)) and a non-fractal (b)) quasigroup.



a) b)

Figure 1.1: Fractal and non-fractal quasigroup

In [21], 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 of the quasigroup as binary vector $x = (x_1, x_2, \ldots, x_n) \in \{0,1\}^n$. Then, for each $x, y \in Q$

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

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 [21], the quasigroups are divided into two classes: linear quasigroups and non-linear quasigroups by Boolean representation. Let $(Q, *)$ be a quasigroup of order 2^n and let

$$f(x_1, \ldots, x_{2n}) = (f_1(x_1, \ldots, x_{2n}), \ldots, f_n(x_1, \ldots, x_{2n}))$$

be its corresponding representation as vector valued Boolean function. The two types of quasigroups are defined as follows.

Definition 1.9. *The quasigroup is linear if all functions f_i for $i = 1, 2, \ldots n$ are linear polynomials.*

Definition 1.10. *The quasigroup is non-linear if there exist function f_i for some $i = 1, 2, \ldots n$ which is not linear.*

There are 144 linear and 432 non-linear quasigroups of order 4. 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. The quasigroup is pure non-linear if all its components are non-linear Boolean functions.

Chapter 2

Error-correcting codes based on quasigroups

In this chapter we will describe Random Codes Based on Quasigroups (RCBQ). We will explain the way of adding the redundant information, the algorithms of coding and decoding. RCBQs are proposed by Danilo Gligoroski, Smile Markovski and Ljupcho Kocarev in [30]. These codes have several parameters and they are a combination of cryptographic algorithms and error-correcting codes.

A usual way to obtain error-correcting codes resistant to an intruder attack consists in application of some of the known ciphers on the codewords, before sending them through an insecure channel. Then two algorithms are used, one for error-correcting codes and another for obtaining information security. The concept of cryptcoding combines the processes of encoding and encryption in one algorithm. RCBQs are crypt-codes. Namely, they are defined by using a cryptographic algorithm during the encoding/decoding process and they allow not only correction of certain amount of errors in the input data, but they also provide an information security, all build in one algorithm. If the data are encoded with these codes, then the recipient can decode the original data only if s/he knows exactly which parameters are used in the process of encoding, even the communication channel is without noise. Here, we are considering RCBQs as error-correcting codes, and consequently we do not analyze its cryptographic properties.

RCBQs are designed using algorithms for encryption/decryption from the implementation of TASC (Totally Asynchronous Stream Ciphers) by quasigroup string transformations [27]. These cryptographic algorithms use the alphabet Q and a quasigroup operation $*$ on Q together with its parastrophe $\backslash$. We will give description of these codes using quasigroups, but from the definition of the algorithms it is clear that in their design other algorithms for encryption and

decryption can be used. In the code design authors use the alphabet of nibbles $Q = \{0, 1, \ldots, 9, a, b, c, d, e, f\}$. But, here we will give the encoding and decoding algorithms for a more general case where we use a-bit letters instead of nibbles.

In my master thesis [89] and [68] we investigated the influence of the code parameters on the code performances and we proposed a method for reducing the unsuccessful decodings. Here, we briefly give some of these results. Also, in this chapter we will investigate the influence of the length of the messages on the code performances.

2.1 TASC и EdonZ

The concept of TASC was introduced in [27]. That cryptographic concept is the corner stone for the random codes based on quasigroups. Therefore in this section we will briefly describe these ciphers.

TASC is a new concept of a stream cipher, different to the concepts of synchronous and self-synchronized stream ciphers.

Definition 2.1. *A synchronous stream cipher is one in which the keystream is generated independently of the plaintext message and of the ciphertext. The encryption process of a synchronous stream cipher can be described by the equations*

$$\sigma_{i+1} = f(\sigma_i, k), \quad z_i = g(\sigma_i, k), \quad c_i = h(z_i, m_i),$$

where σ_0 is the initial state and may be determined from the key k, f is the next-state function, g is the function which produces the keystream z_i, and h is the output function which combines the keystream and plaintext m_i to produce ciphertext c_i.

Definition 2.2. *A self-synchronizing or asynchronous stream cipher is one in which the keystream is generated as a function of the key and a fixed number of previous ciphertext digits. The encryption function of a self-synchronizing stream cipher can be described by the equations:*

$$\sigma_i = (c_{i-l}, c_{i-l+1}, ..., c_{i-1}), \quad z_i = g(\sigma_i, k), \quad c_i = h(z_i, m_i),$$

where $\sigma_0 = (c_{-l}, c_{-l+1}, ..., c_{-1})$ is the initial state, k is the key, g is the function which produces the keystream z_i, and h is the output function which combines the keystream and plaintext m_i to produce ciphertext c_i.

Definition 2.3. *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 that 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. EdonZ works with the alphabet $Q = \{0, 1, \ldots, 9, a, b, c, d, e, f\}$ of nibbles, whose elements are 4-bit words. The encryption function uses the quasigroup $(Q, *)$ and the decryption function uses its parastrophe $(Q, \backslash)$, as well as the operation $\oplus$ of eXclusive OR (XOR) on 4-bit variables. The encryption/decryption functions of EdonZ, by its algorithms, are given in Figure 2.2.

We must note the following property of EdonZ. Let $M = M_1 M_2 ... M_k$ be a representation of the message M into sub-blocks M_i where $M_1 = m_1 ... m_{i_1}$, $M_2 = m_{i_{1+1}} ... m_{i_2}, ...$, and let C be the cipher text of M. Then $C = C_1 C_2 ... C_k$ where each C_i is a cipher text of the corresponding plain text M_i. This property means that even EdonZ is a stream cipher, we can process the input plain text in smaller parts (blocks) and the final cipher text does not depend on the chosen division of M into blocks (before encryption). This property is used in the introduction of the redundant information in the process of coding of RCBQ.

The main characteristic of TASC is that the error propagation is unbounded and propagates until the end of the stream. However, by adding some redundant information in the stream, the correction of some errors can be done. That is in fact the main idea behind TASC Error Correction.

2.2 Description of coding

First we divide the sequence of bits, obtained from the information source, into messages (blocks) of N_{block} bits. We map each of the $2^{N_{block}}$ blocks in codeword

of length $N > N_{block}$ bits.

Let $M = m_1 m_2 ... m_l$ be a block of $N_{block} = la$ bits, where $m_i \in Q$ and Q is the alphabet of all a-bit symbols (so, Q has 2^a letters). First, we add a redundancy of v a-bit zero symbols and produce a block $L = L^{(1)} L^{(2)} ... L^{(s)} = L_1 L_2 ... L_m$ of N bits, where $L^{(i)}$ are sub-blocks of r symbols from Q and $L_i \in Q$. Note that $N = ma$, $m = l + v$ and $m = rs$. After erasing the redundant zeros from each $L^{(i)}$, the message L will produce the original message M. In this way we obtain an (N_{block}, N) code with rate $R = N_{block}/N$. The codeword is produced after applying the encryption algorithm (given in Figure 2.2) on the block L. For that aim, previously, a key $k = k_1 k_2 \dots k_n \in Q^n$ should be chosen. The obtained codeword of M is $C = C_1 C_2 ... C_m$, where $C_i \in Q$.

The coding process is schematically presented in Figure 2.1.

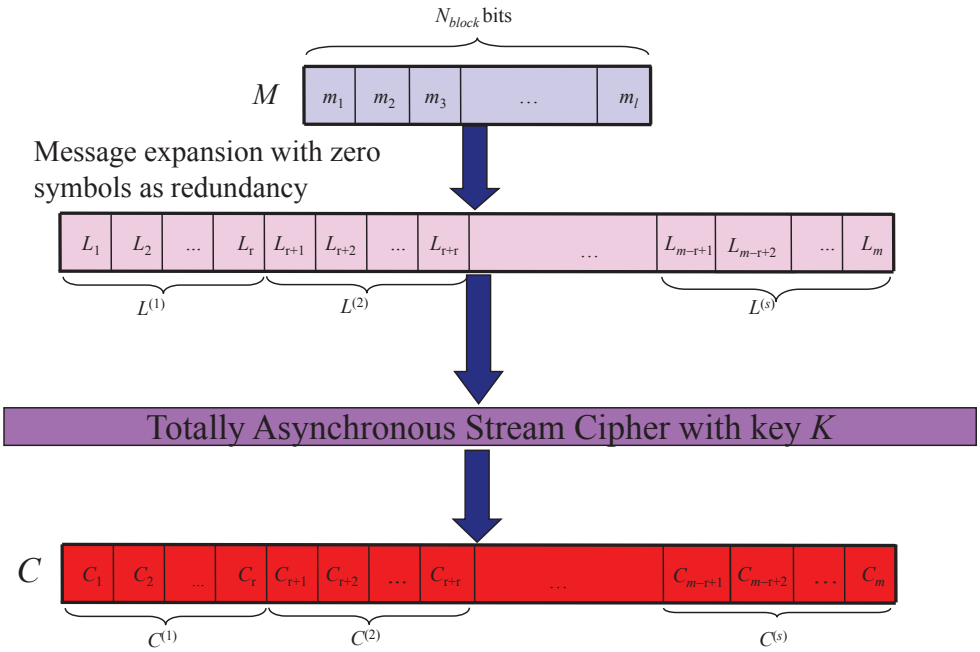


Figure 2.1: Coding with standard algorithm

2.3 Description of decoding

After transmission through a noise channel (for our experiments we a use binary symmetric channel), the codeword C will be transformed to a received message $D = D^{(1)}D^{(2)}...D^{(s)} = D_1D_2...D_m$, where $D^{(i)}$ are sub-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.

Encryption	**Decryption**
Input: Key $k = k_1k_2 \dots k_n$ and $L = L_1L_2 \dots L_m$ **Output**: codeword $C = C_1C_2...C_m$	**Input**: The pair $(a_1a_2 \dots a_s, k_1k_2 \dots k_n)$ **Output**: The pair $(c_1c_2 \dots c_s, K_1K_2 \dots K_n)$
For $j = 1$ to m $X \leftarrow L_j$; $T \leftarrow 0$; For $i = 1$ to n $X \leftarrow k_i * X$; $T \leftarrow T \oplus X$; $k_i \leftarrow X$; $k_n \leftarrow T$ **Output**: $C_j \leftarrow X$	For $i = 1$ to n $K_i \leftarrow k_i$; For $j = 0$ to $s - 1$ $X, T \leftarrow a_{j+1}$; $temp \leftarrow K_n$; For $i = n$ to 2 $X \leftarrow temp \setminus X$; $T \leftarrow T \oplus X$; $temp \leftarrow K_{i-1}$; $K_{i-1} \leftarrow X$; $X \leftarrow temp \setminus X$; $K_n \leftarrow T$; $c_{j+1} \leftarrow X$; **Output**: $(c_1c_2 \dots c_s, K_1K_2 \dots K_n)$

Figure 2.2: Algorithms for encryption and decryption

Generating sets with predefined Hamming distance – The probability that maximum t bits in $D^{(i)}$ are not correctly transmitted is

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

where p is the probability of bit-error in a binary symmetric channel. Let B_{max} be an integer such that $1 - P(p; B_{max}) \leqslant q_B$, where q_B $(0 < q_B \leqslant 1)$ is given. Consider the sets

$$H_i = \{\alpha | \alpha \in Q^r, \;\; H(D^{(i)}, \alpha) \leqslant B_{max}\},$$

for $i = 1, 2, \ldots, s$, where $H(D^{(i)}, \alpha)$ is the Hamming distance between $D^{(i)}$ and α. Then, with probability at least $1 - q_B$ the block $C^{(i)}$ is an element of the set H_i, for $i = 1, 2, \ldots, s$. 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 possible.

Inverse coding algorithm – The inverse coding algorithm is the decrypting algorithm of TASC given in Figure 2.2.

Generating decoding candidate sets –The decoding candidate sets S_0, S_1,..., S_s are defined iteratively. Let $S_0 = (k_1 \ldots k_n; \lambda)$, where λ is the empty sequence. Let S_{i-1} be defined for $i \geqslant 1$. Then S_i is the set of all pairs $(\delta, w_1 w_2 \ldots w_{rai})$ obtained by using the sets S_{i-1} and H_i as follows (w_j are bits). For each $(\beta, w_1 w_2 \ldots w_{ra(i-1)}) \in S_{i-1}$ and each element $\alpha \in H_i$, we apply the inverse coding algorithm (i.e. algorithm for decryption given in Figure 2.2) with input (α, β). If the output is the pair (γ, δ) and if both sequences γ and $L^{(i)}$ have the redundant zeros in the same positions, then the pair $(\delta, w_1 w_2 \ldots w_{ra(i-1)} c_1 c_2 \ldots c_{ra}) \equiv (\delta, w_1 w_2 \ldots w_{rai})$ 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 *uncorrected-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 error of type *null-error* appears). We conclude that for some $i \leqslant j,\ D^{(i)}$ contains more than B_{max} errors, resulting in $C_i \notin H_i$.

Theorem 2.1. [30] *The packet-error probability of these codes is*

$$PER_t = 1 - (1 - q_B)^s.$$

From the proof of this theorem given in [30], it is clear that in the probability PER_t the *more-candidate-errors* are not provided.

2.4 Choosing parameters for optimal RCBQ

Error-correcting codes based on quasigroups have several parameters, and we have investigated the influence of the code parameters on the code performances. Since they are designed using quasigroup string transformations on messages extended by introduced redundancy, we have pointed out how

– the pattern of the redundancy

– the length of the initial key

– the chosen quasigroup

affect the code performances. In these experiments we use the alphabet of nibbles.

We have made experiments in the following way. First, we extend input message using different patterns for redundant zero nibbles, and after that we encode the extended message and transmit it through a binary symmetric channel with probability p of bit error. The outgoing message is decoded and if the decoding

process is completed successfully (the last set S_s of candidates for decoding has only one element), the decoded message is compared with the input message. If they differ in at least one bit (*uncorrected-error* appears) we compute the number of incorrectly decoded bits as Hamming distance between the input and the decoded message. Experiments showed that this type of package error occurs rarely.

In our experiments we also calculate the number of incorrectly decoded bits when the decoding process finishes with *more-candidate-error* or *null-error*. Then, that number is calculated as follows.

When *null-error* appears, i.e., $S_i = \emptyset$, we take all the elements (without redundant symbols) from the set S_{i-1} and we find their maximal common prefix substring. If this substring has k bits and the length of the sent message is m bits ($k \leqslant m$), then we compare this substring with the first k bits of the sent message. If they differ in b bits, then the number of incorrectly decoded bits is $m - k + b$.

If a *more-candidates-error* appears we take all the elements from the set S_s and we find their maximal common prefix substring. The number of incorrectly decoded bits is computed as previously.

The total number of incorrectly decoded bits is the sum of all previously mentioned numbers of incorrectly decoded bits.

We compute the probability of packet-error as

$$PER = \frac{\#(\textit{incorrectly decoded packets})}{\#(\textit{all packets})}$$

and the probability of bit-error as

$$BER = \frac{\#(\textit{incorrectly decoded bits in all packets})}{\#(\textit{bits in all packets})}.$$

The experimental results are presented in the tables below. In each table we give:

- theoretical probability of packet-error (PER_t);
- experimental probability of packet-error without errors of type *more-candidates-error* ($PER_{s\text{-}1}$);

– experimental probability of packet-error with all unsuccessfully decoded blocks (PER_s);

– experimentally obtained probability of cumulative bit-error (BER_s).

We made experiments for different values of bit-error probability p of binary symmetric channel and $B_{max} = 3$ and $B_{max} = 4$ until $BER_s < p$. For $B_{max} > 4$, the experiments do not terminate in real time.

2.4.1 Redundancy pattern

We made experiments for 6 different patterns for redundant zero nibbles for code (72,288) with rate R=1/4. In these experiments we have used the quasigroup given in Table 2.1 (and its parastrophe given in Table 2.2), the initial key $k = 01234$ and the 6 patterns given in Table 2.3.

$*$	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

Table 2.1: 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	8	7	2	0	d	3	6	5	c	e	f	9	1	a	b	4
1	0	5	a	1	f	9	8	6	4	2	b	7	c	3	e	d
2	1	0	f	9	4	5	a	b	d	7	c	e	3	8	2	6
3	b	3	c	8	5	f	0	9	a	4	7	1	d	e	6	2
4	2	f	9	7	0	1	4	3	b	6	a	5	e	c	d	8
5	3	2	5	a	6	c	f	8	e	d	1	b	7	9	4	0
6	7	d	0	3	b	e	c	f	5	a	2	8	4	6	9	1
7	d	4	b	f	c	8	7	e	6	1	3	a	2	5	0	9
8	9	8	3	e	a	7	2	1	f	b	4	6	0	d	c	5
9	f	6	e	5	2	a	b	c	8	3	d	0	9	4	1	7
a	4	9	d	b	1	6	5	7	3	0	e	c	f	2	8	a
b	a	e	4	6	7	2	9	0	1	f	5	d	8	b	3	c
c	6	c	1	d	e	0	3	4	9	5	8	2	a	f	7	b
d	c	a	8	4	3	b	1	d	2	9	0	f	6	7	5	e
e	5	1	6	2	8	d	e	a	7	c	9	4	b	0	f	3
f	e	b	7	c	9	4	d	2	0	8	6	3	5	1	a	f

Table 2.2: Parastrophe of the quasigroup given in Table 2.1

In the patterns, given in Table 2.3, with 1 we denote the place of a message (information) symbol, and with 0 the redundant zero symbol.

patt.1	**patt.2**	**patt.3**	**patt.4**	**patt.5**	**patt.6**
1000 1000	1100 1100	1100 1100	1100 1100	1100 1000	1100 1100
1000 1000	0000 1100	1000 0000	1100 0000	0000 1100	1000 0000
1000 1000	1100 0000	1100 1000	0000 1100	1000 0000	1100 1100
1000 1000	1100 1100	1000 0000	1100 1100	1100 1000	1000 0000
1000 1000	0000 1100	1100 1100	0000 0000	0000 1100	1100 1100
1000 1000	1100 0000	1000 0000	1100 1100	1000 0000	1000 0000
1000 1000	1100 0000	1100 1000	1100 0000	1100 1000	1000 1000
1000 1000	0000 0000	1000 0000	0000 0000	0000 1100	1000 0000
1000 1000	0000 0000	0000 0000	0000 0000	1000 0000	0000 0000

Table 2.3: Patterns of redundancy

Results obtained with the first pattern are given in Table 2.4. They show that

this pattern gives many unsuccessfully completed decodings with *more-candidates-error*. But, we have to note that the cardinality of the sets S_i in the process of decoding is small, so the decoding process takes much less time than for the other patterns.

$B_{max} = 3$				
p	PER_t	PER_{s_1}	PER_s	BER_s
0.02	0.004314	0.004392	0.1185916	0.008851
0.03	0.019674	0.019153	0.1329925	0.016889
0.04	0.055435	0.055659	0.1720190	0.036934
0.05	0.118838	0.117728	0.2258065	0.071293
$B_{max} = 4$				
0.03	0.001447	0.000792	0.6057028	0.084506
0.04	0.005541	0.004896	0.6107431	0.085522

Table 2.4: Experimental results for **patt.1**

$B_{max} = 3$				
p	PER_t	PER_{s_1}	PER_s	BER_s
0.02	0.004314	0.004248	0.0060484	0.003320
0.03	0.019674	0.020449	0.0225374	0.010699
0.04	0.055435	0.054147	0.0559476	0.025255
0.05	0.118838	0.116431	0.1183036	0.053244
$B_{max} = 4$				
0.03	0.001447	0.000500	0.2425000	0.221236
0.04	0.005541	0.004902	0.2504456	0.225082

Table 2.5: Experimental results for **patt.2**

From the results for the second pattern (presented in Table 2.5) we can see

that for $B_{max} = 3$ with this pattern, we get better results than with the first one. For $B_{max} = 4$, the decoding process often ends unsuccessfully with a *more-candidates-error*, although this pattern has a sufficient number of zero nibbles at the end. Also, for $B_{max} = 4$ with this pattern we received larger values of BER_s compared with the results for the first pattern.

Results obtained with the third pattern are given in Table 2.6. The probabilities of packet-error and bit-error for this pattern are much smaller than for the previous two patterns.

$B_{max} = 3$				
p	PER_t	PER_{s_1}	PER_s	BER_s
0.02	0.004314	0.003888	0.0050403	0.00268
0.03	0.019674	0.018073	0.0191532	0.00949
0.04	0.055435	0.052275	0.0531394	0.02529
0.05	0.118838	0.119599	0.1201037	0.05558
$B_{max} = 4$				
0.03	0.001447	0.001658	0.0129353	0.00969
0.04	0.005541	0.005484	0.0170968	0.01128
0.05	0.015319	0.021759	0.0323283	0.02188
0.06	0.034361	0.037812	0.0484375	0.03051
0.07	0.066467	0.065333	0.0756667	0.04645
0.08	0.114889	0.119062	0.1281250	0.07662

Table 2.6: Experimental results for **patt.3**

Results for the fourth considered pattern are given in Table 2.7. The fourth pattern has enough zero symbols after blocks with information nibbles and at the end of the message, but again experimentally obtained probabilities of packet and bit error are good only for $B_{max} = 3$.

$B_{max} = 3$				
p	PER_t	PER_{s_1}	PER_s	BER_s
0.02	0.004314	0.005112	0.0115927	0.006317
0.03	0.019674	0.020593	0.0273617	0.014002
0.04	0.055435	0.056596	0.0623559	0.030743
0.05	0.118838	0.118159	0.1250000	0.061019
$B_{max} = 4$				
0.03	0.001447	0.002273	0.1859091	0.147449
0.04	0.005541	0.006364	0.1959091	0.149223

Table 2.7: Experimental results for **patt.4**

Experimental results for PER_s (Table 2.8) obtained by the fifth pattern are similar to the results obtained by previous pattern, except the results for the probabilities of bit-error, that are better.

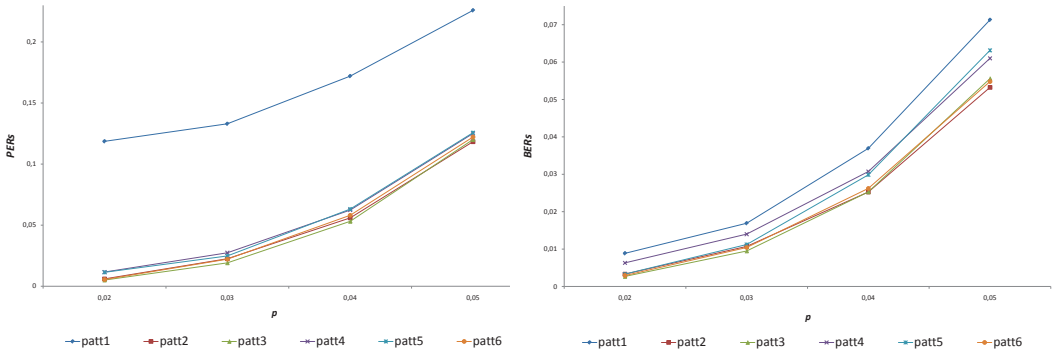
$B_{max} = 3$				
p	PER_t	PER_{s_1}	PER_s	BER_s
0.02	0.004314	0.005184	0.0113767	0.003302
0.03	0.019674	0.019441	0.0247696	0.011202
0.04	0.055435	0.058179	0.0631480	0.029845
0.05	0.118838	0.121039	0.1257201	0.063158
$B_{max} = 4$				
0.03	0.001447	0.000625	0.2534375	0.047635
0.04	0.005541	0.005000	0.2606250	0.052908
0.05	0.015319	0.012188	0.2596875	0.053342
0.06	0.034361	0.030938	0.2615625	0.063424

Table 2.8: Experimental results for **patt.5**

$B_{max} = 3$				
p	PER_t	PER_{s_1}	PER_s	BER_s
0.02	0.004314	0.003672	0.0054723	0.002928
0.03	0.019674	0.020521	0.0221774	0.010418
0.04	0.055435	0.056164	0.0581077	0.026228
0.05	0.118838	0.120679	0.1221198	0.054786
$B_{max} = 4$				
0.03	0.001447	0.001667	0.0456667	0.038995
0.04	0.005541	0.006333	0.0533333	0.045074
0.05	0.015319	0.013333	0.0590000	0.048574
0.06	0.034361	0.035000	0.0800000	0.063222
0.07	0.066467	0.063667	0.1103333	0.082676

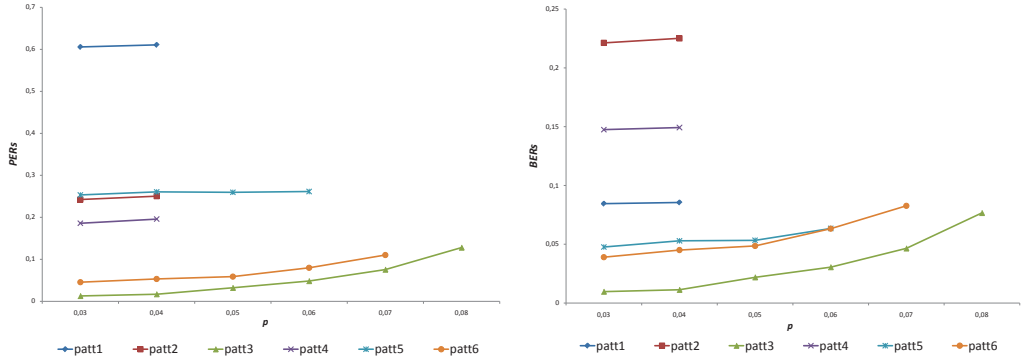
Table 2.9: Experimental results for **patt.6**

Results obtained with the sixth pattern (given in Table 2.9) show that this pattern gives good results for $B_{max} = 3$ and $p < 0.05$ and $B_{max} = 4$ and $p < 0.07$.



a) b)

Figure 2.3: PER_s and BER_s for all six patterns and $B_{max} = 3$



a) b)

Figure 2.4: PER_s and BER_s for all six patterns and $B_{max} = 4$

From the experimental results for all six proposed patterns (presented in Figure 2.3 for $B_{max} = 3$ and Figure 2.4 for $B_{max} = 4$), we can conclude that the best results for PER and BER are obtained for the third pattern, especially for $B_{max} = 4$.

2.4.2 Key length

The theoretical probability of packet-error given in Theorem 2.1 is determined under the assumption that the code is perfectly random (i.e., the r-tuples are uniformly distributed in each codeword with length N, $r \leqslant N$). Therefore, in that theorem the *more-candidates-errors* are not provided. In Theorem 1.2 it is proved that if we apply t quasigroup transformations on a string, we obtain string where n-tuples of letters are uniformly distributed for $n \leqslant t$. In the design of these codes, the length of the key k determines how many quasigroup transformations will be applied in forming of a codeword. Therefore, a longer key of the code gives a "more random" code. This means that the results of experimental PER will be closer to the theoretical values for PER, i.e., the number of *more-candidates-errors* will be reduced. So, we made experiments with the third pattern (which give the best results) with key length of 10. From the results given in Table 2.10 we can see that in some experiments *more-candidates-errors* have not appeared (in this case, in the table the values of PER_{s_1} and PER_s are given in bold), and if they appear, their number is very small (maximum 5 in experiment with 3500

messages and therefore we do not need to make experiments with longer key). We can conclude that when we use a longer key, we can obtain better results for PER with almost the same duration of the decoding process.

$B_{max} = 3$				
p	PER_t	PER_{s_1}	PER_s	BER_s
0.02	0.004314	**0.004752**	**0.004752**	0.002093
0.03	0.019674	**0.018433**	**0.018433**	0.008493
0.04	0.055435	**0.055588**	**0.055588**	0.026289
0.05	0.118838	**0.117584**	**0.117584**	0.054876
$B_{max} = 4$				
0.03	0.001447	0.002188	0.003125	0.001359
0.04	0.005541	0.005313	0.005938	0.003429
0.05	0.015319	0.014688	0.015938	0.009279
0.06	0.034361	**0.035938**	**0.035938**	0.022396
0.07	0.066467	0.065000	0.066563	0.040647
0.08	0.114889	0.112500	0.113125	0.064852

Table 2.10: Experimental results for key length 10

On the other hand, the key length is not unique parameter which has influence on the PER. Namely, we made an experiment with key length of 10 and the first pattern, but the number of *more-candidates-errors* was not smaller than the previous case with the shorter key. Hence, we can conclude that each parameter in this code design has great influence over the performances, i.e., the parameters are mutually dependent.

2.4.3 Choosing of a quasigroup

We also did experiments with several quasigroups, which showed that the choice of the quasigroup does not affect only the values of PER and BER, but they have a great influence on the speed of decoding. This confirms again that the choice of all parameters of these codes affects the performances of the code.

First we did experiments with the cyclic quasigroup (i.e., the group) of order 16 and the key length of 10. Decoding for the third pattern was too slow. So, we did experiment with the first pattern for a binary symmetric channel with $p = 0.02$ and $B_{max} = 3$, and we received $PER_s = 0.734087$ and $BER_s = 0.460359$ (that is much worse than $PER_s = 0.1186$, $BER_s = 0.0089$ obtained with the quasigroup in Table 2.1).

After that we made experiments with 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 for cyclic group. For the first pattern, $p = 0.02$ and $B_{max} = 3$ we got $PER_s = 0.99424$ and $BER_s = 0.80869$.

The cyclic group and the direct product of quasigroups of order 2 are examples of so called fractal quasigroups, that produce biased sequences. The quasigroup in Table 2.1 is an example of so called non-fractal quasigroups. The results obtained using this quasigroup were quite satisfactory.

From the previous examples, we can conclude that the choice of quasigroup has enormous influence over the performances of the code.

In the paper [15] the coefficient of period growth is analyzed. It represents how many times the period has grown (in average) after one application of the quasigroup transformation. At the end of this paper two examples of quasigroups of order 16 are given, one with a very high and one with a very low coefficient of period growth. We made experiments using these two quasigroups and the third pattern. From the obtained experimental results [68] we concluded that there is very small difference in the values of PER and BER, although the difference in the coefficients of period growth of these quasigroups is large. It seems that the coefficient of period growth has no big influence on the performances of the code, but this should be checked by using more quasigroups.

From the experiments, we can conclude that the best results for RCBQ(72,288) were obtained by the third pattern, the key length of 10, the quasigroup given in Table 2.1 (together with its parastrophe) and $B_{max} = 4$. Therefore, the further improvements of RCBQs will be compared with the results obtained for these parameters.

2.5 Method for decreasing the number of *null-errors*

All previous experiments with different patterns, key lengths and quasigroups allow us to see how these parameters affect on the number of *more-candidates-errors*. Changing these parameters does not have great influence on the number of

null-errors. Their number is determined with the theoretical probability of packet-error (given in Theorem 2.1), and that probability does not depend on these parameters.

Unsuccessful decoding with *null-error* occurs when in some of the sub-blocks of encoded message, more than predicted B_{max} bit errors appear during transmission. Therefore, it is clear that some of these errors can be eliminated if we cancel a few of iterations of the decoding process and we reprocess all of them or part of them with a larger value of B_{max}. With this procedure only part of these unsuccessful decoded messages will be eliminated since we cannot know exactly in which iteration the correct sub-block does not enter in the set of candidates for decoding and exactly how many transmission errors ($B_{max}+1$, $B_{max}+2$ or more) occur in this sub-block. Moreover, the cancellation of the iterations slows down the decoding, and the number of elements in sets S_i can become too large leading to unsuccessful decoding of type *more-candidates-error*.

To show this, we repeat the experiments with the third pattern in the following way. If we get an empty set in some iteration (for example i^{th}) of decoding, the two previous iterations ($(i-1)^{th}$ and $(i-2)^{th}$) are canceled. After that we reprocess the $(i-2)^{th}$ iteration with $B_{max} = B_{max} + 1$, and the next iterations continue with the old value of B_{max} . If an empty set appears again in the same iteration (i^{th} iteration), then we stop the process and decoding ends unsuccessfully. But if an empty set is obtained in a next iteration ($(i+1)^{th}$, $(i+2)^{th}$, ...) then the above procedure is repeated for that iteration again.

In Table 2.11 we give the percentage of eliminated unsuccessful decodings with *null-error* using described modification in the decoding process. We can conclude that this modification gives a significant elimination of *null-errors* and the decoding process is only slightly slower.

2.6 The influence of the length of the messages on the code performances

In this section we examine the influence of the length of the messages on the performances of error-correcting codes based on quasigroups. For this purpose we made experiments for codes with the same rate but different lengths of messages.

In the known error-correcting codes, the probabilities of bit-error and packet error decrease when the length of the message increases. In this section we will show that it is not the case for error-correcting codes based on quasigroups. In Theorem 2.1, the theoretical packet-error probability is given by the following

$B_{max} = 3$	
p	Percentage of eliminated *null-errors*
0.02	16.67 %
0.03	18.33 %
0.04	15.01 %
0.05	18.30 %
$B_{max} = 4$	
0.03	0.00 %
0.04	17.65 %
0.05	2.86 %
0.06	14.88 %
0.07	14.29 %
0.08	10.26 %

Table 2.11: Percentage of eliminated unsuccessfully decoding with *null-error*

formula

$$PER_t = 1 - (1 - q_B)^s.$$

From this formula, it is clear that for a larger number s of blocks in the codeword the packet-error probability is larger. This follows from the fact that in this case we have more iterations in the process of decoding and in each iteration the correct block should be in the corresponding set S. If the message M is longer, then in order to obtain the same rate R, the number of blocks in the redundant message must be larger and codeword is longer than for shorter messages. So, the number of iterations in the decoding process is bigger and we get larger packet-error probability. We check this result experimentally.

In Section 2.4, we gave experimental results, for different patterns of redundancy, for code (72,288) with rate $R = 1/4$ for binary symmetric channel. We concluded that the best results are obtained by the third pattern, the key length of 10, the quasigroup given in Table 2.1 (together with its parastrophe) and $B_{max} = 4$. For comparison we have made experiments with same quasigroup, key and rate

for code (144,576) with twice longer messages. In these experiments we used the following pattern:
1100 1100 1000 0000 1100 1000 1000 0000 1100 1100 1000 0000 1100 1000 1000 0000 0000 1100 1100 1000 0000 1100 1000 1000 0000 1100 1100 1000 0000 1100 1000 1000 0000 0000 0000 0000.

The obtained results for PER and BER for different values of bit-error probability p of binary symmetric channel and $B_{max} = 4$ are presented in Table 2.12 and Table 2.13.

p	$PER_s(72, 288)$	$PER_s(144, 576)$
0.03	0.00313	0.00343
0.04	0.00594	0.01200
0.05	0.01594	0.03200
0.06	0.03594	0.06800
0.07	0.06656	0.13200
0.08	0.11313	0.22114
0.09	0.18875	0.32571

Table 2.12: Experimental results for packet-error probability

p	$BER_s(72, 288)$	$BER_s(144, 576)$
0.03	0.00136	0.00228
0.04	0.00343	0.00750
0.05	0.00928	0.01746
0.06	0.02239	0.04121
0.07	0.04065	0.08373
0.08	0.06485	0.14621
0.09	0.11357	0.21843

Table 2.13: Experimental results for bit-error probability

From the results in Table 2.12 and Table 2.13 we can conclude that the packet-error probability and bit-error probability for the code (144,576) are approximately twice larger than for the code (72,288). In Table 2.14 the probabilities p_{more} of *more-candidate-errors* and probabilities p_{null} of *null-errors* for these two codes are given ($p_{more} + p_{null} = PER$).

	$RCBQ(72, 288)$		$RCBQ(144, 576)$	
p	p_{null}	p_{more}	p_{null}	p_{more}
0.03	0.00219	0.00094	0.00343	0
0.04	0.00531	0.00063	0.01200	0
0.05	0.01469	0.00125	0.03143	0.00057
0.06	0.03594	0	0.06800	0
0.07	0.06500	0.00156	0.13200	0
0.08	0.11250	0.00063	0.22114	0
0.09	0.18688	0.00188	0.32571	0

Table 2.14: Probabilities of *more-candidate-errors* and *null-errors*

From the probabilities presented in Table 2.14, we can see that we have more unsuccessful decoding of type *more-candidate-errors* in the process of decoding for the code (72,288). While, in the experiments for the code (144,576), except for $p = 0.05$, these errors do not occur. The reason for this is the smaller number of redundant zero nibbles for the code (72,288). Actually, we do not have enough redundant zero blocks at the end of the redundant message L. These zero blocks are needed for putting off the incorrect blocks from the sets S in the last iterations of the decoding process. On the other hand, the number of unsuccessful decodings with *null-error*, which is provided in the theoretical probability, is approximately twice smaller for the code with twice shorter codewords.

These differences in the number of unsuccessful decoded messages, especially in the number of *more-candidates-errors*, are greater if we take much shorter codewords, for example, the code (36,144). For these short codes the number of *more-candidate-errors* is larger due to the small number of redundant zero blocks in the patterns of redundancy.

To confirm this statement we also made experiments for the code (36,144) with rate $R = 1/4$. In these experiments we use the pattern: 1100 1100 1000 0000 1100 1100 0000 0000 0000, the same key $k = 0123456789$, value of $B_{max} = 4$ and the same quasigroup, as in the previous experiments. The obtained results for PER, BER, p_{null} and p_{more} for different values of bit-error probability p are presented in Table 2.15 and Table 2.16. Note that for this code, for $p > 0.05$, the equality $p_{more} + p_{null} = PER$ is not valid since in these experiments we obtained a few unsuccessful decodings with *uncorrected-error* (decoding process ends with one candidate in the last iteration, but the decoded message is not correct).

p	$PER_s(36, 144)$	$BER_s(36, 144)$
0.03	0.02057	0.01646
0.04	0.02286	0.01769
0.05	0.02543	0.02082
0.06	0.03471	0.02948
0.07	0.04814	0.04109
0.08	0.08071	0.07189
0.09	0.10757	0.09838

Table 2.15: Experimental results for PER and BER for code (36,144)

p	p_{null}	p_{more}
0.03	0.00086	0.01971
0.04	0.00257	0.02029
0.05	0.00743	0.01800
0.06	0.01814	0.01629
0.07	0.03186	0.01614
0.08	0.06057	0.01914
0.09	0.08971	0.01657

Table 2.16: Probabilities of *more-candidate-errors* and *null-errors* for code (36,144)

Comparing the results for p_{null} and p_{more} given in Table 2.14 and Table 2.16 we can see that for the shortest code (36,144) the probabilities for *null-errors* are the smallest, but the probabilities for *more-candidate-errors* are the largest. Also, we can note that probabilities p_{null} for the code (36,144) are approximately twice smaller than p_{null} obtained for the code (72,288) which follows from the formula for the theoretical packet-error probability. In the experiments with code (36,144), for $p < 0.06$ the number of *more-candidate-errors* is much larger than the number of *null-errors*. For this values of p the values of PER are worse than for the code (72,288) (Table 2.12 and Table 2.15). On the other hand, for the code (36,144) and $p \geqslant 0.06$, the number of the *null-errors* is more than twice larger than the number of *more-candidate-errors*. For these values of p, the values of PER become better for the code (36,144) than for the code (72,288).

Comparing the results for $PER_s(36, 144)$ and $BER_s(36, 144)$ we can conclude that for this code the values of BER are closer to the values of PER than for the other codes. The reason for that is the small number of redundant zeros in the first blocks of the pattern. Therefore, the messages in the last non-empty decoding candidate set have a small common prefix and the bit-error is greater.

Conclusion

In this chapter we investigate the properties and the performances of the random codes based on quasigroups. We consider the influence of all parameters on the decoding process and the number of successful decodings. We give some ideas and conclusions how to choose the code parameters in order to improve the code performances. Also, we define a new method for decreasing the number of *null-errors*.

Chapter 3

Cut-Decoding algorithm

Random codes based on quasigroups have several parameters and in the previous chapter we have investigated the influence of the code parameters on the code performances. From the experiments we conclude that the speed of the decoding process is one of the biggest problem for these codes. Depending on the chosen pattern for redundant zero symbols, the decoding process is slow in some iterations since the number of elements in the sets S is very large. If we distribute the redundant zeros more uniformly, then the number of elements in the sets S will not be very large, but many *more-candidate-errors* will appear. Therefore, in the patterns we need to put more redundant zeros at the end. In fact, finding good equilibrium for placing the redundant zeros is an open problem. Experimentally we discovered several enough satisfactory patterns.

In order to improve the decoding speed, we define a new coding/decoding algorithm called Cut-Decoding algorithm. Since the decoding of RCBQ is actually a list decoding, the speed of the decoding process and the probability of correct decoding depend on the size of the lists with the possible candidates for decoded message. Therefore, in the new Cut-Decoding algorithm that we propose, we make modifications in order to reduce the number of decoding candidates in all iterations of the decoding process. In this algorithm, we use two transformations of the redundant message with different parameters, and the candidates for the decoded message are obtained using the intersection of the corresponding sets S. In such a way the decoding process is 4.5 times faster than the original one for code (72,288). Also, we consider some other modifications of the decoding algorithm for decreasing the packet-error probability (PER) and the bit-error probability (BER).

3.1 Description of coding with Cut-Decoding algorithm

In Cut-Decoding algorithm, instead of using a (N_{block}, N) code with rate R, we use two $(N_{block}, N/2)$ codes with rate $2R$, that encode/decode a same message of N_{block} bits.

First we add redundant zero symbols in the message $M = m_1 m_2 ... m_l$ (in the same way as in the standard coding algorithm) and we produce a redundant message $L = L^{(1)} L^{(2)} ... L^{(s/2)} = L_1 L_2 ... L_{m/2}$ of $N/2$ bits, where $L^{(i)}$ are sub-blocks of r symbols from Q and $L_i \in Q$. For coding we apply the encryption algorithm (given in Figure 2.2) twice on the same redundant message L using different parameters (different keys or quasigroups). In this way we obtain the codeword of the message as concatenation of the two codewords of $N/2$ bits.

The coding process is schematically presented in Figure 3.1.

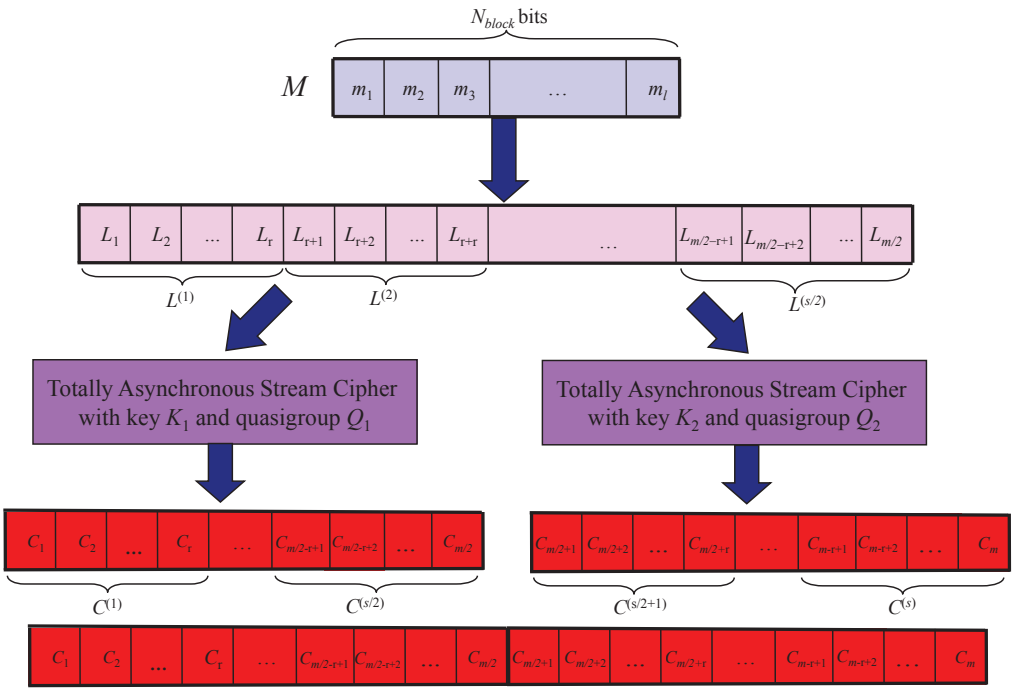


Figure 3.1: Coding with Cut-Decoding algorithm

3.2 Description of decoding with Cut-Decoding algorithm

After transmitting through a binary symmetric channel, we divide the outgoing message $D = D^{(1)}D^{(2)}...D^{(s)}$, where $D^{(i)}$ are sub-blocks of r symbols from the alphabet Q, in 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 lengths and we decode them parallel with the corresponding parameters. In Cut-Decoding algorithm we make modification in the part (iii) of the decoding process, i.e., in the procedure for generating decoding candidate sets. In the new algorithm we generate these sets in the following way.

Step 1. Let $S_0^{(1)} = (k_1^{(1)} \dots k_n^{(1)}; \lambda)$ and $S_0^{(2)} = (k_1^{(2)} \dots k_n^{(2)}; \lambda)$ where λ is the empty sequence, $k_1 = k_1^{(1)} \dots k_n^{(1)}$ and $k_2 = k_1^{(2)} \dots k_n^{(2)}$ are the initials keys used for obtaining the two codewords.

Step 2. Let $S_{i-1}^{(1)}$ and $S_{i-1}^{(2)}$ be defined for $i \geqslant 1$.

Step 3. 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 the standard RCBQ.

Step 4. Let $V_1 = \{w_1 w_2 \dots w_{rai} | (\delta, w_1 w_2 \dots w_{rai}) \in S_i^{(1)}\}$, $V_2 = \{w_1 w_2 \dots w_{rai} \, | (\delta, w_1 w_2 \dots w_{rai}) \in S_i^{(2)}\}$ and $V = V_1 \cap V_2$.

Step 5. 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 $S_i^{(2)}$. *)

Step 6. If $i < s/2$ then increase i and go back to Step 3.

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 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*.

- 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).

In experiments with the new method of decoding, we noticed a significant reduction in the number of elements in the sets S and we achieve big improvement in the speed of the decoding process. Namely, the new method of decoding is 4.5 times faster for code (72,288). In Table 3.1 we give one example of the average number of elements in the sets $S_i^{(1)}$ and $S_i^{(2)}$, in each iteration, before and after the reduction given in **Step 4** and **Step 5** of the new method of decoding. These average numbers are obtained for 1000 decodings. As it is expected, the proposed reduction significantly decreases the number of elements in the decoding candidate sets, that can be seen from Table 3.1 (after the second iteration the number of elements in the reduced sets is approximately 20 times smaller).

The problem in Cut-Decoding algorithm is that for obtaining code with rate R we need a pattern for code with twice larger rate. But, it is hard to make good pattern for larger rates, since the number of redundant zeros in these patterns is smaller. Therefore, with this decoding method we obtain worse results in the number of unsuccessful decodings of type *more-candidate-error*, but the number of unsuccessful decodings with *null-error* is smaller.

To resolve the problem of greater number of *more-candidate-errors* we propose one heuristic in the decoding rule for elimination of this type of errors. Namely, from the experiments with RCBQ we can see that when the decoding process ends with more elements in the reduced decoding candidate sets in the last iteration, almost always the correct message is in these sets (as second part of an element in the both sets). So, 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. If the selected message is the correct one, then the bit-error is 0, so BER will also be reduced. In the experiments we have made with this modification we got that in around half of the cases, the correct message is selected.

No.	No. elements in $S_i^{(1)}$ (before reduction)	No. elements in iteration $S_i^{(2)}$ (before reduction)	No. elements in reduced sets $S_i^{(1)}$, $S_i^{(2)}$
1	10.4	9.9	1.6
2	244.2	244.2	18.2
3	178.5	181.9	8.1
4	79.7	80.5	4.4
5	693.5	695.1	34.8
6	343.9	339.5	14.2
7	140.6	139.3	6.2
8	60.8	61.5	3.4
9	1.1	1.0	1.0

Table 3.1: Average number of elements in the decoding candidate sets before and after reduction

3.3 Comparison of standard and Cut-Decoding algorithm for rate $R = 1/4$

In Section 2.4 we presented the experimental results for standard algorithm of RCBQ for codes (72,288) with rate $R = 1/4$. From the results obtained for different patterns of redundancy, different lengths of the initial key and different quasigroups, we concluded that the best results for these codes we got for the following parameters:

- the pattern of redundancy: 1100 1100 1000 0000 1100 1000 1000 0000 1100 1100 1000 0000 1100 1000 1000 0000 0000 0000,
- the key $k = 0123456789$ of 10 nibbles, and
- the quasigroup given in Table 2.1.

Therefore, we will compare the results obtained by these parameters with suitable results for the new Cut-Decoding algorithm. We have made experiments us-

ing the new algorithm for codes (72,288) over the alphabet of nibbles and transmission through a binary symmetric channel. In the experiments with Cut-Decoding algorithm we have considered 17 different patterns of redundancy for code (72, 144) of rate $R = 1/2$, different lengths of the initial keys and different quasigroups of order 16.

Also, we consider the differences in the performances of the codes when in the two process of coding/decoding, in Cut-decoding algorithm, we use only different keys or different keys and different quasigroups. It is clear that the pattern of redundancy should be the same in both coding processes, due to the reduction of the elements in the decoding candidate sets (introduced in the **Step 4** and **Step 5**). Namely, if different patterns are used then we cannot make intersection of the sets S in **Step 4**. In this section, we will present only the best results.

3.3.1 Experiments with different keys

First, we made experiments using only different keys in the two process of coding/decoding and the same quasigroup. As we mention above, we consider 17 different patterns for adding redundancy. The best results we obtained for the following parameters:

- the pattern of redundancy: 1100 1110 1100 1100 1110 1100 1100 1100 0000,
- two different keys $k_1 = 01234$ and $k_2 = 56789$ of 5 nibbles, and
- the quasigroup given in Table 2.1.

We also tried with keys of length of 10 nibbles but the results were similar.

Let PER_c be the probability of packet-error and BER_c be the probability of bit-error obtained with the new Cut-Decoding algorithm. As previous, PER_s and BER_s are suitable probabilities for standard RCBQ. The results for PER_s and PER_c (BER_s and BER_c), for different values of bit-error probability p of binary symmetric channel and $B_{max} = 4$, are given in Table 3.2 (Table 3.3) and presented in Figure 3.2 (Figure 3.3).

We conclude from the results, given in Table 3.2 and Table 3.3, that we obtain approximately the same results for the values of PER and BER with both decoding algorithms. Moreover, we note that for $p > 0.05$ we have slightly better values of PER with the new Cut-Decoding algorithm.

p	PER_s	PER_c
0.02	0.00125	0.00171
0.03	0.00313	0.00257
0.04	0.00594	0.00514
0.05	0.01594	0.01714
0.06	0.03594	0.02657
0.07	0.06656	0.06171
0.08	0.11313	0.10800
0.09	0.18875	0.15886

Table 3.2: Experimental results for packet-error probability

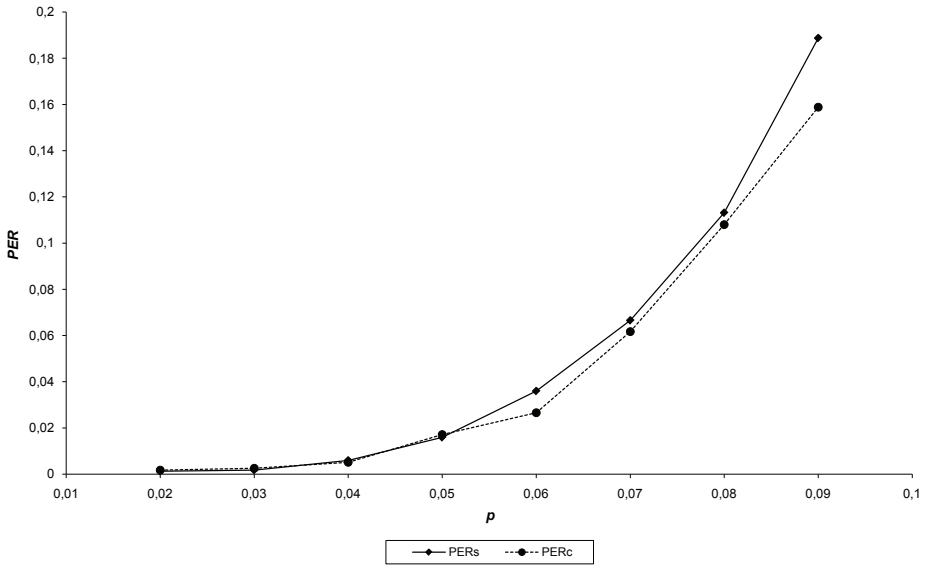


Figure 3.2: Comparison of PER

p	BER_s	BER_c
0.02	0.00076	0.00011
0.03	0.00136	0.00083
0.04	0.00343	0.00302
0.05	0.00928	0.01134
0.06	0.02239	0.01943
0.07	0.04065	0.04243
0.08	0.06485	0.07512
0.09	0.11357	0.11621

Table 3.3: Experimental results for bit-error probability

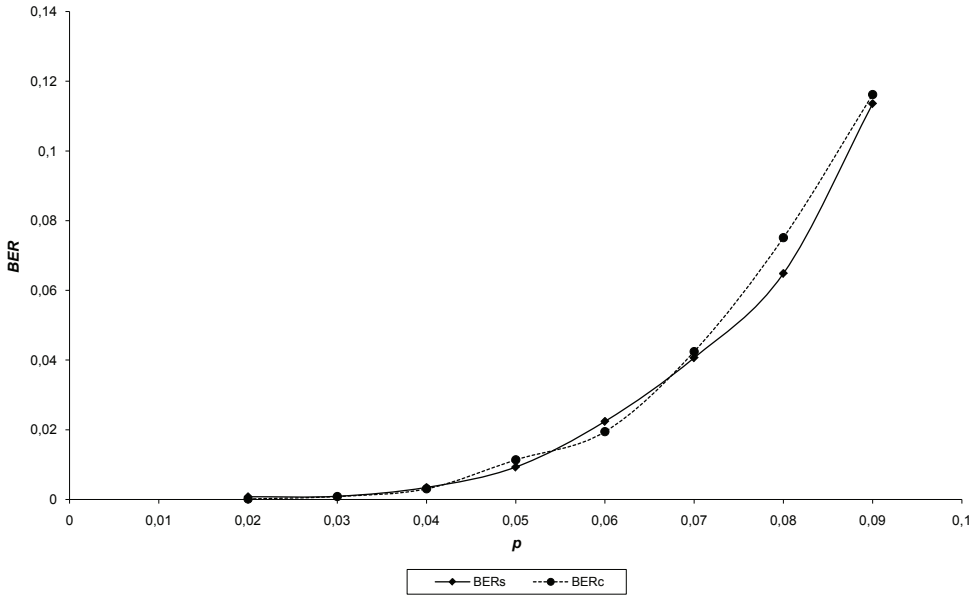


Figure 3.3: Comparison of BER

From the results for BER we can see that the differences between BER_s and BER_c are very small for all values of p. For some p we have slightly better values for BER with the standard algorithm, and for other p - with the new algorithm.

3.3.2 Experiments with different keys and different quasigroups

We have made experiments for code (72,288) with the new method of decoding using two (72,144) codes with different keys and different quasigroups. First, we made experiments using the quasigroup given in Table 2.1 and the quasigroup, with high coefficient of period growth, given at the end of the paper [15]. In these experiments we used the same pattern for rate $R = 1/2$ and $B_{max} = 4$. In this case, we obtained slightly better results using two different keys of 10 nibbles ($k_1 = 0123456789$ and $k_2 = 5432897610$) than using the keys of 5 nibbles. In Table 3.4 we present the results for packet-error probability PER_{c_2} and bit-error probability BER_{c_2} obtained for these parameters.

p	PER_{c_2}	BER_{c_2}
0.02	0.00143	0.00053
0.03	0.00314	0.00150
0.04	0.00571	0.00299
0.05	0.01400	0.01013
0.06	0.03000	0.02183
0.07	0.06314	0.04544
0.08	0.09857	0.07198
0.09	0.16086	0.11839

Table 3.4: Experimental results for packet-error and bit-error probabilities for different quasigroups and different keys

Comparing the results in Table 3.2 and Table 3.3 with the results in Table 3.4 we can conclude that the values of PER_{c_2} and BER_{c_2} are similar with the previous values of PER_c and BER_c obtained using same quasigroup in the both processes. Also, we made experiments using two different quasigroups and same key in the two coding/decoding processes, but we did not obtain better results.

Nevertheless, if we use the cyclic group of order 16 in one of the processes then the results for PER and BER are much worse. For example, for $p = 0.02$ we obtained $PER = 0.18314$ and $BER = 0.05432$.

3.4 Method for reducing *null-errors* in Cut-Decoding algorithm

In Section 2.5 we defined a method for decreasing the number of unsuccessful decodings with *null-error* by backtracking. We showed that in RCBQ with standard algorithms, some of these errors will be eliminated if we cancel few iterations of the decoding process and we reprocess part of them with a larger value of B_{max}.

In this section we apply this idea in the new Cut-Decoding algorithm when a *null-error* appears (i.e. the both reduced sets are empty). We made experiments with returning two or three iterations back and using $B_{max}+1$ or $B_{max}+2$ in the first of canceled iterations (the remain iterations use the previous value of B_{max}). We must note that with this backtracking in some cases we obtain *more-candidate-error* instead of *null-error*.

The best results are obtained with two iterations back and $B_{max}+2$ in the first of the canceled iterations. The percentages of eliminated unsuccessful decodings with *null-error* are given in Table 3.5. These results are obtained by applying this modification on the unsuccessful decodings with *null-error* in the experiments presented in Table 3.2 and Table 3.3. We can conclude that this modification gives good percentage of eliminated *null errors* in Cut-Decoding algorithm.

p	Percentage of eliminated *null-errors*
0.03	28.57%
0.04	20.37%
0.05	24.00%
0.06	23.17%
0.07	27.23%
0.08	25.37%
0.09	22.64%

Table 3.5: Percentage of eliminated unsuccessful decoding with *null-error*

Very important for this modification is that we have only significantly small decreasing of the decoding speed. Also, by eliminating some of the *null-errors* we obtain better results for BER.

In Table 3.6 and Table 3.7 we compare the values PER_c and BER_c from Table 3.2 and Table 3.3, obtained by Cut-Decoding algorithm without backtracking and

values PER_{c_back} and BER_{c_back} obtained by using backtracking (two iterations back and $B_{max} + 2$).

p	PER_c	PER_{c_back}
0.03	0.00257	0.00257
0.04	0.00514	0.00514
0.05	0.01714	0.01429
0.06	0.02657	0.02171
0.07	0.06171	0.04857
0.08	0.10800	0.08200
0.09	0.15886	0.12543

Table 3.6: Experimental results for PER with and without backtracking

p	BER_c	BER_{c_back}
0.03	0.00083	0.00077
0.04	0.00302	0.00165
0.05	0.01134	0.00510
0.06	0.01943	0.00892
0.07	0.04243	0.02194
0.08	0.07512	0.03795
0.09	0.11621	0.05383

Table 3.7: Experimental results for BER with and without backtracking

From the results in Table 3.6 and Table 3.7, we can conclude that for larger values of p we have greater improvement of PER. Also, using this modification we achieve approximately twice better values of BER for all p. This happens since in the cases when the *null-error* will not be eliminated by backtracking, the empty reduced sets can occur in some higher (later) iteration, so the bigger part of the message will be decoded and the value of the bit-error will be smaller.

We also try this modification in the case when only one of the sets $S_i^{(1)}$ or $S_i^{(2)}$ is empty in some iteration. But in these experiments we did not obtain better results than in the previous case when one decoding process continues with the standard algorithm if the other set is empty. In that case, the decoding often ends successfully, especially when the empty set appears in some of the last iterations.

3.5 Method for reducing *more-candidate-errors*

As we mention above, in our initial experiments with Cut-Decoding algorithm we obtained worse results for the number of unsuccessful decodings of type *more-candidate-error*, but the number of unsuccessful decodings with *null-error* was smaller. To resolve the problem of greater number of *more-candidate-errors*, we introduced one heuristic in the decoding rule for elimination of this type of errors. In this heuristic we randomly select one element from the reduced decoding candidate sets in the last iteration and we take its second component as decoded message. Also, from the results in the previous section we can see that using the backtracking method for decreasing the number of *null-errors* in Cut-Decoding algorithm, good improvements of PER and BER are obtained.

In this section, we propose a similar modification with backtracking, in Cut-Decoding algorithm, for decreasing the number of *more-candidate-errors*. If the decoding process ends with *more-candidate-error*, then in order to obtain one candidate for decoded message, we reprocess some iteration with smaller value of B_{max}. Namely, when decoding ends with more elements in the last reduced decoding candidate sets, then we cancel a few of iterations and we reprocess the first of canceled iterations using smaller value of B_{max}. The next iterations use the previous value of B_{max}.

We have made experiments using this modification in Cut-Decoding algorithm for the code (72, 288) with the code parameters that gave the best results for this algorithm, i.e. the pattern: 1100 1110 1100 1100 1110 1100 1100 1100 0000, the keys $k_1 = 01234$, $k_2 = 56789$ and the quasigroup given in Table 2.1. We considered improvements of the probabilities for successful decoding obtained by this method using different number of canceled iterations and using $B_{max} - 1$ or $B_{max} - 2$ in the first canceled iteration.

We obtained the best results, if we cancel the last two iterations and we use $B_{max} - 1$ in the first of the canceled iterations. In Table 3.8 and Table 3.9 we compare the values of PER_c and BER_c (from Table 3.2 and Table 3.3) obtained without backtracking and values of $PER_{c\_back\_more}$ and $BER_{c\_back\_more}$ obtained

using backtracking for *more-candidate-errors*.

p	PER_c	$PER_{c\_back\_more}$
0.02	0.00171	0.00029
0.03	0.00257	0.00086
0.04	0.00514	0.00343
0.05	0.01714	0.01543
0.06	0.02657	0.02600
0.07	0.06171	0.05971
0.08	0.10800	0.10543
0.09	0.15886	0.15743

Table 3.8: Experimental results for PER without and with backtracking for *more-candidate-errors*

p	BER_c	$BER_{c\_back\_more}$
0.02	0.00011	0.00001
0.03	0.00083	0.00051
0.04	0.00302	0.00253
0.05	0.01134	0.01085
0.06	0.01943	0.01932
0.07	0.04243	0.04208
0.08	0.07512	0.07495
0.09	0.11621	0.11583

Table 3.9: Experimental results for BER without and with backtracking for *more-candidate-errors*

From the results in Table 3.8 and Table 3.9 we can conclude that using this modification for reducing the number of *more-candidate-errors* we have improvement of PER and BER, for all values of p.

Improvements for packet-error and bit-error probabilities obtained with the both methods for reducing errors by backtracking, give us an idea to use a combination of these two methods. So, we made experiments using the both methods with backtracking, for *null-errors* and for *more-candidate-errors*. In these experiments, when we obtain *null-error* (i.e., empty sets in some iteration of the decoding process), we cancel two iterations and we reprocess the first of canceled iterations using $B_{max} + 2 = 6$. On the other hand, if the decoding process ends with more elements in the last sets $S_{s/2}^{(1)}$ and $S_{s/2}^{(2)}$, then we go two iterations back and we reprocess the $(s/2 - 1)$-th iteration with $B_{max} - 1 = 3$. In one decoding process we make only one backtracking (for *null-error*), since when we make more than one we obtain too large cardinality of the sets S in some iteration. An exception to this rule is the following case. If after the backtracking for *null-error* we obtain more candidates in the last iteration then we make one more backtracking for *more-candidate-error*.

Let denote by $PER_{c\_back\_2}$ and $BER_{c\_back\_2}$ the packet-error probability and bit-error probability obtained in this case. In Table 3.10 we compare PER_c, PER_{c_back}, $PER_{c\_back\_more}$ and $PER_{c\_back\_2}$, and in Table 3.11 we compare BER_c, BER_{c_back}, $BER_{c\_back\_more}$ and $BER_{c\_back\_2}$.

From the results in Table 3.10 and Table 3.11, we can see that for smaller values of p, we obtain better results with the backtracking for *more-candidate-errors*, due to the fact that for smaller p the larger number of unsuccessful decodings ends with this type of error. On the other hand, for larger values of p we have larger number of *null-errors*. Hence, for these p, better improvements are obtained using the method by backtracking for *null-errors*. Also, we can conclude that with the proposed combination of the methods by backtracking for both types of errors we obtain best improvement for the values of PER and BER for all p. Moreover, the values of $BER_{c\_back\_2}$ are more than twice smaller than BER_c.

3.6 Cut-Decoding algorithm with longer messages

In Section 2.6 we examined the influence of the length of the messages on the performances of error-correcting codes based on quasigroups. There we showed that the packet-error probability and bit-error probability for the code (144, 576), with longer messages and codewords, are approximately twice larger than for the code (72, 288). Also, we have seen that in the experiments for code (72,288) we obtain more unsuccessful decodings of type *more-candidate-error*. But, this type of errors does not occur (except for $p = 0.05$) in the experiments for the code (144,576).

p	PER_c	PER_{c_back}	$PER_{c\_back\_more}$	$PER_{c\_back\_2}$
0.02	0.00171	0.00171	0.00029	0.00029
0.03	0.00257	0.00257	0.00086	0.00029
0.04	0.00514	0.00514	0.00343	0.00314
0.05	0.01714	0.01429	0.01543	0.01200
0.06	0.02657	0.02171	0.02600	0.02000
0.07	0.06171	0.04857	0.05971	0.04486
0.08	0.10800	0.08200	0.10543	0.08114
0.09	0.15886	0.12543	0.15743	0.12486

Table 3.10: Experimental results for PER without and with the methods by backtracking

p	BER_c	BER_{c_back}	$BER_{c\_back\_more}$	$BER_{c\_back\_2}$
0.02	0.00011	0.00011	0.00001	0.00001
0.03	0.00083	0.00077	0.00051	0.00024
0.04	0.00302	0.00165	0.00253	0.00142
0.05	0.01134	0.00510	0.01085	0.00507
0.06	0.01943	0.00892	0.01932	0.00869
0.07	0.04243	0.02194	0.04208	0.02017
0.08	0.07512	0.03795	0.07495	0.03459
0.09	0.11621	0.05383	0.11583	0.05378

Table 3.11: Experimental results for BER without and with the methods by backtracking

In this section, we will examine whether this happens in the experiments with the new Cut-Decoding algorithm.

In Section 3.3 we gave experimental results obtained by Cut-Decoding algorithm for code (72,288) with rate $R = 1/4$. For comparison we made experiments with the same quasigroup, the same keys of 5 nibbles and the same rate for code

(144, 576) with twice longer messages and codewords. We obtained the best results with Cut-Decoding algorithm for a code (144, 576) using the pattern: 1100 1110 1100 1100 1110 1100 1100 1110 1100 1100 1110 1100 1100 1110 1100 1000 0000 0000. In the experiments with this code, unsuccessful decodings of type *more-candidate-error* do not occur. But, as with the standard algorithm, the number of *null-errors* is greater.

p	$PER_c(72, 288)$	$PER_c(144, 576)$
0.02	0.00171	0.00000
0.03	0.00257	0.00114
0.04	0.00514	0.00800
0.05	0.01714	0.02914
0.06	0.02657	0.05657
0.07	0.06171	0.11029
0.08	0.10800	0.17886
0.09	0.15886	0.28286

Table 3.12: Experimental results for packet-error probability

p	$BER_c(72, 288)$	$BER_c(144, 576)$
0.02	0.00011	0.00000
0.03	0.00083	0.00080
0.04	0.00302	0.00621
0.05	0.01134	0.02186
0.06	0.01943	0.04516
0.07	0.04243	0.08534
0.08	0.07512	0.13512
0.09	0.11621	0.22163

Table 3.13: Experimental results for bit-error probability

From the results in Table 3.12 and Table 3.13 we can conclude that, as with the standard algorithm, the packet-error probabilities and bit-error probabilities for the code with longer messages and codewords are larger than for the code with shorter. This is not true only for $p = 0.02$ and $p = 0.03$, since for these values of p the number of *null-errors* is very small. Namely, for $p = 0.02$ and $p = 0.03$, the number of *null-errors* is the same for both codes (for $p = 0.02$ this number is 0). But, for these values of p, in the experiments with the code (72,288), we obtain few unsuccessful decodings of type *more-candidate-error*. For the code (144,576), we obtained that the probability of occurrence of *more-candidate-error* is 0, for all values of p.

3.7 Experiments with quasigroups of order 4 and order 256

In this section we investigate the performances of the random codes based on quasigroups when quasigroups of order 4 and order 256 are used in coding/decoding processes. Then the messages and the codewords are strings of 2-bit letters or 8-bit letters (bytes), correspondingly. We have made several experiments with the standard method of coding/decoding and with Cut-Decoding algorithm defined above.

In Section 2.4 we investigated the performances for code (72,288) with the standard algorithm, for a binary symmetric channel. There we made experiments using alphabet $Q = \{0, 1, \ldots 9, a, b, c, d, e, f\}$ of nibbles and different quasigroups of order 16. We obtained the best results for the quasigroup given in Table 2.1, key of 10 nibbles and the pattern of redundancy: 1100 1100 1000 0000 1100 1000 1000 0000 1100 1100 1000 0000 1100 1000 1000 0000 0000 0000. In Section 3.3 we compared these results with the best results obtained by Cut-Decoding algorithm with alphabet of nibbles.

Now, we made experiments with both algorithms (standard and Cut-decoding) in which we use quasigroups of order 4 and order 256 instead of quasigroups of order 16. We made simulations for a binary symmetric channel with different patterns for redundancy, different keys, different length of the blocks in the decoding process and several quasigroups of order 4 and order 256. In the experiments we considered the packet-error probability, the bit-error probability and the number of *null-errors* and *more-candidate-errors*.

3.7.1 Experiments with quasigroups of order 4

Using image pattern, Dimitrova and Markovski (see [17]), give a classification of quasigroups of order 4 as fractal and non-fractal quasigroups. In [21], the authors give a classification of these quasigroups as linear and nonlinear by Boolean representation. We investigate the performances of RCBQ with messages of 2-bit symbols using quasigroups from different classes of these classifications. We make experiments using fractal quasigroups; non-fractal and weak non-linear; and non-fractal and pure non-linear quasigroups.

In the experiments with quasigroups of order 4, we obtained the worst results using fractal quasigroups. Namely, if we use a fractal quasigroup in the algorithms for encryption/decryption (given in Figure 2.2), we obtain many unsuccessful decodings with *more-candidate-error*, even for $B_{max} = 3$. On the other hand, in the experiments with non-fractal and weak non-linear quasigroups or non-fractal and pure non-linear quasigroups the values of packet-error probability and bit-error probability are similar, with slightly better results for a non-fractal and weak non-linear quasigroup.

Although the codes with nibbles and the codes with 2-bit symbols are two different codes, in this section we will compare the results obtained for code (72,288) with rate $1/4$ for the both alphabets. For $B_{max} = 3$, we obtained better results in the experiments with 2-bit symbols, compared with experiments with nibbles. But, for larger values of B_{max}, we have a lot of unsuccessful decodings with *more-candidate-error* in all experiments for codes with 2-bit symbols (with standard algorithm and with Cut-Decoding algorithm).

The best results for code (72,288) using alphabet of 2-bit symbols and the standard algorithm were obtained for the following parameters:

- the pattern of redundancy: 111000 111000 110000 000000 111000 110000 110000 000000 110000 110000 000000 000000 111000 111000 110000 000000 111000 110000 110000 000000 110000 000000 000000 000000,

- the key $k = 012301230123213023103210$ of 24 symbols, and

- the quasigroup (3.1) (non-fractal and weak non-linear quasigroup).

$*$	0	1	2	3
0	0	2	1	3
1	1	3	2	0
2	2	0	3	1
3	3	1	0	2

(3.1)

In Table 3.14 and Table 3.15 we compare the best results of PER and BER for the both codes (72,288) (with 2-bit symbols and nibbles) using the standard algorithm and $B_{max} = 3$. There, PER_{s_2} and BER_{s_2} are packet-error and bit-error probabilities for the code with 2-bit symbols, and PER_s and BER_s - for the code with nibbles (from Table 2.10 for $B_{max} = 3$) .

p	PER_{s_2}	PER_s
0.02	0.00171	0.00475
0.03	0.00849	0.01843
0.04	0.02429	0.05559
0.05	0.05429	0.11758
0.06	0.09886	0.21314
0.07	0.15743	0.32971

Table 3.14: Experimental results for packet-error probability for $B_{max} = 3$

From the results in Table 3.14 we can conclude that for $B_{max} = 3$ the values of PER_{s_2} are approximately twice better than the values of PER_s. The same conclusion is true for the values BER_{s_2} and BER_s, given in Table 3.15. The better results for the number of *null-errors* for the code with 2-bit symbols follows from the formula for theoretical packet-error probability given in Theorem 2.1. This theorem provides only this type of errors. But, in the experiments with the codes with 2-bit symbols we obtained a few unsuccessful decodings with *more-candidate-error*, even for $B_{max} = 3$. Therefore, the experimental probabilities PER_{s_2} are greater than the theoretical PER_t given in Theorem 2.1.

p	BER_{s_2}	BER_s
0.02	0.00121	0.00209
0.03	0.00486	0.00849
0.04	0.01491	0.02629
0.05	0.03467	0.05488
0.06	0.05917	0.10655
0.07	0.09941	0.16738

Table 3.15: Experimental results for bit-error probability for $B_{max} = 3$

Also, we made experiments using Cut-Decoding algorithm for code (72,288) with 2-bit symbols. In these experiments we obtain the best results using the following parameters in the code design:

– the pattern of redundancy: 111100 111100 111000 111000 111100 111000 111000 111100 111000 111000 110000 000000,

– two different keys $k_1 = 012301230123213023103210$ and $k_2 = 321023102130012301230123$ of 24 symbols, and

– the quasigroup (3.1).

The obtained results for packet-error and bit-error probabilities for different values of bit-error probability p of binary symmetric channel and $B_{max} = 3$ are presented in Table 3.16.

Comparing the suitable probabilities in Table 3.14 (Table 3.15) and Table 3.16 we can conclude that for the code with 2-bit symbols Cut-Decoding algorithm gives slightly better results for PER and BER than the standard decoding algorithm. Also, for these codes the decoding process with Cut-Decoding algorithm is twice faster than with the standard algorithm.

p	PER_{c_2}	BER_{c_2}
0.02	0.00114	0.00059
0.03	0.00714	0.00354
0.04	0.02057	0.01232
0.05	0.04886	0.02965
0.06	0.09200	0.05493
0.07	0.15000	0.09566

Table 3.16: Experimental results with Cut-Decoding algorithm for $B_{max} = 3$

3.7.2 Experiments with quasigroups of order 256

Further on, we made experiments with alphabet of bytes (8-bit symbols) using different patterns, keys and quasigroups of order 256. In these experiments, with both decoding algorithms (standard and Cut-Decoding) we obtained almost the same values for PER and BER as for codes with alphabet of nibbles.

In Table 3.17, we give an example of it. These results are obtained for codes (72,288) with following parameters:

– in the standard algorithm - the pattern of redundancy: 10 10 10 00 00 10 10 00 00 10 10 00 00 10 10 00 00 00 and the key $k = 0123456789$ of 10 bytes,

– in Cut-Decoding algorithm - the pattern: 11 10 10 10 10 10 10 10 00 and the two different keys $k_1 = 0123456789$ and $k_2 = 5432897610$ of 10 bytes.

In the both experiments we used same quasigroup of order 256. In Table 3.17, we compare packet-error and bit-error probabilities obtained with the standard algorithm and Cut-Decoding algorithm for the codes with 8-bit symbols (bytes) and nibbles .

From the results in Table 3.17 we can conclude that with both decoding algorithms we obtained almost the same values for PER and BER for the both alphabets.

	PER		BER	
algorithm	$nibbles$	$bytes$	$nibbles$	$bytes$
standard	0.1131	0.1100	0.0649	0.0749
Cut − Decoding	0.1080	0.1029	0.0751	0.0838

Table 3.17: Experimental results for packet-error and bit-error probabilities for $p = 0.08$, $B_{max} = 4$

Conclusion

In this chapter we define new coding/decoding algorithm for the random codes based on quasigroups. In such a way we obtained 4.5 times faster decoding process for a rate 1/4. We define a new method for decreasing the number of *more-candidates-errors* and we propose a combination of the both methods for decreasing the number of unsuccessful decodings. With this combination we obtain better values of the probability for successful decoding. Also, we investigate the performances of these codes when quasigroups of order 4 and order 256 are used in coding/decoding process.

Chapter 4

Application of RCBQ for decoding images

In this chapter we investigate performances of random codes based on quasigroups for decoding images transmitted through a binary symmetric channel. For that aim we made experiments using the standard coding/decoding algorithm and Cut-Decoding algorithm. Experimental results are compared with suitable results obtained with Reed-Solomon codes.

4.1 Modifications of the algorithms for their application in decoding images

In Chapter 2 we have defined the standard algorithm for RCBQ, and in Chapter 3 we have proposed new Cut-Decoding algorithm for these codes. In the both version of the decoding rule there are two types of unsuccessful decodings: *null-error* and *more-candidate-error*. When *null-error* appears then the decoding process ends early and we have only a part of the message that is decoded. In the case of *more-candidate-error* we have more than one candidate for the decoded message. For application of RCBQ for coding/decoding images we have to resolve these cases of only-part and non-unique decoded message. So, in the experiments with the both algorithms we use the following solutions for the unsuccessful decodings.

Unsuccessful decoding with *more-candidate-error*. Since, in the experiments with Cut-decoding algorithm we have a problem with greater number of *more-candidate-errors*, we have proposed a heuristic for elimination of these type of errors. Namely, if *more-candidate error* appears then we randomly select a message from the reduced sets in the last iteration and we take it as the decoded message. Now, we apply this heuristic in the experiments by images with the both

algorithms for RCBQ.

Unsuccessful decoding with *null-error*. In the cases when *null-error* appears, i.e., $S_i = \emptyset$ in the standard algorithm (or the two reduced sets $S_i^{(1)}$ and $S_i^{(2)}$ are empty in Cut-Decoding algorithm), we take the strings without redundant symbols from all elements in the set S_{i-1} (or $S_{i-1}^{(1)}$ and $S_{i-1}^{(2)}$) 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 we add $l-k$ zero symbols at the end of the message. In the experiments with images we notice that this type of error makes the most visible changes in the decoded images. Therefore, in the experiments with Cut-Decoding algorithm we used the method by backtracking for decreasing the number of unsuccessful decodings with *null-errors* defined in Section 3.4.

4.2 Experiments

We made experiments with a few (color and black-white) images. Here, we present the results obtained by the image given in Figure 4.1. In our experiments we use a binary symmetric channel with the following values of bit-error probability: $p = 0.03$, $p = 0.06$, $p = 0.09$ and $p = 0.12$. We consider only these values of p since for the smaller p there are no visible changes in the decoded images. On the other hand, for $p > 0.12$ coding does not have sense (bit-error probability is bigger than p). Also, we take step 0.03 between the chosen values of p in order to obtain visible differences in the decoded images.

For each value of p, we consider the image after transmission through the channel in the following cases:

a) without using any error-correcting code,

b) using RCBQ with the standard algorithm,

c) using RCBQ with Cut-Decoding algorithm (with backtracking in case of *null-error*),

d) using Reed-Solomon code.

All experiments are made for the code (72,288) with rate $R = 1/4$. In the experiments we use the alphabet of nibbles $Q = \{0, 1, \ldots, 9, a, b, c, d, e, f\}$ and the

Figure 4.1: Original image

quasigroup given in Table 2.1. In the both algorithms for RCBQ we use the patterns, the keys and the lengths of the blocks that gave the best results. Experiments are made using $B_{max} = 4$ in the decoding process.

In our experiments with Reed-Solomon codes (RSC) we used a shortened version of RSC(63,27) ([63]). It is the code RSC(48,12) defined over the Galois field GF(2^6) with primitive polynomial $p(x) = 1 + X + X^6$ (and it has the same good properties as general RSC). This shortened RSC has the same length of the codewords (288bits) and the same rate (1/4) as the considered RCBQ.

4.2.1 Experimental results for PER and BER

Using Matlab, we converted the image given in Figure 4.1, first in matrix of bytes, then in sequence of nibbles. In that way, we obtained a list of 16756 messages of 18 nibbles (72 bits). In this subsection we will consider the obtained values of packet-error and bit-error probabilities, and in the next subsection we will present the obtained images.

First we transmit these messages through the binary symmetric channel without using any error-correcting code. Since the errors in the channel appear randomly and independently, each bit has equal chance to be transmitted incorrectly. Therefore, in this case in almost all messages there is at least one incorrect symbol

and the values of $PER = 1 - (1 - p)^{72}$ are not less than 0.9 for all values of bit-error probability p in the binary symmetric channel.

Further on, we made experiments for the same values of p and the same image, but using RCBQ (with the standard algorithm and with Cut-Decoding algorithm with backtracking in case of *null-error*) and Reed-Solomon code as error-correcting codes. In these experiments, first we encode the messages with the corresponding coding algorithm. After transmission through the binary symmetric channel we decode the outgoing messages using the corresponding decoding algorithm (standard, Cut-Decoding or Reed-Solomon). In this way, some of the errors (occurred during transmission) are successfully corrected. Experimental results for packet-error probabilities (PER) and bit-error probabilities (BER) for different values of bit-error probability p are given in Table 4.1 and Table 4.2. In these tables PER_s and BER_s are the probabilities obtained using the standard algorithm, PER_{c_back} and BER_{c_back} using Cut-Decoding algorithm with backtracking and PER_{rs} and BER_{rs} the probabilities obtained with Reed-Solomon code.

p	PER_s	PER_{c_back}	PER_{rs}
0.03	0.0018	0.0032	0.0003
0.06	0.0338	0.0266	0.1157
0.09	0.1813	0.1282	0.7116
0.12	0.4719	0.3620	0.9748

Table 4.1: Experimental results for packet-error probability

p	BER_s	BER_{c_back}	BER_{rs}
0.03	0.0011	0.0010	0.0001
0.06	0.0251	0.0112	0.0241
0.09	0.1379	0.0570	0.1625
0.12	0.3715	0.1742	0.2565

Table 4.2: Experimental results for bit-error probability

From the results given in Table 4.1 we can see that PER obtains the smallest value for Reed-Solomon code only for $p = 0.03$. While, for $p \geqslant 0.06$, RCBQs give much better results, especially with our Cut-Decoding algorithm.

From Table 4.2, we can see that the differences between the results for BER_s and BER_{rs} are not so significant. The reason for that lies in the construction of RCBQ. Namely, in these codes, when a bit is incorrectly decoded, then almost all consecutive bits are incorrectly decoded. Consequently, we obtain empty decoding candidate set in some iteration and only a part of the message is decoded. Also, we can conclude that the differences between the results for BER_s and BER_{c_back} are more significant. Namely, for $p \geqslant 0.06$, BER_{c_back} is approximately twice smaller than BER_s. This happens since in the experiments with RCBQ, the bit-error is the largest when the decoding ends with *null-error*. In our experiments with Cut-Decoding algorithm we use the proposed method by backtracking for decreasing this type of errors. As we explain previously, with this method we decrease BER even when the *null-error* will not be eliminated, since the bigger part of the message can be decoded with the backtracking. Notice that these results and conclusions are similar with the suitable results (given in the previous chapters) when we encode/decode messages instead of images.

Also, we can notice that for RCBQ with the standard algorithm and Reed-Solomon code, coding/decoding does not have sense for $p \geqslant 0.09$ since $BER > p$. But, for $p = 0.09$, RCBQ with Cut-Decoding algorithm still give $BER_{c\_back} < p$. For $p = 0.12$, we obtain worse BER than p with all considered algorithms.

4.2.2 Visual illustration of the experiments

Here, we will give visual illustration of the results, i.e., all transformation of the image given in Figure 4.1 obtained in our experiments.

Images obtained for the considered values of bit-error probability p in the binary-symmetric channel are presented in Figures 4.2 - 4.5. In these figures the images in a) are obtained after transmission through the channel without using any error-correcting code. In b) and c) we give the images obtained using RCBQ with the standard and Cut-Decoding algorithm, respectively. And in d) the images coded/decoded with Reed-Solomon code are given.

a) b)

c) d)

Figure 4.2: Images for $p = 0.03$

a) b)

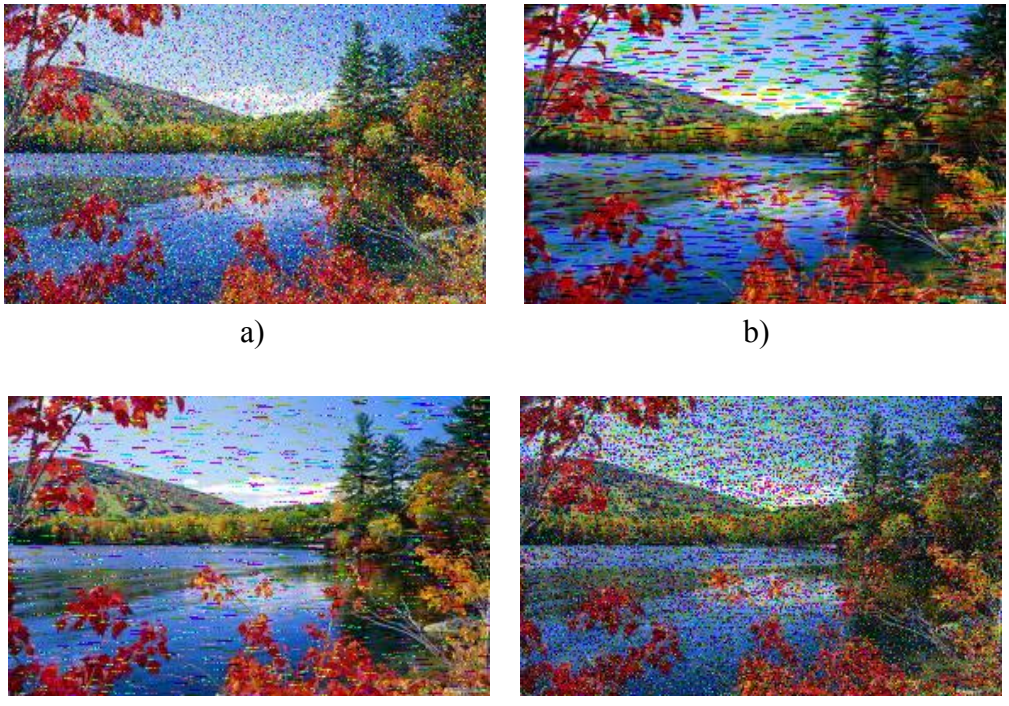
c) d)

Figure 4.3: Images for $p = 0.06$

The images given in Figures 4.2 - 4.5 visually present our conclusions for the values of packet-error probability given above. Namely, for $p = 0.03$ the images obtained using the both algorithms for RCBQ (Figure 4.2 b) and Figure 4.2 c)) are similar. But, for $p \geqslant 0.06$ the images obtained using RCBQ with our Cut-Decoding algorithm are cleaner than the images obtained using the standard RCBQ and Reed-Solomon code.

From the figures, we can see that for all considered values of p (except for $p = 0.03$ and $p = 0.12$) the images obtained using RCBQ with Cut-Decoding algorithm are cleanest.

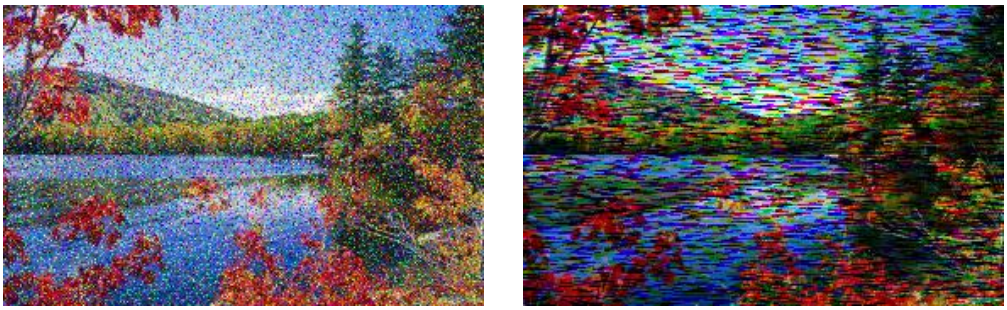
Also, as we note above, for $p = 0.12$ coding/decoding of the image does not have sense, since $BER > p$ and therefore the image in Figure 4.5 a) is more visible than the other images in the same figure.



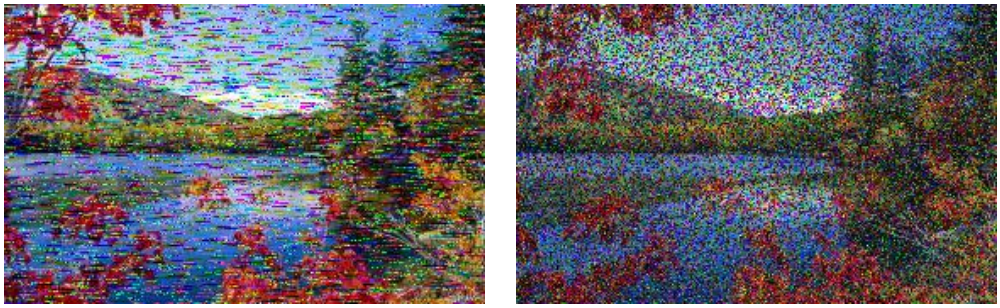
a) b)

c) d)

Figure 4.4: Images for $p = 0.09$



a) b)



c) d)

Figure 4.5: Images for $p = 0.12$

From the previous discussion we can conclude that RCBQs with Cut-Decoding algorithm (with backtracking) have better performances than the other considered algorithms for decoding images transmitted through a binary symmetrical channel.

4.2.3 Patterns of the non-decoded part of messages

As we explain above, in the experiments with the both algorithms of RCBQ we put zero symbols in the place of the non-decoded part of the message when the decoding process ends with *null-error*. In fact, these zero symbols are the horizontal lines that can be seen in Figures 4.2 b) – 4.5 b) for standard RCBQ and Figures 4.2 c) – 4.5 c) for Cut-Decoding algorithm. On the other hand, the images obtained without using error-correcting codes (Figures 4.2 a) – 4.5 a)) do not have these lines, but the entire images have points that are incorrectly transmitted symbols.

Using the fact that these horizontal lines are the most visible changes in the images decoded with RCBQ, we can make the images clearer. Since we know the shape of the changes, analyzing the surrounding pixels we can correct the un-

successful decoded pixels. Using these ideas we can define a filter for correcting images decoded with RCBQ and we can obtain much clearer images. This is one of the open problems for further research.

Conclusion

In this chapter we make modifications of the decoding algorithms for random codes based on quasigroups for their application in decoding images. We investigate performances of these codes for coding/decoding images transmitted through a binary symmetric channel. Also, we compare these results with suitable results obtained with Reed-Solomon codes. From the obtained results we concluded that RCBQs with Cut-Decoding algorithm (with backtracking) have the best performances for decoding images transmitted through a binary symmetric channel. Also, we give ideas for defining filters for further correction of the decoded images.

Chapter 5

4-Set-Cut-Decoding algorithms

As we mentioned before, the decoding speed is the biggest problem for random codes based on quasigroups defined in [30]. In Chapter 3 we have defined new Cut-Decoding algorithm in order to improve the decoding speed of RCBQ. The modified decoding process is 4.5 times faster than the original one for code (72,288). This improvement of the decoding speed gave us an idea for using intersections of more sets S_i in order to obtain greater increasing of the decoding speed. In Cut-Decoding algorithm we use two transformations of the redundant message with different parameters, and the candidates for the decoded messages are obtained by using intersection of the corresponding decoding candidate sets. Now, we make a modifications of that algorithm where we use four transformations of the redundant message. With this new algorithm, called 4-Sets-Cut-Decoding algorithm we obtain greater improvement of the decoding speed. Also, for improving the packet-error and bit-error probabilities we have defined several methods for generating reduced decoding candidate sets.

In this chapter we propose the new 4-Sets-Cut-Decoding algorithms and we analyze the performances of different decoding algorithms of RCBQ (the standard, Cut-Decoding and 4-Sets-Cut-Decoding algorithms) for a code with rate $R = 1/8$. Also we consider the application of the methods for reducing the number of unsuccessful decodings in the new proposed algorithms.

5.1 Coding with 4-Sets-Cut-Decoding algorithms

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.

So, in the process of coding we apply the encryption algorithm, given in Figure 2.2, on the same redundant message L four times using different parameters (different keys or quasigroups) and we obtain the codeword of the message as concatenation of the four codewords of $N/4$ bits.

5.2 First version of 4-Sets-Cut-Decoding algorithm (4-Sets-Cut-Decoding algorithm#1)

After transmitting through a binary symmetric 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 lengths and we decode them parallelly with the corresponding parameters. Similarly, as in Cut-Decoding algorithm (with two sets), in each iteration of the decoding process we reduce the decoding candidate sets obtained in the four decoding processes. In this new modification of Cut-Decoding algorithm named 4-Sets-Cut-Decoding algorithm, we generate decoding candidate sets in the following way.

Step 1. Let $S_0^{(1)} = (k_1^{(1)} \dots k_n^{(1)}; \lambda)$, ..., $S_0^{(4)} = (k_1^{(4)} \dots k_n^{(4)}; \lambda)$, where λ is the empty sequence, and $k_1 = k_1^{(1)} \dots k_n^{(1)}$, ..., $k_4 = k_1^{(4)} \dots k_n^{(4)}$ are the initials keys used for obtaining the four codewords, respectively.

Step 2. Let $S_{i-1}^{(1)}$, ..., $S_{i-1}^{(4)}$ be defined for $i \geqslant 1$.

Step 3. Let four decoding candidate sets $S_i^{(1)}$, ..., $S_i^{(4)}$ be obtained in the four decoding processes, in the same way as in the standard RCBQ.

Step 4. Let $V_1 = \{w_1w_2 \dots w_{r\cdot a\cdot i} | (\delta, w_1w_2 \dots w_{r\cdot a\cdot i}) \in S_i^{(1)}\}$, ...,
$V_4 = \{w_1w_2 \dots w_{r\cdot a\cdot i} | (\delta, w_1w_2 \dots w_{r\cdot a\cdot i}) \in S_i^{(4)}\}$ and $V = V_1 \cap V_2 \cap V_3 \cap V_4$.

Step 5. For each $j = 1, 2, 3, 4$ and for each $(\delta, w_1w_2 \dots w_{r\cdot a\cdot i}) \in S_i^{(j)}$, if $w_1w_2 \dots w_{r\cdot a\cdot i} \notin V$ then $S_i^{(j)} \leftarrow S_i^{(j)} \setminus \{(\delta, w_1w_2 \dots w_{r\cdot a\cdot i})\}$.
(* Actually, for each $j = 1, 2, 3, 4$, we eliminate from $S_i^{(j)}$ all elements whose second part does not match with the second part of an element in all other three sets. In the next iteration the four processes use the corresponding reduced sets $S_i^{(1)}$, $S_i^{(2)}$, $S_i^{(3)}$, $S_i^{(4)}$*).

Step 6. If $i < s/4$ then increase i and go back to Step 3.

The decoding rule in the new 4-Sets-Cut-Decoding algorithm is defined in the following way.

- 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 $w_1 \dots w_{r\cdot a\cdot s/4}$, then $L = w_1 \dots\ w_{r\cdot a\cdot s/4}$ is the decoded redundant message. In this case, we say that we have a *successful decoding*.

- If the reduced sets $S_{s/4}^{(1)}$,..., $S_{s/4}^{(4)}$ have more than one element, after the last iteration, then we have *more-candidate-error*. In this case we apply the same heuristic as in Cut-Decoding algorithm (we randomly select a message from the reduced sets in the last iteration).

- If we obtain only one empty decoding candidate set (or two empty sets) then the decoding continues with the three (or two) nonempty sets (the set V in Step 4 is an intersection of the non-empty sets only).

- If we obtain only one nonempty set, in some iteration, then the decoding continues with the nonempty set using the standard decoding algorithm of RCBQ.

- If we obtain $S_i^{(1)} = S_i^{(2)} = S_i^{(3)} = S_i^{(4)} = \emptyset$ in some iteration, then the process will be stopped (*null-error* appears).

5.3 Second version of 4-Sets-Cut-Decoding algorithm (4-Sets-Cut-Decoding algorithm#2)

In the experiments with 4-Sets-Cut-Decoding algorithm#1 we have seen that when the decoding process ends with *null-error*, i.e., when all four reduced sets are empty, very often the correct message is in three of four non-reduced sets. Therefore, in the second version of 4-Sets-Cut-Decoding algorithm (4-Sets-Cut-Decoding algorithm#2) we make the following modification in Step 4 of the procedure for generating decoding candidate sets.

Step 4$^{\#2}$ Let $V_1 = \{w_1w_2 \dots w_{r\cdot a\cdot i} | (\delta, w_1w_2 \dots w_{r\cdot a\cdot i}) \in S_i^{(1)}\}, \dots,$
$V_4 = \{w_1w_2 \dots w_{r\cdot a\cdot i} | (\delta, w_1w_2 \dots w_{r\cdot a\cdot i}) \in S_i^{(4)}\}$ and $V = V_1 \cap V_2 \cap V_3 \cap V_4$.

1. If $V = \emptyset$ then $V' = V_1 \cap V_2 \cap V_3$ and $V = V'$.

2. If $V' = \emptyset$ then $V'' = V_1 \cap V_2 \cap V_4$ and $V = V''$.

3. If $V'' = \emptyset$ then $V''' = V_1 \cap V_3 \cap V_4$ and $V = V'''$.

4. If $V''' = \emptyset$ then $V^{iv} = V_2 \cap V_3 \cap V_4$ and $V = V^{iv}$.

In fact, in this modification if the intersection of the four sets V_1, V_2, V_3, V_4 is empty then we try to find a nonempty intersection of three sets.

In this way we obtain a great improvement of the packet-error and bit-error probabilities without decreasing the decoding speed. But, for greater improvement of the performances we consider two more modifications of Step 4 when the intersection of all four sets is empty set.

5.4 Third version of 4-Sets-Cut-Decoding algorithm (4-Sets-Cut-Decoding algorithm#3)

In the experiments with 4-Sets-Cut-Decoding algorithm#2, we obtain better results for PER and BER for all values of bit-error probability p of a binary symmetric channel. But, analyzing the experiments with *null-error* obtained with this version, we notice the following situation. Namely, in some experiments we obtained $V' \neq \emptyset$, but the correct message is not in V' and it is in V'' or V''' or V^{iv}. Similarly, if ($V' = \emptyset$ and $V'' \neq \emptyset$) or ($V' = \emptyset$, $V'' = \emptyset$ and $V''' \neq \emptyset$) and the correct message is in some of the next intersections (which are not considered if a previous intersection is not empty).

Therefore, we consider another modification of Step 4. Namely, if the intersection of all four sets is empty then the set $V = V' \cup V'' \cup V''' \cup V^{iv}$, i.e., the new modification of Step 4 is the following.

Step 4#3 Let $V_1 = \{w_1 w_2 \dots w_{r \cdot a \cdot i} | (\delta, w_1 w_2 \dots w_{r \cdot a \cdot i}) \in S_i^{(1)}\}, \dots,$
$V_4 = \{w_1 w_2 \dots w_{r \cdot a \cdot i} | (\delta, w_1 w_2 \dots w_{r \cdot a \cdot i}) \in S_i^{(4)}\}$ and $V = V_1 \cap V_2 \cap V_3 \cap V_4$.

If $V = \emptyset$ then $V = (V_1 \cap V_2 \cap V_3) \cup (V_1 \cap V_2 \cap V_4) \cup (V_1 \cap V_3 \cap V_4) \cup (V_2 \cap V_3 \cap V_4)$.

With this modification, if new $V \neq \emptyset$ we obtain a better improvement of the probabilities for packet-error and bit-error than in 4-Set-Cut-Decoding algorithm#2. Also, this modification does not decrease the speed of the decoding.

5.5 Fourth version of 4-Sets-Cut-Decoding algorithm (4-Sets-Cut-Decoding algorithm#4)

In the third version of 4-Sets-Cut-Decoding algorithm, if new $V = \emptyset$ then we have unsuccessful decoding with *null-error*. In order, to reduce these errors, we make a new modification in the algorithm. Actually, if after Step $4^{\#3}$ we obtain again empty set V then we set V to be a union of all intersections of two sets, i.e., the new Step 4 is the following.

Step 4$^{\#4}$ Let $V_1 = \{w_1 w_2 \dots w_{r \cdot a \cdot i} | (\delta, w_1 w_2 \dots w_{r \cdot a \cdot i}) \in S_i^{(1)}\}, \dots,$
$V_4 = \{w_1 w_2 \dots w_{r \cdot a \cdot i} | (\delta, w_1 w_2 \dots w_{r \cdot a \cdot i}) \in S_i^{(4)}\}$ and $V = V_1 \cap V_2 \cap V_3 \cap V_4$.
If $V = \emptyset$ then $V = (V_1 \cap V_2 \cap V_3) \cup (V_1 \cap V_2 \cap V_4) \cup (V_1 \cap V_3 \cap V_4) \cup (V_2 \cap V_3 \cap V_4)$.
If $V = \emptyset$ then
$V = (V_1 \cap V_2) \cup (V_1 \cap V_3) \cup (V_1 \cap V_4) \cup (V_2 \cap V_3) \cup (V_2 \cap V_4) \cup (V_3 \cap V_4)$.

With this new modification we obtain better results only for $B_{max} = 4$. When $B_{max} = 5$ we obtain a good percentage of eliminated *null-errors* but a larger number of *more-candidate-errors*.

5.6 Comparison of the algorithms for rate $R = 1/8$

In this section we give the experimental results for the probabilities for packet-error (PER) and bit-error (BER) obtained using the new 4-Sets-Cut-Decoding algorithms and we compare them with the results obtained by the standard decoding algorithm and Cut-Decoding algorithm. Also, we analyze the improvement of the performances of the code with all modifications of Step 4, proposed above. In this section we also compare the decoding speeds of all algorithms.

In the previous chapters of this thesis we have given experimental results for the code (72,288) with rate $1/4$ obtained by the standard and Cut-Decoding algorithm. For obtaining a code (N_{block}, N) with rate R in the proposed 4-Sets-Cut-Decoding algorithms, we use four $(N_{block}, N/4)$ codes with rate $4R$. Therefore, in these algorithms for a code with rate $1/4$ we do not have redundancy (if $R = 1/4$ then $N/4 = N_{block}$, i.e., the length of the codeword is equal to the length of the

message). So, for comparison we made experiments for code (72,576) with rate $R = 1/8$.

In all experiments we used the alphabet $Q = \{0, 1, \ldots 9, a, b, c, d, e, f\}$ of nibbles, the quasigroup operations $*$ and $\backslash$ on Q given in Table 2.1 and Table 2.2 and blocks of 4 nibbles in the decoding process.

We obtained the best results for code a (72,576) with the standard decoding algorithm for the following parameters:

- the pattern of redundancy: 1100 1000 0000 0000 0000 0000 1100 1000 0000 0000 0000 0000 1100 1000 0000 0000 0000 0000 1100 1000 0000 0000 0000 0000 1100 1000 0000 0000 0000 0000 1100 1000 0000 0000 0000 0000, and
- the key $k = 0123456789$ of 10 nibbles.

With Cut-Decoding algorithm, we obtained the best results with the following parameters:

- the pattern of redundancy: 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 $k_1 = 0123456789$ and $k_2 = 5432897610$ of 10 nibbles.

In the experiments with the new 4-Sets-Cut-Decoding algorithms, the best results are obtained for the following parameters:

- the pattern of redundancy: 1100 1110 1100 1100 1110 1100 1100 1100 0000 for rate $1/2$, and
- four different keys $k_1 = 0123456789$, $k_2 = 5678901234$, $k_3 = abcdef0123$ and $k_4 = f0123abcde$ of 10 nibbles.

Experimental results for packet-error probabilities for $B_{max} = 4$ and different values of bit-error probability p of a binary symmetric channel are given in Table 5.1 and presented in Figure 5.1. In this table PER_s are the packet-error probabilities obtained by the standard algorithm, and PER_c by Cut-Decoding algorithm. There, we denote by $PER_{c4.1}$, $PER_{c4.2}$, $PER_{c4.3}$ and $PER_{c4.4}$ the packet-error probabilities obtained using 4-Sets-Cut-Decoding#1, 4-Sets-Cut-Decoding#2, 4-Sets-Cut-Decoding#3 and 4-Sets-Cut-Decoding#4, correspondingly.

For all considered algorithms we made experiments until we get $BER > p$, since using the codes does not have sense when $BER > p$ (the number of incorrectly decoded bits is greater than the number of incorrectly transmitted bits).

p	PER_s	PER_c	$PER_{c4.1}$	$PER_{c4.2}$	$PER_{c4.3}$	$PER_{c4.4}$
0.02	0.0011	0	0	0	0	0
0.03	0.0029	0.0017	0.0020	0.0006	0	0
0.04	0.0106	0.0074	0.0086	0.0031	0.0006	0.0006
0.05	0.0326	0.0134	0.0254	0.0074	0.0026	0.0026
0.06	0.0663	0.0371	0.0600	0.0106	0.0034	0.0029
0.07	0.1300	0.0643	0.1129	0.0237	0.0094	0.0063
0.08	0.2200	0.1257	0.1906	0.0383	0.0209	0.0091
0.09	/	0.1791	/	0.0791	0.0437	0.0189
0.10	/	/	/	0.1329	0.0906	0.0277
0.11	/	/	/	0.2166	0.1649	0.0534
0.12	/	/	/	0.3140	0.2666	0.0977
0.13	/	/	/	/	/	0.1537
0.14	/	/	/	/	/	0.2429
0.15	/	/	/	/	/	0.3631

Table 5.1: Experimental results for packet-error probabilities for $B_{max} = 4$

From the results obtained for PER, given in Table 5.1, we can derive the following conclusions. Using Cut-Decoding algorithm instead of the standard algorithm we obtain a great improvement of the probabilities for packet-error (for $p > 0.04$, PER_c are approximately twice smaller than PER_s). Also, Cut-Decoding algorithm is more than twice faster than the standard algorithm.

With 4-Sets-Cut-Decoding algorithm#1 we obtain worse results for PER than those with Cut-Decoding algorithm. But this algorithm is more than 3 times faster than the standard algorithm and about 1.4 times faster than Cut-Decoding algorithm.

From the values for $PER_{c4.2}$ in Table 5.1 we can see that with this modification we obtain better results for PER for all values of p compared with the values obtained by the standard and Cut-Decoding algorithm($PER_{c4.2}$ are from 2 to 3.5 times smaller than PER_c). Also, with 4-Sets-Cut-Decoding algorithm#2 we have almost the same decoding speed as with 4-Sets-Cut-Decoding algorithm#1 (more than 3 times faster than the standard algorithm).

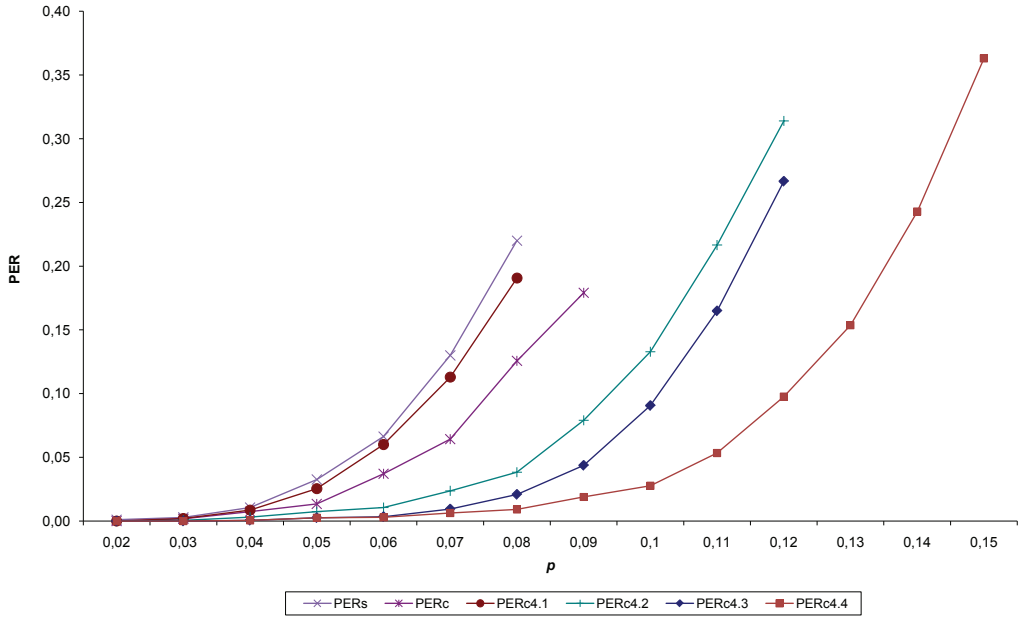


Figure 5.1: Comparison of PER for $B_{max} = 4$

From the results for $PER_{c4.3}$ we can see that with 4-Sets-Cut-Decoding algorithm#3 we have succeeded to eliminate a large number of *null-errors* (for $p \geqslant 0.04$, $PER_{c4.3}$ are from 1.2 to 5.2 times smaller than $PER_{c4.2}$), again without decreasing of the decoding speed. And with the last modification, i.e., 4-Sets-Cut-Decoding algorithm#4, for $p > 0.05$ we have obtained better results for packet-error probabilities with almost the same decoding speed. Moreover, for $p > 0.09$, the values of $PER_{c4.4}$ are approximately 3 times smaller than $PER_{c4.3}$.

p	BER_s	BER_c	$BER_{c4.1}$	$BER_{c4.2}$	$BER_{c4.3}$	$BER_{c4.4}$
0.02	0.0007	0	0	0	0	0
0.03	0.0016	0.0011	0.0010	0.0003	0	0
0.04	0.0054	0.0036	0.0050	0.0019	0.0003	0.0003
0.05	0.0162	0.0079	0.0137	0.0043	0.0013	0.0013
0.06	0.0333	0.0198	0.0339	0.0057	0.0019	0.0017
0.07	0.0672	0.0345	0.0639	0.0125	0.0048	0.0033
0.08	0.1323	0.0712	0.1067	0.0196	0.0090	0.0049
0.09	/	0.1051	/	0.0424	0.0206	0.0103
0.10	/	/	/	0.0670	0.0395	0.0136
0.11	/	/	/	0.1072	0.0771	0.0279
0.12	/	/	/	0.1618	0.1214	0.0515
0.13	/	/	/	/	/	0.0828
0.14	/	/	/	/	/	0.1294
0.15	/	/	/	/	/	0.2009

Table 5.2: Experimental results for bit-error probabilities for $B_{max} = 4$

In Table 5.2 and Figure 5.2 we present values for bit-error probability (BER) from the same experiments and we use the same labels, as in Table 5.1, for values obtained using different algorithms.

From the results for BER in Table 5.2 we can derive the same conclusions as for PER (for all algorithms and for all p, BER is approximately $PER/2$).

Also, we made experiments for the same codes using $B_{max} = 5$ in the decoding process. In Table 5.3 (Figure 5.3) and Table 5.4 (Figure 5.4), we present the obtained results for packet-error and bit-error probabilities for different values of bit-error probability p of a binary symmetric channel.

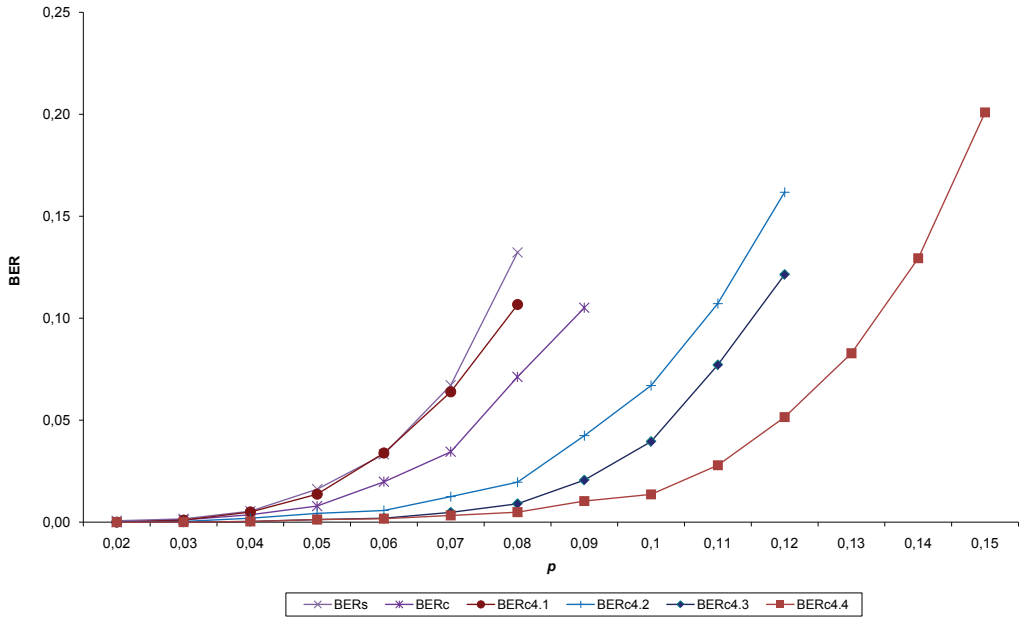


Figure 5.2: Comparison of BER for $B_{max} = 4$

From the values for PER_s and PER_c in Table 5.3 we can conclude that using Cut-Decoding algorithm for this code we obtain better results than with the standard algorithm. On the other hand, in terms of the decoding speed, for $p < 0.06$ the experiments with Cut-Decoding algorithm are 5.2 times faster than with the standard one. But, for $p \geqslant 0.06$, in some experiments with Cut-Decoding algorithm, we obtained a very large cardinality of the decoding candidate sets (after an iteration), so the decoding speed decreases (but it is still better than the speed obtained with the standard algorithm).

Similarly as for $B_{max} = 4$, now for $p > 0.04$, using the new 4-Sets-Cut-Decoding algorithm#1 we obtained worse results for PER and BER than those with Cut-Decoding algorithm. But, from the duration of our experiments, we obtain that this algorithm is 6.3 times faster than the standard algorithm and from 1.2 to 6.2 times faster than Cut-Decoding algorithm.

p	PER_s	PER_c	$PER_{c4.1}$	$PER_{c4.2}$	$PER_{c4.3}$	$PER_{c4.4}$
0.03	0.00571	0.00007	0.00007	0	0	0
0.04	0.00714	0.00086	0.00057	0.00057	0.00029	0.00029
0.05	0.01000	0.00257	0.00286	0.00200	0.00114	0.00086
0.06	0.01086	0.00571	0.00743	0.00371	0.00229	0.00229
0.07	0.02457	0.01429	0.01571	0.00657	0.00286	0.00286
0.08	0.04829	0.03057	0.03171	0.02171	0.01114	0.01114
0.09	0.07171	0.05086	0.05914	0.03629	0.01514	0.01486
0.10	0.12057	0.08800	0.09514	0.05543	0.02429	0.02429
0.11	0.18057	0.13629	0.15114	0.09686	0.04543	0.04514
0.12	/	0.18886	0.22257	0.15029	0.07914	0.07829
0.13	/	0.28948	/	0.21514	0.10943	0.10629
0.14	/	/	/	0.28943	0.17200	0.16029
0.15	/	/	/	/	0.22857	0.22600
0.16	/	/	/	/	0.32314	0.30457
0.17	/	/	/	/	/	0.39143

Table 5.3: Experimental results for packet-error probabilities for $B_{max} = 5$

From the results obtained by 4-Sets-Cut-Decoding algorithm#2, we can see that we obtain an improvement of the values for PER and BER with significantly small decreasing of the decoding speed (maximum difference is 0.84 sec. per message). Also, with 4-Sets-Cut-Decoding algorithm#3, we obtain almost twice better values for PER and BER than with 4-Sets-Cut-Decoding algorithm#2. Again, the speed is almost unchanged.

The results for $PER_{c4.3}$ and $PER_{c4.4}$ are almost identical, i.e., for the both modifications of the algorithm, the total number of unsuccessful decodings is very close. But, the ratios of the number of *null-errors* and *more-candidate-errors* obtained by these two modifications of the algorithm are very different. Namely, with 4-Sets-Cut-Decoding algorithm#3, we obtained only few unsuccessful decodings with *more-candidate-error*. On the other hand, with the proposed modification (for reducing the *null-errors*) in 4-Sets-Cut-Decoding algorithm#4, we eliminated

many of these type of errors, but we obtained a much larger number of *more-candidate-errors* (especially for $p \geqslant 0.08$). Therefore, the results for $BER_{c4.4}$ are smaller than $BER_{c4.3}$. Also, in some experiments for $p \geqslant 0.08$ with 4-Sets-Cut-Decoding algorithm#4 we obtained (after an iteration) a very large cardinality of the decoding candidate sets and in some cases we had to stop the decoding of the message (due to insufficient memory for processing the elements). Therefore, with the last modification of 4-Sets-Cut-Decoding algorithm, the decoding speed continuously decreases (as p increases). For example, for $p = 0.13$, 4-Sets-Cut-Decoding algorithm#4 is 6.4 times slower than 4-Sets-Cut-Decoding algorithm#3.

p	BER_s	BER_c	$BER_{c4.1}$	$BER_{c4.2}$	$BER_{c4.3}$	$BER_{c4.4}$
0.03	0.00117	0.00005	0.00004	0	0	0
0.04	0.00195	0.00047	0.00037	0.00021	0.00011	0.00007
0.05	0.00218	0.00148	0.00179	0.00096	0.00049	0.00025
0.06	0.00509	0.00302	0.00413	0.00251	0.00152	0.00087
0.07	0.01323	0.00831	0.00902	0.00339	0.00178	0.00113
0.08	0.02569	0.01778	0.01909	0.01186	0.00636	0.00378
0.09	0.03876	0.02836	0.03303	0.02115	0.00899	0.00521
0.10	0.07013	0.05131	0.05193	0.03279	0.01505	0.00906
0.11	0.11062	0.07755	0.08812	0.05017	0.02546	0.01472
0.12	/	0.10314	0.13179	0.08590	0.04704	0.02698
0.13	/	0.16065	/	0.12251	0.06365	0.03789
0.14	/	/	/	0.16827	0.10289	0.06789
0.15	/	/	/	/	0.13609	0.10059
0.16	/	/	/	/	0.19017	0.13277
0.17	/	/	/	/	/	0.17911

Table 5.4: Experimental results for bit-error probabilities for $B_{max} = 5$

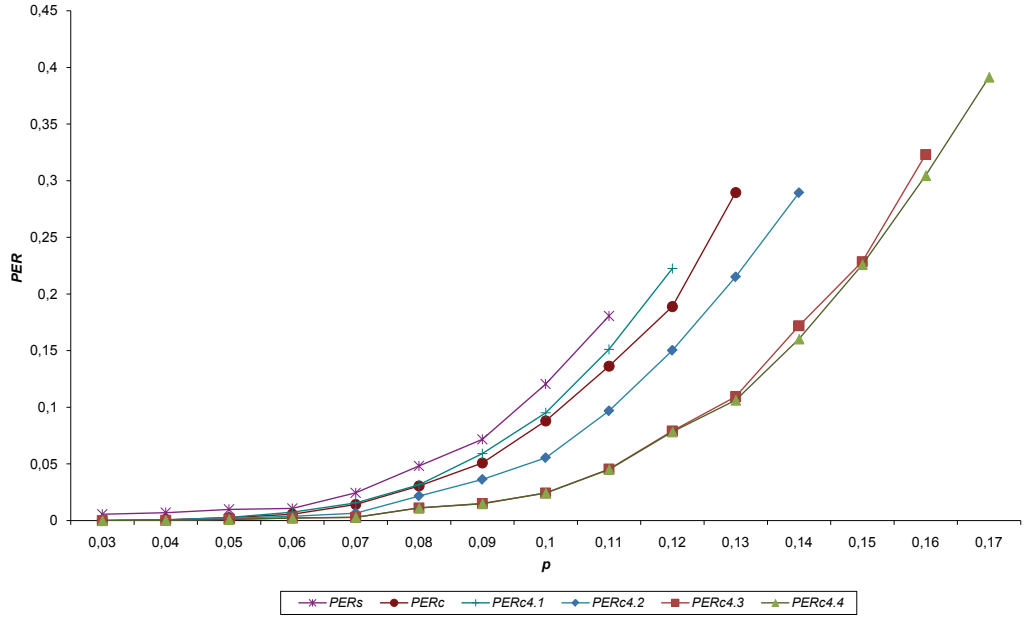


Figure 5.3: Comparison of PER for $B_{max} = 5$

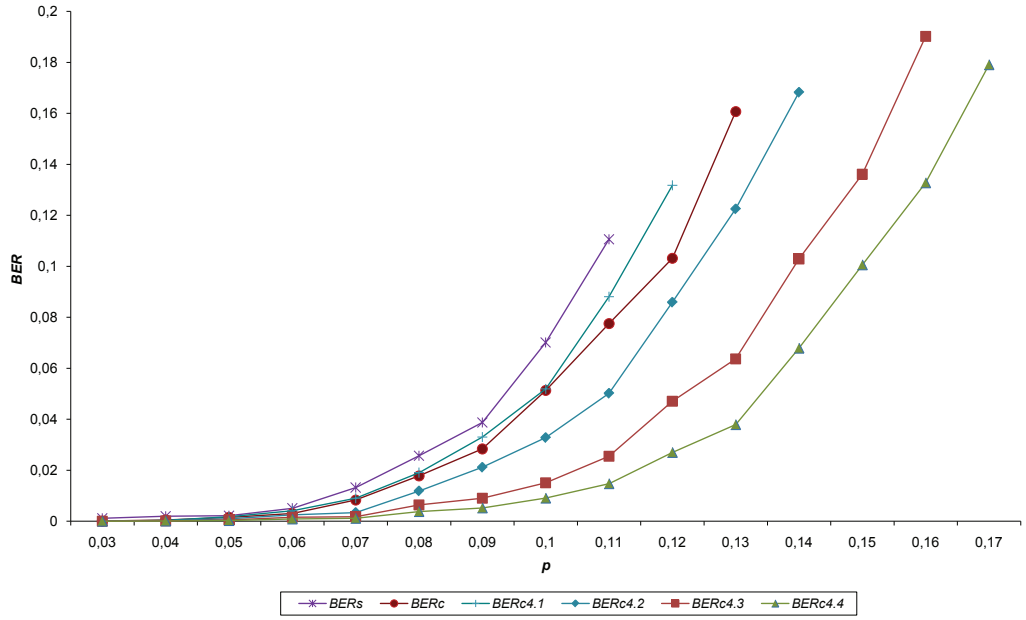


Figure 5.4: Comparison of BER for $B_{max} = 5$

From all presented results obtained in several experiments with different decoding algorithms for a code (72,576) we can conclude that for this code with the new 4-Sets-Cut-Decoding algorithms we obtain a great improvement of the decoding speed and much better values for packet-error and bit-error probabilities.

Also, from the experiments we can conclude that 4-Sets-Cut-Decoding algorithm#4 gives the best values of PER and BER, especially for $B_{max} = 4$. But, for $B_{max} = 5$, the decoding speed of this algorithm is smaller than the decoding speed of 4-Sets-Cut-Decoding algorithm#3 and we obtain a larger number of *more-candidate-errors*.

We can conclude that 4-Sets-Cut-Decoding algorithm#4 is the best for code (72,576) for $B_{max} = 4$. On the other hand, for $B_{max} = 5$, as a compromise between the speed and the correct decoding, 4-Sets-Cut-Decoding algorithm#3 is the best algorithm.

5.7 Experiments with methods for reducing the number of errors

As we mention in Chapter 2, an unsuccessful decoding with *null-error* (in RCBQ) occurs when more than predicted B_{max} bit errors appear during transmission of some blocks. Some of these errors can be eliminated if we cancel a few of iterations of the decoding process and we reprocess all of them or part of them with a larger value of B_{max}. Therefore, in Section 2.5 we have proposed a method for decreasing the number of *null-errors* by backtracking. In this method if we get an empty set in some iteration (for example, i^{th} iteration), then k previous iterations $((i-1)^{th}$, $(i-2)^{th}$, ..., $(i-k)^{th})$ are canceled. After that we reprocess the first of canceled iterations $((i-k)^{th})$ with $B_{max} = B_{max} + 1$ or $B_{max} + 2$, and the next iterations continue with the old value of B_{max}. We obtained a good improvement of the packet-error and bit-error probabilities using this method for reducing the number of *null errors* in the standard algorithm (Section 2.5) and Cut-Decoding algorithm (Section 3.4).

These improvements gave us an idea to use similar method with backtracking in the case of *more-candidate error*, defined in Section 3.5. In this method, if the decoding process ends with more elements in the reduced decoding candidate sets

after the last iteration then we cancel a few of iterations and we reprocess the first of cancelled iterations using smaller value of B_{max} (the next iterations use the previous value of B_{max}).

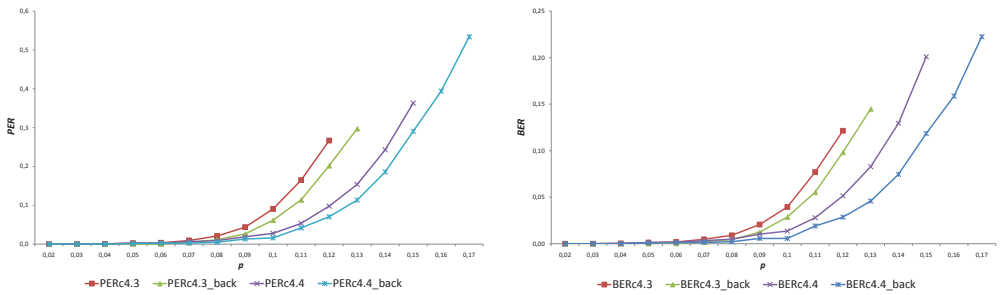
We made experiments with the new 4-Sets-Cut-Decoding algorithms using the following combination of these two methods with backtracking. If we obtain *null-error*, i.e., empty reduced sets in some iteration of the decoding process then we cancel two iterations (or one iteration) and we reprocess the first of canceled iterations using $B_{max} + 2$. If the decoding process ends with more elements in the sets after the last iteration, then we go two iterations back and we reprocess the penultimate iteration with $B_{max} - 1$. Also, if after the backtracking for *null error* we obtain more candidates in the last iteration then we make one more backtracking for *more-candidate-error*. We repeat the experiments, using the above combination of the methods for reducing the number of unsuccessful decodings, with the third and the fourth version of 4-Sets-Cut-Decoding algorithm (by which we obtained the best results).

For $B_{max} = 5$, with the both versions of the algorithm, we obtain best results if we cancel two iterations in case of *null-error*. But, for $B_{max} = 4$ with 4-Sets-Cut-Decoding algorithm#3 we obtain better results if we cancel one iteration in the backtracking for *null-error*.

In Table 5.5, Figure 5.5 a) (for $B_{max} = 4$) and Table 5.7, Figure 5.6 a) (for $B_{max} = 5$) we compare the packet-error probabilities ($PER_{c4.3\_back}$, $PER_{c4.4\_back}$) obtained using the above combination of the both methods with backtracking and the probabilities ($PER_{c4.3}$, $PER_{c4.4}$) obtained by the third and fourth version of 4-Sets-Cut-Decoding algorithm without backtracking (from the tables given in the previous section). Also, in Table 5.6, Figure 5.5 b) ($B_{max} = 4$) and Table 5.8, Figure 5.6 b) ($B_{max} = 5$) we compare the suitable bit -error probabilities ($BER_{c4.3\_back}$, $BER_{c4.4\_back}$, $BER_{c4.3}$, $BER_{c4.4}$) obtained in the same experiments. With the proposed methods by backtracking, we also made experiments until we got $BER > p$.

p	$PER_{c4.3}$	$PER_{c4.3\_back}$	$PER_{c4.4}$	$PER_{c4.4\_back}$
0.02	0	0	0	0
0.03	0	0	0	0
0.04	0.0006	0	0.0006	0.0003
0.05	0.0026	0.0003	0.0026	0.0017
0.06	0.0034	0.0006	0.0029	0.0026
0.07	0.0094	0.0037	0.0063	0.0029
0.08	0.0209	0.0114	0.0091	0.0049
0.09	0.0437	0.0263	0.0189	0.0131
0.10	0.0906	0.0617	0.0277	0.0160
0.11	0.1649	0.1140	0.0534	0.0417
0.12	0.2666	0.2020	0.0977	0.0703
0.13	/	0.2974	0.1537	0.1134
0.14	/	/	0.2429	0.1860
0.15	/	/	0.3631	0.2903
0.16	/	/	/	0.3940
0.17	/	/	/	0.5340

Table 5.5: Experimental results for PER for $B_{max} = 4$ with and without backtracking



a) b)

Figure 5.5: PER and BER without and with backtracking for $B_{max} = 4$

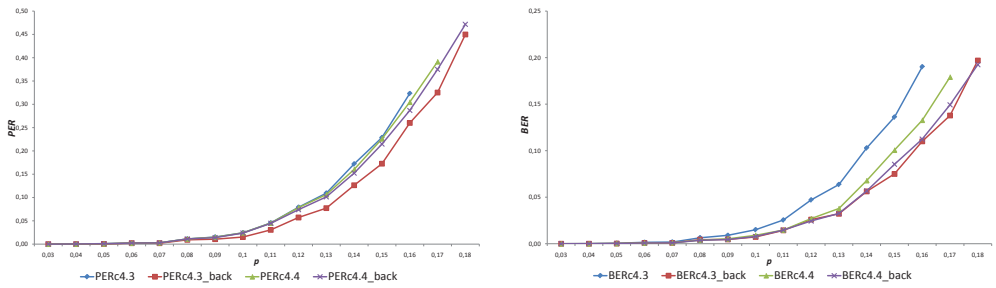
p	$BER_{c4.3}$	$BER_{c4.3\_back}$	$BER_{c4.4}$	$BER_{c4.4\_back}$
0.02	0	0	0	0
0.03	0	0	0	0
0.04	0.0003	0	0.0003	0.0001
0.05	0.0013	0.0002	0.0013	0.0008
0.06	0.0019	0.0003	0.0017	0.0012
0.07	0.0048	0.0020	0.0033	0.0014
0.08	0.0090	0.0044	0.0049	0.0021
0.09	0.0206	0.0124	0.0103	0.0056
0.10	0.0395	0.0287	0.0136	0.0057
0.11	0.0771	0.0554	0.0279	0.0191
0.12	0.1214	0.0983	0.0515	0.0285
0.13	/	0.1449	0.0828	0.0459
0.14	/	/	0.1294	0.0745
0.15	/	/	0.2009	0.1184
0.16	/	/	/	0.1586
0.17	/	/	/	0.2225

Table 5.6: Experimental results for BER for $B_{max} = 4$ with and without backtracking

We conclude from the results in Table 5.5 and Table 5.6 that for $B_{max} = 4$ with the proposed backtracking (in both algorithms) we have improvement of PER and BER for all p. For 4-Sets-Cut-Decoding algorithm#4, the values of BER obtained by backtracking are approximately twice smaller than suitable values without backtracking.

p	$PER_{c4.3}$	$PER_{c4.3\_back}$	$PER_{c4.4}$	$PER_{c4.4\_back}$
0.03	0	0	0	0
0.04	0.00029	0	0.00029	0.00029
0.05	0.00114	0.00029	0.00086	0.00086
0.06	0.00229	0.00200	0.00229	0.00229
0.07	0.00286	0.00200	0.00286	0.00257
0.08	0.01114	0.00914	0.01114	0.01114
0.09	0.01514	0.01057	0.01486	0.01400
0.10	0.02429	0.01514	0.02429	0.02371
0.11	0.04543	0.03057	0.04514	0.04457
0.12	0.07914	0.05714	0.07829	0.07400
0.13	0.10943	0.07714	0.10629	0.10143
0.14	0.17200	0.12629	0.16029	0.15229
0.15	0.22857	0.17257	0.22600	0.21457
0.16	0.32314	0.26000	0.30457	0.28686
0.17	/	0.32514	0.39143	0.37486
0.18	/	0.44971	/	0.47143

Table 5.7: Experimental results for PER for $B_{max} = 5$ with and without backtracking



a) b)

Figure 5.6: PER and BER without and with backtracking for $B_{max} = 5$

p	$BER_{c4.3}$	$BER_{c4.3\_back}$	$BER_{c4.4}$	$BER_{c4.4\_back}$
0.03	0	0	0	0
0.04	0.00011	0	0.00007	0.00007
0.05	0.00049	0.00015	0.00025	0.00025
0.06	0.00152	0.00080	0.00087	0.00080
0.07	0.00178	0.00087	0.00113	0.00097
0.08	0.00636	0.00437	0.00378	0.00361
0.09	0.00899	0.00463	0.00521	0.00453
0.10	0.01505	0.00724	0.00906	0.00778
0.11	0.02546	0.01437	0.01472	0.01465
0.12	0.04704	0.02592	0.02698	0.02438
0.13	0.06365	0.03212	0.03789	0.03256
0.14	0.10289	0.05622	0.06789	0.05675
0.15	0.13609	0.07500	0.10059	0.08532
0.16	0.19017	0.11002	0.13277	0.11241
0.17	/	0.13777	0.17911	0.14929
0.18	/	0.19689	/	0.19246

Table 5.8: Experimental results for BER for $B_{max} = 5$ with and without backtracking

Also, from the results in Table 5.7 and Table 5.8 we can see that for $B_{max} = 5$, with the proposed backtracking, we obtain a greater improvement of PER and BER using 4-Sets-Cut-Decoding algorithm#3. Even more, for all values of p, $PER_{c4.3\_back}$ are smaller than $PER_{c4.4\_back}$, although these probabilities in the experiments without backtracking are almost identical. The reason for that is the increased cardinality of the decoding candidate sets in 4-Sets-Cut-Decoding algorithm#4. Hence, the unsuccessful decodings with *more-candidate-error* end with a great number of elements in the last sets. Therefore, the cardinality of these sets is not decreased to 1 if we use $B_{max} - 1$ in the first canceled iteration. But, if we apply $B_{max} - 2$, then the correct message usually is eliminated from the sets S.

5.8 Visual illustration of the experiments

In this section we investigate performances of the new proposed 4-Sets-Cut-Decoding algorithm#3 (with the above combination of the both methods with backtracking for obtaining the best results) for decoding images transmitted through a binary symmetric channel. For that aim we made experiments with Figure 4.1. In these experiments we use a binary symmetric channel and we code/decode the image using the code (72,576) with rate $R = 1/8$ and the parameters that gave best results for 4-Sets-Cut-Decoding algorithms. Experiments are made using $B_{max} = 5$ in the decoding process and following values of bit-error probability in the channel: $p = 0.05$, $p = 0.08$, $p = 0.11$, $p = 0.14$ and $p = 0.17$ (for $p > 0.17$ the values of $BER_{c4.3\_back}$ are greater than p). In the experiments we use the same solutions for the unsuccessful decodings as in Chapter 4.

Also, for comparison we made experiments with Reed-Solomon codes. In the experiments with Reed-Solomon codes (RSC) we used a shortened version of RSC(127,57) [63]. It is the code RSC(80,10) defined over the Galois field GF(2^7) with primitive polynomial $p(x) = 1 + X + X^7$. This shortened RSC has the same rate 1/8 and approximately the same length of messages and codewords ((70, 560)) as the considered RCBQ.

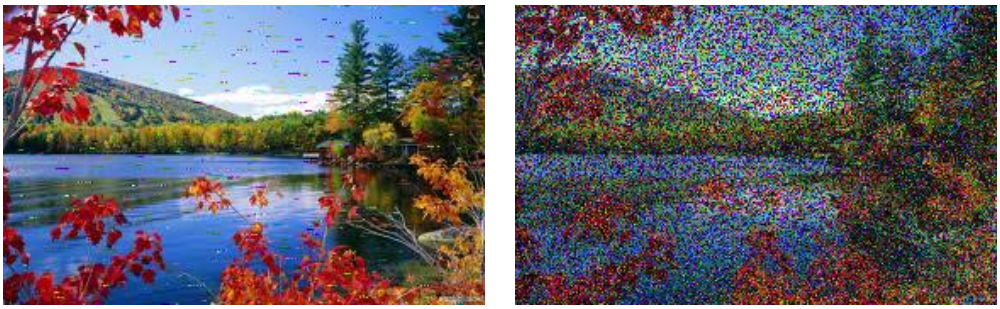
Images obtained for the considered values of bit-error probability p in the binary-symmetric channel are presented in Figures 5.7 - 5.11. In these figures in a) we give the images obtained using RCBQ with 4-Sets-Cut-Decoding algorithm#3 with the proposed combination of the both methods with backtracking. And in b) the images coded/decoded with Reed-Solomon code are given.

a) b)

Figure 5.7: Images for $p = 0.05$

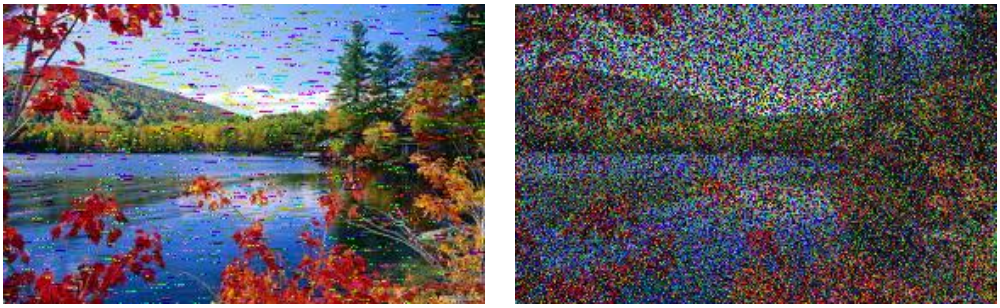
a) b)

Figure 5.8: Images for $p = 0.08$



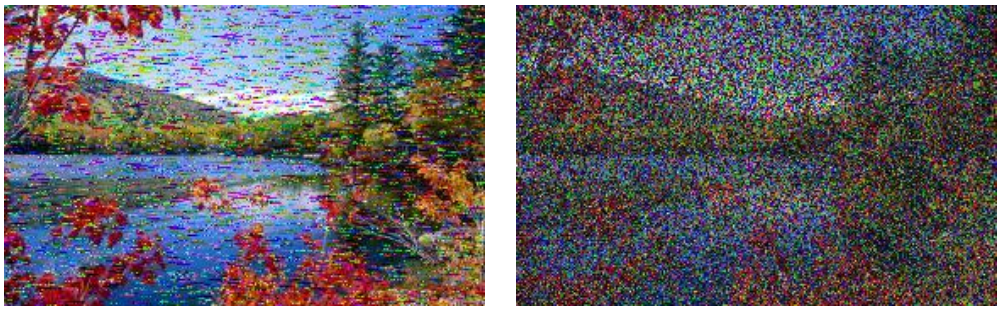
a) b)

Figure 5.9: Images for $p = 0.11$



a) b)

Figure 5.10: Images for $p = 0.14$



a) b)

Figure 5.11: Images for $p = 0.17$

From the figures, we can see that for all considered values of p, the images obtained using RCBQ with 4-Sets-Cut-Decoding algorithm#3 (with backtracking) are cleaner than the suitable images obtained using Reed-Solomon code. Also, for $p \geqslant 0.08$ coding/decoding of the image with Reed-Solomon code does not have sense ($BER > p$). But, RCBQ with 4-Sets-Cut-Decoding algorithm#3 give $BER_{c\_back} < p$ for all $p \leqslant 0.17$ (as in Table 5.8).

Conclusion

In this chapter we propose a new coding/decoding algorithm for random codes based on quasigroups, called 4-Sets-Cut-Decoding algorithm (with 4 different methods for generating reduced decoding candidate sets). With this algorithm the decoding process is 6.3 times faster then the decoding process with the standard RCBQ for a code with rate 1/8. Also, for some values of p we obtain 24 times smaller values for PER then with the standard RCBQ. We analyze the performances of different decoding algorithms of RCBQ (the standard, Cut-Decoding and 4-Sets-Cut-Decoding algorithms) for a code with rate $R = 1/8$. Also we consider the application of the methods for reducing the number of unsuccessful decodings in the new proposed algorithms. At the end of this chapter we investigate performances of RCBQ with 4-Sets-Cut-Decoding algorithm#3 for coding/decoding images transmitted through a binary symmetric channel. Also, we compare these results with suitable results obtained with Reed-Solomon code.

Chapter 6

Some theoretical results for the new algorithms of random codes based on quasigroups

In this chapter we will give some theoretical results for the new algorithms (Cut-Decoding and 4-Sets-Cut-Decoding algorithm) of RCBQ proposed in this thesis.

We will derive a theoretical upper bound for packet-error probabilities (without *more-candidate-errors*) for Cut-Decoding and 4-Sets-Cut-Decoding algorithm. We will show that this bound is equal to theoretical PER_t for the standard algorithm of RCBQ. First we will give a theoretical proof of the upper bound for PER and then we will verify it experimentally.

Also, at the end of this chapter we will give approximate formulas for cardinality of the reduced decoding candidate sets obtained in the decoding process with the new algorithms.

6.1 Theoretical upper bound for packet-error probability in the new algorithms

The formula for theoretical packet-error probability for the standard algorithm is given in Theorem 2.1 ([30]). But, from the proof of this theorem, given in [30], it is clear that in this formula *more-candidate-errors* are not provided. The number of these errors strongly depends on the chosen quasigroup, the pattern for adding redundancy and the length of the initial key. Therefore, it is very difficult to predict the number of *more-candidate-errors* in the experiments with these codes.

In this section we will prove that the formula for PER_t in Theorem 2.1 (for

same code rate, length of the blocks and B_{max} in the decoding process) gives an upper bound for PER obtained by Cut-Decoding and 4-Sets-Cut-Decoding algorithms. From the same reasons as above, in the following theorem and its proof we will consider probability only for *null-errors* and *uncorrected-errors* (decoding that completes successfully, but the decoded message is incorrect).

Theorem 6.1. *The packet-error probability* (*without more-candidate-errors*) *obtained by the standard decoding algorithm of RCBQ is an upper bound for the packet-error probability obtained using Cut-Decoding and 4-Sets-Cut-Decoding algorithms* (*for same code rate, same length of the blocks and same* B_{max} *in the decoding processes*).

Proof. The probability that maximum t bits in one block $D^{(i)}$ of r symbols are not correctly transmitted is

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

where p is probability of bit-error in a binary symmetric channel.

The events A_i : "maximum B_{max} errors appear in a block $D^{(i)}$", for all i are independent and the probability that the correct block $C^{(i)}$ is contained in the set H_i is $P(p; B_{max}) = 1 - q_B$.

Since, for a code with rate R in Cut-Decoding algorithm, two codes with rate $2R$ are used, we have twice smaller number of iterations in the decoding process. So, if the decoding process with the standard algorithm has s iterations for a given code, then for the same code the number of iterations in the two parallel processes of decoding in Cut-Decoding algorithm is $s/2$.

When we use Cut-Decoding algorithm, then the correct block $L^{(i)}$, $i = 1, 2, ..., s/2$ is contained (as second part of an element) in the reduced sets $S_i^{(1)}$ and $S_i^{(2)}$ if in the both processes the correct blocks $C^{(i)}$ and $C^{(s/2+i)}$ (from the two codewords) are contained in the corresponding sets $H_i^{(1)}$ and $H_i^{(2)}$. The probability for that event is $(1 - q_B)(1 - q_B)$ for each i, since the both decoding processes are independent. Therefore the probability each correct block $L^{(i)}$ to be contained in the reduced sets $S_i^{(1)}$ and $S_i^{(2)}$ for $i = 1, 2, ..., s/2$ is $((1 - q_B)(1 - q_B))^{s/2} = (1 - q_B)^s$. So, the probability for packet-error is $1 - (1 - q_B)^s = PER_t$, where PER_t is the theoretical packet-error probability for the same code (same code rate, length of the blocks and B_{max}) with the standard decoding algorithm.

But, according to the decoding rule of Cut-Decoding algorithm, when one of the decoding candidate sets is empty decoding of the message continues using the

standard algorithm. In this case the process will end successfully if the correct block is contained in the non-empty set. Therefore, probability $1-(1-q_B)^s$ (which is probability PER_t in Theorem 2.1) is an upper bound for the packet-error probability (without *more-candidate-errors*) obtained by Cut-Decoding algorithm.

Similarly, we obtained the same upper bound for 4-Sets-Cut-Decoding algorithm. In this algorithm we have four decoding processes and for a code with rate R we use four codes with rate $4R$. Therefore, in 4-Sets-Cut-Decoding algorithm we have four times ($s/4$) smaller number of iterations. In all iterations the correct block $L^{(i)}$, $i=1,2,...,s/4$ is contained (as second part of an element) in the reduced sets $S_i^{(1)}$, $S_i^{(2)}$, $S_i^{(3)}$ and $S_i^{(4)}$ if the correct blocks $C^{(i)}$, $C^{(s/4+i)}$, $C^{(s/2+i)}$ and $C^{(3s/4+i)}$ (from the four codewords) are contained in the corresponding sets $H_i^{(1)}$, $H_i^{(2)}$, $H_i^{(3)}$ and $H_i^{(4)}$. As previous, the upper bound for packet-error probability is $1-((1-q_B)(1-q_B)(1-q_B)(1-q_B))^{s/4}=1-(1-q_B)^s= PER_t$. Again this probability is an upper bound for PER (without *more-candidate-errors*) since the decoding of the message also continues when at least one of the decoding candidate sets is not empty. □

With this theorem we proved that with the new decoding algorithms we have improved not only the decoding speed, but also we decrease the number of unsuccessful decodings (when the upper bound is not reached).

6.2 Experimental verification of the theoretical upper bound for packet-error probability

For experimental verification of Theorem 6.1 proved in the previous section, we will compare the values of PER_t together with the experimental results for PER without *more-candidate-errors* obtained in the experiments with Cut-Decoding and 4-Sets-Cut-Decoding algorithm.

6.2.1 Comparison for rate $R=1/4$

First we will compare PER_t with the experimental results for packet-error probabilities without *more-candidate-errors* PER_{c_1} obtained by Cut-Decoding algorithm and PER_{s_1} by the standard algorithm for the code (72,288) with rate $R=1/4$. These results are for the codes which parameters are given in Section 3.3 and Section 2.4.2, respectively. In these experiments we have used blocks

of 4 nibbles (16 bits) in the process of decoding. So, in the formula for theoretical packet-error probability PER_t, we calculate the probabilities q_B using the following formula:

$$q_B = 1 - P(p; B_{max}) = 1 - \sum_{k=0}^{B_{max}} \binom{16}{k} p^k (1-p)^{16-k}.$$

The number of blocks in the codewords, i.e., the number of iterations in the decoding process is 18.

In Table 6.1, we give the experimental results for PER_{c_1} and PER_{s_1} together with the values of PER_t for different values of bit-error probability p of a binary symmetric channel.

p	PER_t	PER_{c_1}	PER_{s_1}
0.02	0.000209	0	0.000313
0.03	0.001447	0.000857	0.002188
0.04	0.005541	0.003429	0.005313
0.05	0.015319	0.014286	0.014688
0.06	0.034361	0.025714	0.035938
0.07	0.066467	0.057714	0.065000
0.08	0.114889	0.104000	0.112500
0.09	0.181424	0.155429	0.186875

Table 6.1: Comparison of theoretical upper bound with experimental results for code (72,288) with rate $R = 1/4$

From the results in Table 6.1 we can see that for all p, $PER_{c\_1} < PER_t$, i.e., the theoretical packet-error probability is an upper bound for PER (without *more-candidate-errors*) obtained by Cut-Decoding algorithm. Also, comparing the experimental results obtained by the standard (PER_{s_1}) and Cut-Decoding algorithm (PER_{c_1}) we can conclude that using Cut-Decoding algorithm we decrease the number of unsuccessful decodings with *null-error* and *uncorrected-error*. Namely, for all values of p, $PER_{s\_1} > PER_{c\_1}$.

6.2.2 Comparison for rate $R = 1/8$

Now, we will compare PER_t with the experimental results for packet-error probability (without *more-candidate-errors*) obtained by the standard algorithm, Cut-Decoding algorithm and 4-Sets-Cut-Decoding algorithm for code the (72, 576) with rate $R = 1/8$. The results for these codes and used parameters are given in Section 5.6. In the formula for PER_t we use the same formula for probabilities q_B, as above, since in the experiments for $R = 1/8$ we also used blocks of 4 nibbles (16 bits) in the decoding process. But, now the number of blocks in the codewords (number of iterations) is 36.

The values of PER_t together with the experimental results for PER (without *more-candidate-errors*) obtained by the standard algorithm (PER_{s_1}), Cut-Decoding (PER_{c_1}) and 4-Sets-Cut-Decoding algorithm#1 ($PER_{c4.1\_1}$) are presented in Table 6.2 (for $B_{max} = 4$) and Table 6.3 (for $B_{max} = 5$). We compare the upper bound for PER only with the results obtained by 4-Sets-Cut-Decoding algorithm#1, since in the other versions of this algorithm we have additional methods for increasing the number of *null-errors* and their probabilities are smaller than $PER_{c4.1\_1}$.

p	PER_t	PER_{c_1}	$PER_{c4.1\_1}$	PER_{s_1}
0.02	0.0004	0.0000	0.0000	0.0009
0.03	0.0029	0.0017	0.0020	0.0026
0.04	0.0111	0.0074	0.0086	0.0103
0.05	0.0304	0.0134	0.0254	0.0320
0.06	0.0675	0.0371	0.0600	0.0660
0.07	0.1285	0.0643	0.1129	0.1294
0.08	0.2166	0.1257	0.1906	0.2200

Table 6.2: Comparison of theoretical upper bound with experimental results for $B_{max} = 4$ for code (72,576) with rate $R = 1/8$

Comparing the values of PER_t with experimental results for PER_{c_1} and $PER_{c4.1\_1}$ we can see that the Theorem 6.1 holds for rate $R = 1/8$, too. Namely, with both new algorithms (Cut-Decoding and 4-Sets-Cut-Decoding algorithm) we

p	PER_t	PER_{c_1}	$PER_{c4.1\_1}$	PER_{s_1}
0.03	0.00016	0.00007	0.00007	0.00029
0.04	0.00083	0.00057	0.00057	0.00086
0.05	0.00291	0.00229	0.00286	0.00314
0.06	0.00793	0.00514	0.00714	0.00743
0.07	0.01818	0.01286	0.01571	0.02086
0.08	0.03667	0.02829	0.03171	0.04171
0.09	0.06685	0.04400	0.05914	0.06257
0.10	0.11209	0.08029	0.09486	0.11457
0.11	0.17491	0.12171	0.15086	0.17857
0.12	0.25605	0.14314	0.22229	/

Table 6.3: Comparison of theoretical upper bound with experimental results for $B_{max} = 5$ for code (72,576) with rate $R = 1/8$

obtain $PER_{c\_1} < PER_t$ and $PER_{c4.1\_1} < PER_t$ for all values of p and two considered values of B_{max}.

Also, from the results in Table 6.2 and Table 6.3, we can see that for all values of bit-error probability p of a binary symmetric channel, $PER_{s\_1} > PER_{c\_1}$ and $PER_{s\_1} > PER_{c4.1\_1}$. That confirms that with the new algorithms we have decreased the number of unsuccessful decodings with *null-error* and *uncorrected-error* for rate $R = 1/8$, too. These differences are greater for larger values of p. Moreover, with the third and fourth versions of 4-Sets-Cut-Decoding algorithm we have obtained much greater improvements of the performances in terms of the number of *null-errors*.

6.3 Approximate formulas for cardinality of reduced decoding candidate sets

In [30] the authors give an approximate formula for calculating the cardinality $|S_i|$ of the decoding candidate sets obtained by the standard algorithm of RCBQ. They obtained approximation for $|S_i|$ in the following way.

Consider (N_{block}, N) code with rate R and let $L = L^{(1)}L^{(2)}...L^{(s)}$ be a redundant message. Let N_i be the number of information nibbles in the sub-block $L^{(i)}$ of r nibbles. Let B_{checks} be the cardinality of the set H_i, for $i \geqslant 1$. If we suppose that the strings obtained as output of the inverse coding algorithm are almost random, then the probability for obtaining string with $r - N_i$ zero nibbles (as output) is $\frac{1}{2^{4(r-N_i)}}$. So, the approximate formula for the (expected) cardinality of the first set S_1 is:

$$|S_1| \approx \frac{B_{checks}}{2^{4(r-N_1)}}.$$

Since, in the i^{th} iteration (for $i > 1$) we apply the inverse coding algorithm with the key from each element in the set S_{i-1} and each element in the set H_i, the cardinality of the sets S_i can be approximately calculated by the following formula

$$|S_i| \approx |S_{i-1}| \frac{B_{checks}}{2^{4(r-N_1)}}.$$

These approximate formulas for the cardinality of decoding candidate sets give better approximation if for calculation of $|S_i|$ we use experimentally obtained values of $|S_{i-1}|$ instead of the approximate one (obtained in the previous step of approximation).

Here, we will derive similar approximate formulas for the cardinality of the reduced decoding candidate sets obtained by Cut-Decoding and 4-Sets-Cut-Decoding algorithms. These approximations are obtained using the same assumption, as above, that the process of forming the decoding candidate sets is almost random.

Let $S_1^{(1)}$ and $S_1^{(2)}$ be the sets obtained in the first iteration of Cut-Decoding algorithm before the proposed reduction in Step 4 and Step 5 and let $\left|S_1^{(1)}\right| = n_1^{(1)}$ and $\left|S_1^{(2)}\right| = n_1^{(2)}$. As we mention above we suppose that the decoding candidate sets are random. If N_1 is the number of information nibbles in the sub-block $L^{(1)}$ of r nibbles, then the strings (second parts of the elements) in the sets $S_1^{(j)}, j = 1, 2$ can be consider as strings of $4N_1$ bits. For any string k of $4N_1$ bits, the probability

p_1 that k is a second part of some element in the set $S_1^{(1)}$, i.e., $k \in V_1$ (see algorithm in Section 3.2) is:

$$p_1 = P\{k \in V_1\} = 1 - \left(1 - \frac{1}{2^{4N_1}}\right)^{n_1^{(1)}}.$$

Similarly, the probability p_2 that the string k is a second part of some element in the set $S_1^{(2)}$ is:

$$p_2 = P\{k \in V_2\} = 1 - \left(1 - \frac{1}{2^{4N_1}}\right)^{n_1^{(2)}}.$$

Thus the probability that $k \in V = V_1 \cap V_2$, i.e., that the elements with second part k will be in the reduced sets is:

$$P\{k \in V_1 \cap V_2\} = p_1 p_2 = \left(1 - q^{n_1^{(1)}}\right)\left(1 - q^{n_1^{(2)}}\right),$$

where $q = 1 - \dfrac{1}{2^{4N_1}}$.

Hence, the approximate (due to the assumption for randomness) formula for the (expected) cardinality of the set V in Step 4 of Cut-Decoding algorithm, i.e., the cardinality of the decoding candidate sets $S_1^{(1)}$ and $S_1^{(2)}$ after the reduction in Step 5 is:

$$\left|S_1^{(j)}\right| \approx \sum_k P\{k \in V_1 \cap V_2\} = 2^{4N_1}\left(1 - q^{n_1^{(1)}}\right)\left(1 - q^{n_1^{(2)}}\right), \quad j = 1, 2.$$

Now, let $S_i^{(1)}$ and $S_i^{(2)}$ be the sets obtained in Step 3 (before reduction) of the i^{th} iteration of the decoding process. We suppose that the number of elements in $S_i^{(1)}$ (and $S_i^{(2)}$) obtained from each element in the corresponding decoding candidate set in the previous iteration is approximately equal. This means that we have an approximately equal number of elements in $S_i^{(1)}$ (and $S_i^{(2)}$) with strings that have equal prefix. The experimental results show that this is (approximately) true. Let m_{i-1} be the number of elements in the reduced decoding candidate sets obtained in the $(i-1)^{th}$ iteration and $\left|S_i^{(1)}\right| = n_i^{(1)}$ and $\left|S_i^{(2)}\right| = n_i^{(2)}$. Then using the above assumption, we find that in the set $S_i^{(j)}$, i.e., sets V_j (in Step 4), for $j = 1, 2$, there are m_{i-1} classes of $\dfrac{n_i^{(j)}}{m_{i-1}}$ elements that have strings in the second part with equal prefix of $r(i-1)$ nibbles. So, in each of these classes the elements

differ only in the last r nibbles, i.e., more precisely in N_i nibbles, where N_i is the number of information nibbles in the sub-block $L^{(i)}$ of the redundant message. The intersection $V = V_1 \cap V_2$ (in Step 4) will be non-empty only for the elements from the corresponding classes (of elements with the same prefix) in the sets V_1 and V_2. Using the same method, as for the sets in the first iteration, we have that the cardinality of the intersection of corresponding classes from the sets V_1 and V_2 is approximately:

$$2^{4N_i}\left(1-q^{\frac{n_i^{(1)}}{m_{i-1}}}\right)\left(1-q^{\frac{n_i^{(2)}}{m_{i-1}}}\right),$$

where $q = 1-\frac{1}{2^{4N_i}}$. Since, we have m_{i-1} classes of elements with the same prefix of $r(i-1)$ nibbles, the approximate formula for the cardinality of the reduced decoding candidate sets in the i^{th} iteration is:

$$\left|S_i^{(j)}\right| \approx m_{i-1}\left(2^{4N_i}\left(1-q^{\frac{n_i^{(1)}}{m_{i-1}}}\right)\left(1-q^{\frac{n_i^{(2)}}{m_{i-1}}}\right)\right),\ \ j=1,2.$$

Similarly, we derive approximate formulas for the cardinality of the reduced decoding candidate sets obtained by 4-Sets-Cut-Decoding algorithm#1 . The only difference is the number of sets in this algorithm. Here, we have 4 sets instead of 2. Using the same notations as above and in Section 5.2 we obtain the following approximate formulas:

– for the cardinality of the reduced decoding candidate sets in the first iteration

$$\begin{aligned}\left|S_1^{(j)}\right| &\approx \textstyle\sum_k P\{k \in V_1 \cap V_2 \cap V_3 \cap V_4\}\\ &= 2^{4N_1}\left(1-q^{n_1^{(1)}}\right)\left(1-q^{n_1^{(2)}}\right)\left(1-q^{n_1^{(3)}}\right)\left(1-q^{n_1^{(4)}}\right),\end{aligned}$$

for $j = 1,2,3,4$;

– for the cardinality of the reduced decoding candidate sets in the i^{th} iteration, $i > 1$

$$\left|S_i^{(j)}\right| \approx m_{i-1}\left(2^{4N_i}\left(1-q^{\frac{n_i^{(1)}}{m_{i-1}}}\right)\left(1-q^{\frac{n_i^{(2)}}{m_{i-1}}}\right)\left(1-q^{\frac{n_i^{(3)}}{m_{i-1}}}\right)\left(1-q^{\frac{n_i^{(4)}}{m_{i-1}}}\right)\right),$$

$j = 1,2,3,4$.

With these approximate formulas we obtain good approximation for the cardinality of the decoding candidate sets after the reduction in the new algorithms for RCBQ. The same as previous, the derived formulas give better approximation if for calculation of $|S_i|$ we use experimentally obtained values of $|S_{i-1}|$ instead of an approximate one. The precision is almost the same as in the formulas given in [30] for the standard algorithm (the maximum deviation that we have obtained is 10).

Using these approximations for the cardinality of the decoding candidate sets, we can check whether some pattern for redundancy is a good. Also, the speed of decoding and the number of *more-candidate-errors* depend on the number of elements in the sets S. From the formulas (derived in this section) it is clear that the proposed reduction of the decoding candidate sets, in the new algorithms, significantly decreases the number of elements in the sets.

Conclusion

In this chapter we have proved Теорема 6.1 for the theoretical packet-error probability obtained with the new algorithms proposed in the previous chapters. Also, we derive approximate formulas for cardinality of the reduced decoding candidate sets obtained in the decoding process with the new algorithms. With these results we give theoretical (mathematical) methods for choosing good parameters in the new proposed algorithms for RCBQ. Also, with the derived formulas we prove that with the new algorithms the performances of these codes are improved.

Chapter 7

Parastrophic Quasigroup Transformation and its application in cryptography

The quasigroup string transformations and their properties were considered in several papers. In Chapter 1 we have described E -transformation and we have considered its properties useful for application in cryptography. Using quasigroup parastrophes, Krapež in [35] gave an idea for a quasigroup string transformation that can also be applied in cryptography. In this chapter we define modification of this quasigroup transformation. First, we will describe the new transformation called parastrophic quasigroup transformation and further on, we will consider its cryptographic properties.

7.1 Parastrophes

A quasigroup $(Q, *)$ has a set of five quasigroups, called parastrophes denoted by $/, \backslash, \cdot, //, \backslash\backslash$ which are defined in Table 7.1 .

Parastrophe operations				
$x\backslash y = z$	$\Longleftrightarrow$	$x * z = y$		
$x/y = z$	$\Longleftrightarrow$	$z * y = x$		
$x \cdot y = z$	$\Longleftrightarrow$	$y * x = z$		
$x//y = z$	$\Longleftrightarrow$	$y/x = z$	$\Longleftrightarrow$	$z * x = y$
$x\backslash\backslash y = z$	$\Longleftrightarrow$	$y\backslash x = z$	$\Longleftrightarrow$	$y * z = x$

Table 7.1: Parastrophes of quasigroup operation $*$

For each $f \in \{*, \backslash, /, \cdot, //, \backslash\backslash\}$, (Q, f) is a quasigroup.

Example 7.1. Let $Q = \{1, 2, 3, 4\}$. An example of a quasigroup $(Q, *)$ of order 4 and its parastrophes, are given on (7.1).

$*$	1	2	3	4
1	1	2	3	4
2	2	1	4	3
3	4	3	1	2
4	3	4	2	1

$/$	1	2	3	4
1	1	2	3	4
2	2	1	4	3
3	4	3	1	2
4	3	4	2	1

$\backslash$	1	2	3	4
1	1	2	3	4
2	2	1	4	3
3	3	4	2	1
4	4	3	1	2

(7.1)

$\cdot$	1	2	3	4
1	1	2	4	3
2	2	1	3	4
3	3	4	1	2
4	4	3	2	1

$//$	1	2	3	4
1	1	2	4	3
2	2	1	3	4
3	3	4	1	2
4	4	3	2	1

$\backslash\backslash$	1	2	3	4
1	1	2	3	4
2	2	1	4	3
3	3	4	2	1
4	4	3	1	2

We must note that not all parastrophes of a quasigroup are different. We can see that from the 6 quasigroups given in Example 7.1, only three are different. Using the number of different parastrophes of each quasigroup, in Section 7.3.1 we will give a classification of the quasigroups of order 4.

In this thesis we use the following notation for the quasigroup operation $*$ and its parastrophes:

$$f_1(x, y) = x * y, \quad f_2(x, y) = x\backslash y, \quad f_3(x, y) = x/y,$$
$$f_4(x, y) = x \cdot y, \quad f_5(x, y) = x//y, \quad f_6(x, y) = x\backslash\backslash y.$$

7.2 Parastrophic Quasigroup Transformation

Let $A = \{1, \ldots, a\}$ be an alphabet of integers ($a \geqslant 2$) and denote by $A^+ = \{x_1 \ldots x_k |\ x_i \in A,\ k \geqslant 1\}$ the set of all finite strings over A. Note that $A^+ = \bigcup_{k \geqslant 1} A^k$, where $A^k = \{x_1 \ldots x_k |\ x_i \in A\}$.

Let $M = x_1 x_2 \ldots x_k$ be an input message. Let d_1 be an random integer such that $2 \leqslant d_1 < k$ and l be a random chosen element (leader) from A. Also, let $(A, *)$ be a quasigroup and $f_1, \ldots, f_6$ be its parastrophe operations.

Using previous transformation E, for chosen l, d_1 and a quasigroup $(A, *)$ we define a parastrophic transformation $PE = PE_{l,d_1} : A^+ \to A^+$ as follows.

At first, let $q_1 = d_1$ be the length of the first block, i.e., $M_1 = x_1x_2 \dots x_{q_1-1}x_{q_1}$. Let $s_1 = (d_1 \bmod 6) + 1$. Applying the transformation E on the block M_1 with leader l and quasigroup operation f_{s_1}, we obtain the encrypted block

$$C_1 = y_1y_2 \dots y_{q_1-1}y_{q_1} = E_{f_{s_1},l}(x_1x_2 \dots x_{q_1-1}x_{q_1}).$$

Further on, using the last two symbols in C_1 we calculate the number $d_2 = 4y_{q_1-1} + y_{q_1}$ which determines the length of the next block. Let $q_2 = q_1 + d_2$, $s_2 = (d_2 \bmod 6) + 1$ and $M_2 = x_{q_1+1} \dots x_{q_2-1}x_{q_2}$. After applying $E_{f_{s_2},y_{q_1}}$, the encrypted block C_2 is

$$C_2 = y_{q_1+1} \dots y_{q_2-1}y_{q_2} = E_{f_{s_2},y_{q_1}}(x_{q_1+1} \dots x_{q_2-1}x_{q_2}).$$

In general case, for given i, let the encrypted blocks C_1,..., C_{i-1} be obtained and d_i be calculated using the last two symbols in C_{i-1}, i.e., $d_i = 4y_{q_{i-1}-1} + y_{q_{i-1}}$. Let $q_i = q_{i-1} + d_i$, $s_i = (d_i \bmod 6) + 1$ and $M_i = x_{q_{i-1}+1} \dots x_{q_i-1}x_{q_i}$. We apply the transformation $E_{f_{s_i},y_{q_{i-1}}}$ on the block M_i and obtain the encrypted block

$$C_i = E_{f_{s_i},y_{q_{i-1}}}(x_{q_{i-1}+1} \dots x_{q_i-1}x_{q_i}).$$

Now, the parastrophic transformation is defined as

$$PE_{l,d_1}(M) = PE_{l,d_1}(x_1x_2 \dots x_n) = C_1||C_2|| \dots ||C_r, \tag{7.2}$$

where $||$ is a concatenation of blocks. Note that the length of the last block M_r may be shorter than d_r. It depends on the number of letters in the input message. The transformation PE is schematically presented in Figure 7.1.

For arbitrary quasigroup on a set A, random leaders $l_1, \dots l_n$ and random lengths $d_1^{(1)}, \dots, d_1^{(n)}$, we define mappings PE_1, PE_2, ..., PE_n as in (7.2) such that PE_i is corresponding to $d_1^{(i)}$ and l_i. Using them, we define the transformation $PE^{(n)}$ as follows:

$$PE^{(n)} = PE^{(n)}_{(l_n,d_1^{(n)}),\dots,(l_1,d_1^{(1)})} = PE_n \circ PE_{n-1} \circ \dots \circ PE_1,$$

where $\circ$ is the usual composition of mappings.

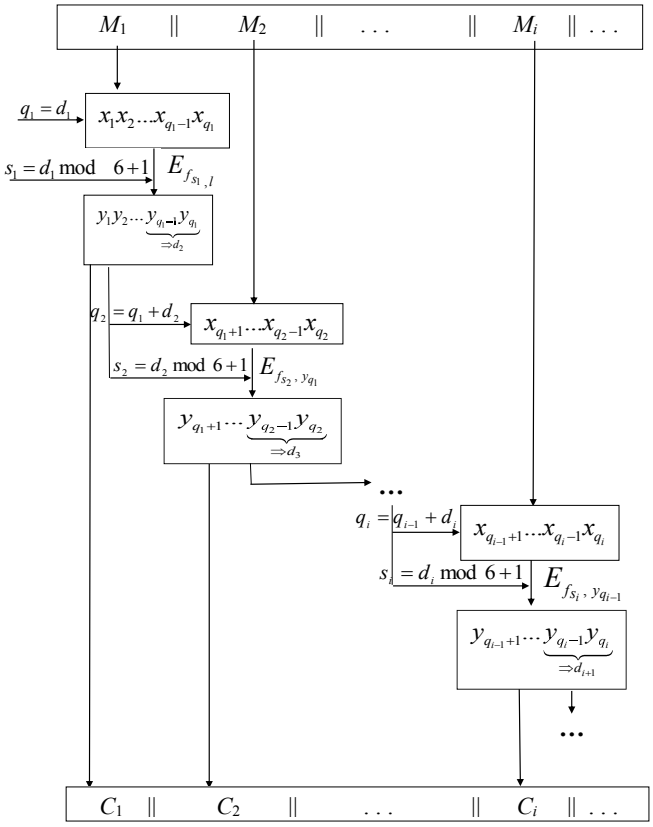


Figure 7.1: Parastrophic transformation PE

We made experiments with PE-transformation using different ways to determine the length of the next block and the quasigroup operation, in each iteration. Analyzing the output, we get the best results if we use the last two symbols from the previous encrypted block in order to compute the length of the next block and the corresponding quasigroup operation. Namely, if we take one symbol then we can obtain only 4 values to choose the quasigroup operations, but we have 6 parastrophes. On the other hand, when we take more than 2 symbols to compute the length of the next block and the corresponding quasigroup operation, we conclude that the parastrophes do not change often enough. Therefore, in these cases we

obtain worse results in terms of fractality, i.e., we have not increased the number of quasigroups suitable for cryptography (Section 7.3).

7.3 Classifications of quasigroups of order 4 useful in cryptography

Using an image pattern, Dimitrova and Markovski give a classification of quasigroups of order 4 in [17] as fractal and non-fractal. The number of fractal quasigroups of order 4 is 192 and the number of non-fractal quasigroups is 384. Fractal quasigroups are not good for designing cryptographic primitives since they produce regular structures.

In order to increase the number of quasigroups suitable for application in cryptography, we give new classifications of quasigroups of order 4:

- the classification by number of different parastrophes;
- the classification by parastrophic fractality.

7.3.1 Classification by number of different parastrophes

We consider all 576 quasigroups of order 4 and for each quasigroup we find the set of all parastrophes. The cardinality of each of these sets is less than or equal to 6, i.e., not all parastrophes of a quasigroup are different.

Using the number of different parastrophes of each quasigroup we divide the set of all quasigroups of order 4 into 4 classes. The number of quasigroups in each of these classes is given in Table 7.2.

No. parastrophes	No. quasigroups
1	16
2	2
3	240
6	318
Total	576

Table 7.2: Cardinality of classes by number of different parastrophes

In Table 7.3 we give a sub-classification of the previous one. Namely, according to the number of different parastrophes we classify separately, the fractal and non-fractal quasigroups of order 4.

No. parastrophes	No. fractal quasigroups	No. non-fractal quasigroups
1	16	0
2	2	0
3	96	144
6	78	240
Total	192	384

Table 7.3: Cardinality of subclasses by number of different parastrophes

From Table 7.3, we can see that the class of fractal quasigroups of order 4 is divided in 4 subclasses, and the class of non-fractal quasigroups is divided in just 2 subclasses. Comparing Table 7.2 and Table 7.3 we can conclude that all quasigroups with 1 or 2 parastrophes are fractal. Quasigroups with 3 and 6 parastrophes can be fractal or non-fractal.

The following proposition is proved by exhaustive verification.

Proposition 7.1. *Parastrophes of fractal quasigroups of order 4 are fractal as well.*

Consequently, all non-fractal quasigroups of order 4 have non-fractal parastrophes.

7.3.2 Classification by parastrophic fractality

Using an image pattern as in [17], we made a similar classification using new PE-transformation instead of E-transformation. We apply the new transformation $PE^{(n)}$ to the sequence $123412341234...$ as before and consider the fractal structure of the obtained image. Depending on that structure, we define new types of fractal quasigroups.

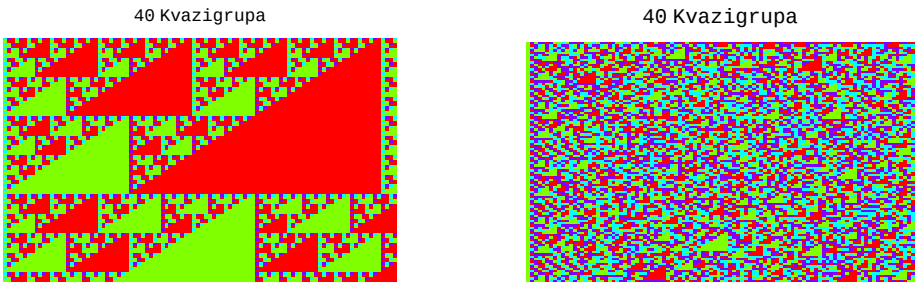
Definition 7.1. *Quasigroups with fractal structure obtained after applying of PE-transformation are called **parastrophic-fractal quasigroups**. In the opposite case, the quasigroup is called **parastrophic-non-fractal quasigroup**.*

We made experiments with the image pattern for all 576 quasigroups of order 4 and found that 88 are parastrophic-fractal and 488 are parastrophic-non-fractal. We conclude that all parastrophic-fractal quasigroups are in the class of fractal quasigroups, but not all fractal quasigroups are parastrophic-fractal. This means that the class of all 192 fractal quasigroups is divided in 2 subclasses:

1) parastrophic-fractal (88 quasigroups) and

2) fractal, but parastrophic-non-fractal (104 quasigroups), called ***fractal parastrophic-non-fractal quasigroups***.

In Figure 7.2, we give the image patterns obtained with the quasigroup with a lexicographic number 40 which is fractal (a), but it is parastrophic-non-fractal (b).



(a) (b)

Figure 7.2: Fractal, but parastrophic-non-fractal quasigroup

On the other hand, the class of all 384 non-fractal quasigroups is completely contained in the class of parastrophic-non-fractal quasigroups.

According to the above, we make a new classification of the quasigroups of order 4:

1) class of parastrophic-fractal quasigroups (88 quasigroups);

2) class of fractal parastrophic-non-fractal quasigroups (104 quasigroups);

3) class of non-fractal quasigroups (384 quasigroups).

With this classification we increase the number of quasigroups useful for application in cryptography. Namely, for cryptographic purposes we can use not only the quasigroups from class 3, but the quasigroups from class 2, too.

No. parastrophes	No. parastrophic-fractal	No. fractal parastrophic-non-fractal
1	16	0
2	0	2
3	72	24
6	0	78
Total	88	104

Table 7.4: Cardinality of subclass of fractal quasigroups by number of different parastrophes

Further on, according to the number of different parastrophes, we find the cardinality of subclasses of parastrophic-fractal and fractal parastrophic-non-fractal quasigroups, separately. The results are given in Table 7.4.

Comparing Table 7.3 and Table 7.4 we can conclude that all fractal quasigroups with 1 parastrophe are parastrophic-fractal, all fractal quasigroups with 2 and 6 different parastrophes are fractal parastrophic-non-fractal quasigroups. Only the subclass of fractal quasigroups with 3 different parastrophes contains both parastrophic-fractal and fractal parastrophic-non-fractal quasigroups.

7.4 Algebraic properties of parastrophic fractal quasigroups

In [45], the authors give a mathematical model of fractality of quasigroups using some identities. In this subsection our goal is to find a similar model, but for parastrophic fractality of quasigroups. For this purpose we investigate the algebraic properties of parastrophic-fractal quasigroups of order 4. In order to find suitable identities to separate parastrophic-fractal quasigroups, we investigated many identities, especially symmetric ones . Essential for our model are the following:

- commutativity ($x * y = y * x$),
- skew symmetry ($(x * y) * (y * x) = const$),
- left loops ($e * x = x$),
- right loops ($x * e = x$),

– right symmetry ($(x * y) * y = x$),

– left symmetry ($y * (y * x) = x$),

– total symmetry (commutativity and left symmetry).

Using an exhaustive verification we find that each parastrophic-fractal quasigroup of order 4 belongs to the set of quasigroups I that satisfies the identity $x * (x * (x * (x * y))) = y$ and belongs to one of the following sets of quasigroups:

– Loops (L)

– Totally Symmetric quasigroups (TS)

– Left Loops (LL) and Right Symmetric quasigroups (RS)

– Right Loops (RL) and Left Symmetric quasigroups (LS)

– Left Loops (LL) and Skew Symmetric quasigroups (SS)

– Right Loops (RL) and Skew Symmetric quasigroups (SS)

– Commutative quasigroups (C) and Skew Symmetric quasigroups (SS).

Using the above notation, we give the following proposition:

Proposition 7.2. *The set of all parastrophic-fractal quasigroups PFQ of order 4 can be presented as:*

$$PFQ = I \cap [L \cup TS \cup (LL \cap RS) \cup (RL \cap LS) \cup (SS \cap (LL \cup RL \cup C))].$$

In this way, we have the mathematical model for parastrophic fractality and without using the image pattern we can check if a given quasigroup is parastrophic-fractal.

7.5 Theoretical proof for resistance to statistical kind of attacks

To complete the proof of goodness of parastrophic quasigroup transformation for cryptography, we have to prove that Theorem 1.2 holds for this transformation, too. An important property of one transformation for application in cryptography is the uniform distribution of the substrings in the output message. It will

guarantee that message encrypted by the parastrophic quasigroup transformation will be resistant to a statistical kind of attacks. Therefore, in this section, we will investigate the distribution of substrings in the output message obtained using PE-transformation.

Let the alphabet A be as above. A randomly chosen element of the set A^k can be considered as a random vector $(X_1, X_2, \ldots, X_k)$, where A is the range of X_i, $i = 1, \ldots, k$. We consider that vector as an input message. The transformation $PE = PE_{l,d_1} : A^+ \to A^+$ can be defined as:

$$PE_{l,d_1}(X_1, \ldots, X_k) = (Y_1, \ldots, Y_k) \Leftrightarrow$$

$$\Leftrightarrow \begin{cases} Y_1 = f_{s_1}(l, X_1), \;\; Y_j = f_{s_1}(Y_{j-1}, X_j), \; j = 2, \ldots, d_1, \\ \\ Y_{q_i+j} = f_{s_{i+1}}(Y_{q_i+j-1}, X_{q_i+j}), i = 1, \ldots, r-1, \; j = 1, \ldots, d_{i+1} \end{cases} \tag{7.3}$$

Let $(p_1, p_2, ..., p_a)$ be the probability distribution of the letters $1, ..., a$ in an input message. That implies $p_i > 0$ for each $i = 1, 2, ..., a$ and $\sum_{i=1}^{a} p_i = 1$.

At first we will prove that after applying the transformation $PE^{(1)}$ on an input message α, the letters in transformed message are uniformly distributed.

Theorem 7.1. *The letter Y_t has uniform distribution on the set $A = \{1, \ldots, a\}$, i.e., $Y_t \sim U(\{1, \ldots, a\})$ for each t $(t = 1, 2, \ldots, k)$.*

Proof. In this proof we use the same notations as in construction of parastrophic quasigroup transformation given in the previous section.

At first, note that the leader l can be considered as uniformly distributed random variable on the set A since it is randomly chosen from the set A. Therefore, $l \sim U(\{1, \ldots, a\})$, i.e.,

$$P\{l = i\} = \frac{1}{a}, \qquad \text{for each } i \in A.$$

Also, the leader l is independent of each letter X_i in the input message.

Let $t = 1$. Using the equation (7.3) and the total probability theorem, for distribution of Y_1, we obtain

$$\begin{aligned} P\{Y_1 = j\} &= P\{f_{s_1}(l, X_1) = j\} \\ &= \sum_{i=1}^{a} P\{l = i\} P\{f_{s_1}(l, X_1) = j | l = i\} \\ &= \sum_{i=1}^{a} \frac{1}{a} P\{f_{s_1}(l, X_1) = j | l = i\} \\ &= \sum_{i=1}^{a} \frac{1}{a} P\{f_{s_1}(i, X_1) = j\} \\ &= \frac{1}{a} \sum_{i=1}^{a} P\{X_1 = f'_{s_1}(i, j)\} \end{aligned}$$

Here, f'_{s_1} is the inverse quasigroup transformation of f_{s_1}, i.e. if $f_{s_1}(u, x) = v$, then $f'_{s_1}(u, v) = x$. Note that if i runs over all values of A then for fixed j, the expression $X_1 = f'_{s_1}(i, j)$ runs over all values of A, too. Therefore,

$$P\{Y_1 = j\} = \frac{1}{a} \sum_{i=1}^{a} P\{X_1 = f'_{s_1}(i, j)\} = \frac{1}{a} \sum_{i=1}^{a} p_i = \frac{1}{a},$$

i.e., $Y_1 \sim U(\{1, \ldots, a\})$.

We proceed by induction, and assuming that $Y_r \sim U(\{1, 2, \ldots, a\})$. Similarly as previous, using that $f_{s_{r+1}}$ is the parastrophe operation applied in $(r+1)^{th}$ step we compute the distribution of Y_{r+1} as follows.

$$\begin{aligned} P\{Y_{r+1} = j\} &= P\{f_{s_{r+1}}(Y_r, X_{r+1}) = j\} \\ &= \sum_{i=1}^{a} P\{Y_r = i\} P\{f_{s_{r+1}}(Y_r, X_{r+1}) = j | Y_r = i\} \\ &= \sum_{i=1}^{a} \frac{1}{a} P\{f_{s_{r+1}}(i, X_{r+1}) = j | Y_r = i\} \end{aligned}$$

According to the definition of parastrophic quasigroup transformation, given with (7.3), we can conclude that the random variables X_{r+1} and Y_r are independent.

Applying that in the previous equation, we obtain

$$\begin{aligned} P\{Y_{r+1}=j\} &= \sum_{i=1}^{a}\frac{1}{a}P\{f_{s_{r+1}}(i,X_{r+1})=j\} \\ &= \frac{1}{a}\sum_{i=1}^{a}P\{X_{r+1}=f'_{s_{r+1}}(i,j)\} \\ &= \frac{1}{a}. \end{aligned}$$

As previous, $f'_{s_{r+1}}$ is the inverse quasigroup transformation of $f_{s_{r+1}}$. In the last equation, we use that $X_{r+1}=f'_{s_{r+1}}(i,j)$ runs over all values of A when j is fixed and i runs over all values of A, i.e.

$$\sum_{i=1}^{a}P\{X_{r+1}=f'_{s_{r+1}}(i,j)\}=\sum_{i=1}^{a}p_i=1.$$

In this way, we proved that Y_t has uniform distribution on the set A, for each $t\geqslant 1$. □

From Theorem 7.1 we can conclude the following. If $M\in A^k$ and $C=PE_{l,d_1}(M)$ then the letters in the message C are uniformly distributed, i.e., the probability of the appearance of a letter i at an arbitrary place of the string C is $\frac{1}{a}$, for each $i\in A$.

Theorem 7.2. *Let $M\in A^+$ be an arbitrary string and $C=PE^{(n)}(M)$. Then the m-tuples in C are uniformly distributed for $m\leqslant n$.*

Proof. Let $(Y_1^{(n)},Y_2^{(n)},\ldots,Y_k^{(n)})=PE^{(n)}(X_1,X_2,\ldots,X_k)$. We will prove this theorem by induction. Let suppose that the statement is satisfied for $n=r$, i.e., $(Y_{t+1}^{(r)},Y_{t+2}^{(r)}\ldots Y_{t+l}^{(r)})\sim U(\{1,2,\ldots,a\}^l)$ for each $1\leqslant l\leqslant r$ and each $t\geqslant 0$. Now, let $n=r+1$. We consider the distribution of $(Y_{t+1}^{(r+1)},Y_{t+2}^{(r+1)}\ldots Y_{t+l}^{(r+1)})$ for each $1\leqslant l\leqslant r+1$ and arbitrary t.

$$\begin{aligned} &P\{Y_{t+1}^{(r+1)}=y_{t+1}^{(r+1)},Y_{t+2}^{(r+1)}=y_{t+2}^{(r+1)},\ldots,Y_{t+l}^{(r+1)}=y_{t+l}^{(r+1)}\} \\ &\quad=P\{Y_{t+1}^{(r+1)}=y_{t+1}^{(r+1)},f_{s_{t+2}}(Y_{t+1}^{(r+1)},Y_{t+2}^{(r)})=y_{t+2}^{(r+1)},\ldots \\ &\qquad\qquad\ldots,f_{s_{t+l}}(Y_{t+l-1}^{(r+1)},Y_{t+l}^{(r)})=y_{t+l}^{(r+1)}\}, \end{aligned}$$

where f_{s_j} is the parastrophe operation applied in the step j and f'_{s_j} is its inverse transformation, $j = t+2, \ldots, t+l$. Now,

$$\begin{aligned}
&P\{Y_{t+1}^{(r+1)} = y_{t+1}^{(r+1)}, Y_{t+2}^{(r+1)} = y_{t+2}^{(r+1)}, \ldots, Y_{t+l}^{(r+1)} = y_{t+l}^{(r+1)}\} \\
&\quad = P\{Y_{t+1}^{(r+1)} = y_{t+1}^{(r+1)}, f_{s_{t+2}}(y_{t+1}^{(r+1)}, Y_{t+2}^{(r)}) = y_{t+2}^{(r+1)}, \ldots \\
&\qquad\qquad \ldots, f_{s_{t+l}}(y_{t+l-1}^{(r+1)}, Y_{t+l}^{(r)}) = y_{t+l}^{(r+1)}\} \\
&\quad = P\{Y_{t+1}^{(r+1)} = y_{t+1}^{(r+1)}, Y_{t+2}^{(r)} = f'_{s_{t+2}}(y_{t+1}^{(r+1)}, y_{t+2}^{(r+1)}), \ldots \\
&\qquad\qquad \ldots, Y_{t+l}^{(r)} = f'_{s_{t+l}}(y_{t+l-1}^{(r+1)}, y_{t+l}^{(r+1)})\} \\
&\quad = P\{Y_{t+1}^{(r+1)} = y_{t+1}^{(r+1)}\} P\{Y_{t+2}^{(r)} = f'_{s_{t+2}}(y_{t+1}^{(r+1)}, y_{t+2}^{(r+1)}), \ldots \\
&\qquad\qquad \ldots, Y_{t+l}^{(r)} = f'_{s_{t+l}}(y_{t+l-1}^{(r+1)}, y_{t+l}^{(r+1)})\}.
\end{aligned}$$

The last equality is obtained by using the fact that $Y_{t+1}^{(r+1)}$ is independent of the vector $(Y_{t+2}^{(r)}, \ldots, Y_{t+l}^{(r)})$, since $Y_{t+2}^{(r)}, \ldots, Y_{t+l}^{(r)}$ are not used for obtaining $Y_{t+1}^{(r+1)}$.

Using the inductive hypothesis $(Y_{t+2}^{(r)}, \ldots, Y_{t+l}^{(r)}) \sim U(\{1, 2, \ldots, a\}^{l-1})$, $Y_{t+1}^{(r+1)} \sim U(\{1, 2, \ldots, a\})$ and from the previous expression we obtain that

$$P\{Y_{t+1}^{(r+1)} = y_{t+1}^{(r+1)}, Y_{t+2}^{(r+1)} = y_{t+2}^{(r+1)}, \ldots, Y_{t+l}^{(r+1)} = y_{t+l}^{(r+1)}\} = \frac{1}{a} \cdot \frac{1}{a^{l-1}} = \frac{1}{a^l}.$$

So, we have proved that $(Y_{t+1}^{(n)}, Y_{t+2}^{(n)} \ldots Y_{t+l}^{(n)}) \sim U(\{1, 2, \ldots, a\}^l)$ for each $l \leqslant n$ and each $t \geqslant 0$. □

7.6 Experimental results

We made many experiments in order to present our theoretical results. Here we give an example. We have randomly chosen a message M with 1,000,000 letters of the alphabet $A = \{1, 2, 3, 4\}$ with the distribution of letters given in Table 7.5.

1	2	3	4
0.70	0.15	0.10	0.05

Table 7.5: The distribution of letters in the input message

We used the quasigroup (7.4) and its parastrophes.

$*$	1	2	3	4
1	3	2	1	4
2	1	4	2	3
3	4	1	3	2
4	2	3	4	1

(7.4)

After applying $PE^{(3)}$ on M, we got the encrypted message $C = PE^{(3)}(M)$. In each PE-transformation, we chose the length of the first block $d_1 = 3$ and the initial leader $l_1 = 1$.

The distribution of letters in the output message C is given in Table 7.6.

1	2	3	4
0.2505	0.2498	0.2502	0.2494

Table 7.6: The distribution of letters in the output message

From the probabilities in Table 7.6, it is obvious that the distribution of letters in the output message C is uniform.

The distribution of pairs, triplets and 4-tuples of letters in C are given in Figure 7.3, Figure 7.4 and Figure 7.5. In Figure 7.3, the pairs are presented on the x-axis in the lexicographic order ($'11' \to 1, '12' \to 2, \ldots, '44' \to 16$). In the similar way, the triplets and 4-tuples are presented in Figure 7.4 and Figure 7.5.

We can see in Figure 7.3 and Figure 7.4 that after three applications of PE-transformation, the pairs and triplets are also uniformly distributed as we proved in Theorem 7.2. We can also see in Figure 7.5 that the distribution of the 4-tuples in C is not uniform, but that distribution is closer to the uniform distribution than the distribution of 4-tuples in the input message.

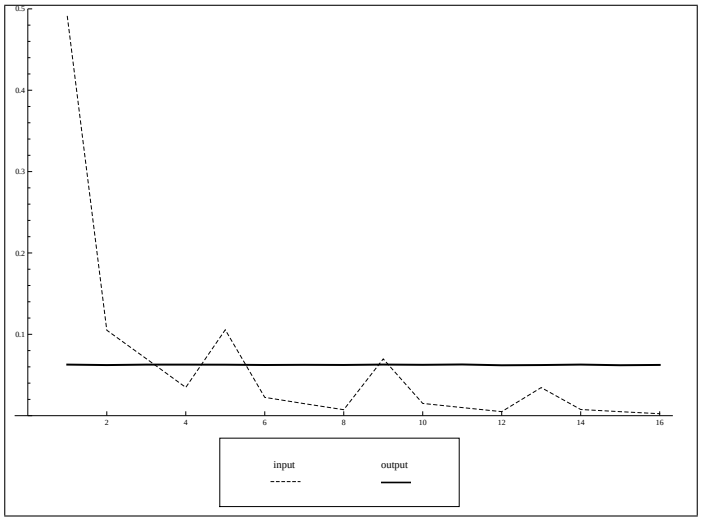


Figure 7.3: The distribution of pairs in the input message and the output message

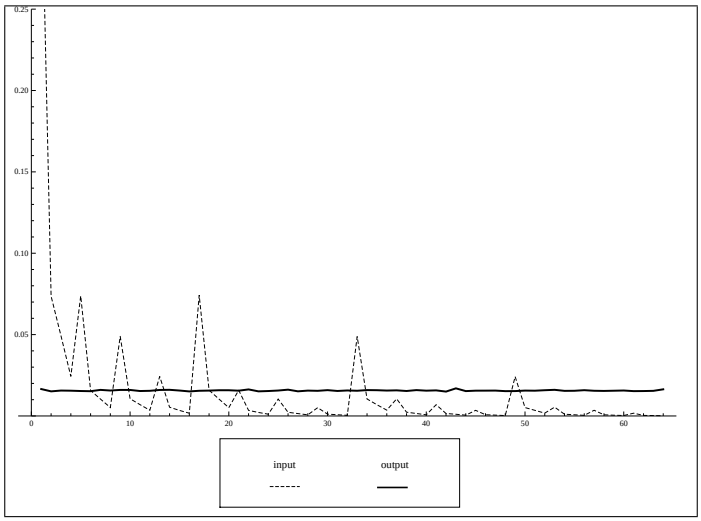


Figure 7.4: The distribution of triplets in the input message and the output message

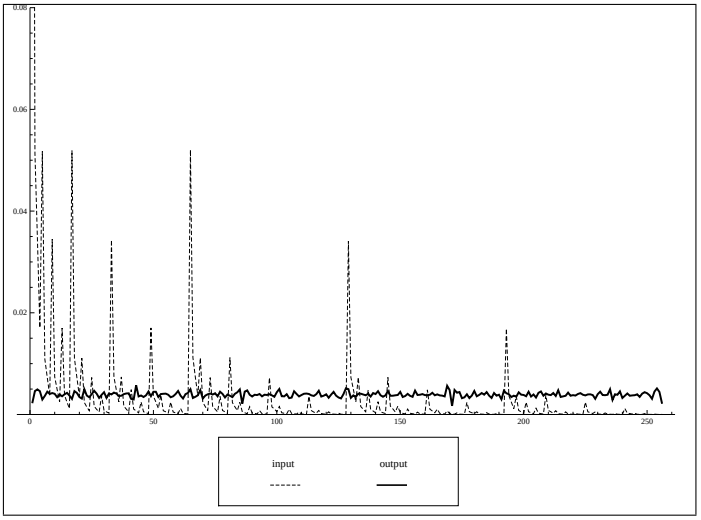
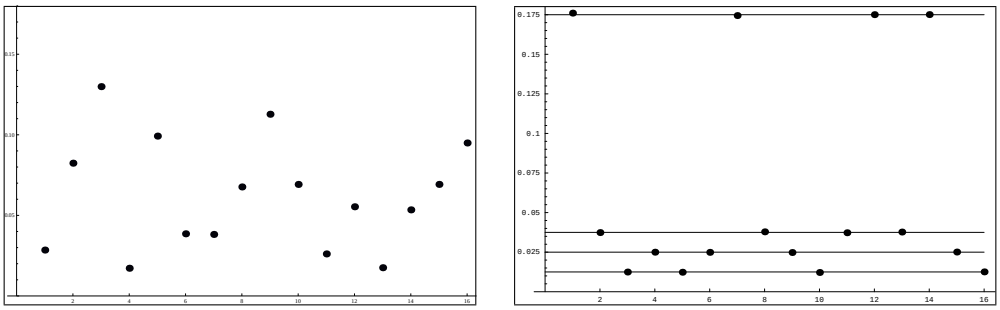


Figure 7.5: The distribution of 4-tuples in the input message and the output message

In Theorem 1.4 ([2]), Bakeva and Dimitrova proved that the probabilities of $(n+1)$-tuples in $\beta = E^{(n)}(\alpha)$ are divided in a classes where $a = |A|$, if $(p_1, p_2, \ldots, p_a)$ is the distribution of letters in the input string and $p_1, p_2, \ldots, p_a$ are distinct probabilities, i.e., $p_i \neq p_j$ for $i \neq j$. Each class contains a^n elements with the same probabilities and the probability of each $(n+1)$-tuple in i-th class is $\frac{1}{a^n}p_i$, for $i = 1, 2, \ldots, a$. If $p_{i_1} = p_{i_2} = \cdots = p_{i_\nu}$ for some $1 \leqslant i_1 < \cdots < i_\nu \leqslant a$, then the classes with probabilities $\frac{1}{a^n}p_{i_1} = \frac{1}{a^n}p_{i_2} = \cdots = \frac{1}{a^n}p_{i_\nu}$ will be merged in one class with νa^n elements. Using these results, the authors proposed an algorithm for cryptanalysis.

Next, we check whether this theorem is satisfied when PE-transformation is applied. The distribution of pairs after one application of PE-transformation is presented in Figure 7.6 a). In Figure 7.6 b), we present the distribution of pairs after one application of E-transformation. We can see that probabilities of pairs in Figure 7.6 b) are divided in 4 classes as Theorem1.4 ([2]) claims. But we cannot distinguish any classes for the probabilities in Figure 7.6 a). This means that the algorithm for cryptanalysis proposed in [2] cannot be applied when an input message is encrypted by PE-transformation. Therefore we can conclude that encryption by PE-transformation is more resistant to statistical kind of attacks.



a) b)

Figure 7.6: The distribution of the pairs in output messages obtained by PE- and E-transformation

Note that for relevant statistical analyses, we must have an input message large enough. Namely, in experiments, the probabilities of n-tuples are computed as relative frequencies. So, a relative frequency of an event tends to probability only if we have sample large enough. The relevant statistical analysis cannot be done for a shorter message. Therefore, statistical kind of attack is impossible on not enough large input message. Note that if an intruder catches and concatenates a lot of short messages encrypted by the same $PE^{(n)}$-transformation, it will obtain a long message and it can apply a statistical attack. But, the attack will be impossible if we change quasigroups used in encryption with $PE^{(n)}$-transformation more often.

In [47], the authors concluded that E-transformation can be applied in cryptography as encryption function since the number of quasigroups is huge (there are more than 10^{58000} quasigroups when $|A| = 256$) and the brute force attack is not feasible.

If PE-transformation is used in encryption algorithm then the secret key will be a triplet $(*, l, d_1)$. In that case, the brute force attack also is not possible since the key contains not only the quasigroup operation $*$ and the leader l, but also the length of the first block d_1 which has influence on the dynamic of changing of parastrophes.

Conclusion

In this chapter we define new quasigroup transformation (PE-transformation) and we make two new classifications of the quasigroups of order 4 useful in cryptogra-

phy. Also, we give a mathematical model for parastrophic fractality and we prove Theorem 7.1 and Theorem 7.2 for the uniform distribution of the substrings in the message obtained with the new transformation.

If we summarize the obtained results we can conclude that:

- parastrophic-fractal quasigroups should not be used for cryptographic primitives, since they have a fractal structure and properties of symmetry;
- we have a mathematical model for separating the parastrophic-fractal quasigroups from other ones;
- the parastrophic transformation is more suitable to design cryptographic primitives, since it increases the number of quasigroups useful for designing of cryptographic primitives.

Finally, from all results presented in this chapter, we can conclude that PE-transformation is better as encryption function than E-transformation.

Bibliography

[1] Bakeva V.: *Probabilistic model in cryptography and coding theory*, Pliska Studia Mathematica Bulgarica, Vol.16, (2004), pp.13-22.

[2] Bakeva V., Dimitrova V.: *Some Probabilistic Properties of Quasigroup Processed Strings useful in Cryptanalysis*, Gusev, M., Mitrevski, P. (eds.) ICT-Innovations 2010, Springer (2010), pp. 61-70.

[3] Bakeva V., Dimitrova V., Popovska-Mitrovikj A.: *Parastrophic Quasigroup String Processing*, Proc. of the 8^{th} Conference on Informatics and Information Technology with International Participants, Macedonia (2011) pp. 19-21.

[4] Bakeva V., Ilievska, N.: *A probabilistic model of error-detecting codes based on quasigroups*, Quasigroups and Related Systems, Vol. 17 (2009), pp. 135-148.

[5] Bakeva V., Popovska-Mitrovikj A., Dimitrova V.: *Resistance to statistical attacks of Parastrophic Quasigroup Transformation*, submitted to Serdica Journal of Computing

[6] Belousov V. D.: n*-arnie Kvazigruppi* (*n-ary Quasigroups*), Stiinca, Kisiniev, 1972.

[7] Bossert M.: *Channel Coding for Telecommunications*, John Wiley and Sons, Ltd, 1999.

[8] Bowman J. C.: *Coding Theory*, University of Alberta, Edmonton, Canada, 2003.

[9] Cooke B.: *Reed-Muller Error Correcting Codes*, MIT Undergraduate Journal of Mathematics, pp. 21-26.

[10] Cox D.R., Miller H.D.: *The Theory of Stochastic Processes*, Chapman and Hall, 1994.

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

[12] Denes J., Keedwell A. D: *Latin squares: New developments in the theory and applications*, Elsevier science publishers, 1991.

[13] Denes J., Keedwell A. D: *Some applications of non-associative algebraic systems in cryptology*, Pure Mathematics and Applications 12(2), 2001, 147-195.

[14] 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.

[15] Dimitrova V., Markovski J.: *On Quasigroup Pseudo Random Sequence Generators*, Proceedings of the 1st Balkan Conference in Informatics, Thessaloniki, Greece, (2003) pp. 393–401.

[16] Dimitrova V., Markovski J.: *Implementation of pseudo random sequence generator using quasigroup processing*, Proc. of the VI National Conference ETAI 2003, 17-20 Sep. 2003, Ohrid, Macedonia, pp. I-86–I-90.

[17] 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.

[18] Dimitrova V., Markovski S.: *Construction of n-ary qusigroups*, III Congress of Mathematicians of Macedonia, Macedonia, 2005.

[19] Dimitrova V., Markovski S., Mileva A.: *Periodic quasigroups string transformations*, The Journal Quasigroups and Related Systems vol.17 No. 2, 2009, pp. 191-204.

[20] Gligoroski D.: *Stream cipher based on quasigroup string transformations in* $\mathbb{Z}_p^*$, Contributions, Sec. Math. Tech. Sci., MANU, 2004.

[21] 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.

[22] Gligoroski D., Knapskog S.: *Edon-$\mathcal{R}(256, 384, 512)$-an Efficient Implementation of Edon-$\mathcal{R}$ Family of Cryptographic Hash Functions*, Cryptology ePrint Archive, Report 2007/154.

[23] Gligoroski D., Knapskog S.: *Adding MAC Functionality to Edon80*, International Journal of Computer Science and Network Security 7(1), 2004, 194-204.

[24] Gligoroski D., Markovski S., Bakeva V.: *On infinite class of strongly collision resistant hash functions "EDON-F" with variable length of output*, Proceedings of the 1st MII 2003 Conference, Thessaloniki, April 14-16,2003,pp. 302-308.

[25] Gligoroski D., Markovski S., Knapskog S. J.: *The Stream Cipher Edon80*, New Stream Cipher Designs: The eSTREAM Finalists, 152-169, 2008, Springer-Verlag.

[26] Gligoroski D., Markovski S., Kocarev Lj.: *Edon-$\mathcal{R}$, an Infinite Family of Cryptographic Hash Functions*, The Second NIST Cryptographic Hash Workshop, UCSB, Santa Barbara, CA, 2006, pp. 275–285.

[27] 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.

[28] Gligoroski D., Markovski S., Kocarev Lj.: *A Synchronous Stream Cipher + Redundancy = Cryptcoding - Extended Abstract*, The 2007 Intern. Confer. on Security and Management SAM'07, June 2007, Las Vegas, pp. 25-28.

[29] Gligoroski D., Markovski S., Kocarev Lj.: *New Directions in Coding: From statistical Physics to Quasigroup String Transformations*, 2004 Inter. Symp. on Nonlinear Theory and Applc. (NOLAT2004), Fukuoka, Japan, Nov. 29- Dec. 3, (2004), pp. 545-548.

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

[31] Gligoroski D., Markovski S., Kocarev L., Gusev M.: *Edon80 - Hardware Synchronous stream cipher*, SKEW 2005 - Symmetric Key Encryption Workshop, May 2005, Aarhus Denmark.

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

[33] Hoffman D. G., Leonard D. A., Lindner C. C., Phelps K.T., Rodger C. A., Wall J. R.: *Coding Theory, The Essentials*, Auburn University, Auburn, Alabama, 1992.

[34] Huffman W. C., Pless V.: *Fundamentals of Error Correcting Codes*, Cambridge University Press, 2003.

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

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

[37] Lin S., Costello D. J.: *Error Control Coding: Fundamentals and Applications*, Prentice-Hall, Inc. Englewood Cliffs, New Jersey, 1983.

[38] MacKay D. J. C.: *Information Theory, Inference, and Learning Algorithms*, Cambridge University Press, 2003.

[39] Markovski J., Dimitrova V.: *Improving existing PRSG using QSP*, Fourth International for Informatics and Information Technology, 11-14 Dec. 2003, Bitola, Macedonia.

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

[41] Markovski S., Bakeva V.: *On a Stream Error Correcting Codes*, Proceedings of the 2nd CIIT, Bitola, Macedonia (2001), pp. 14-19.

[42] Markovski S., Bakeva V.: *On Error-Detecting Codes Based on Quasigroup Operations*, CIIT, Molika, (2003), pp. 400-405.

[43] Markovski S., Bakeva V.: *Error-Detecting Codes with Cyclically Defined Redundancy*, Зборник на трудови од III Конгрес на математичарите и информатичарите на Македонија, Струга, (2006), pp. 485-492.

[44] Markovski S., Bakeva V.:*Quasigroup String Processing: Part* 4, Contributions, Sec. Math. Tech.l Sci., MANU, XX 1-2, Skopje, Macedonia (2006-2007), pp.41-53.

[45] Markovski S., Dimitrova V., Samardziska S.: *Identities sieves for quasigroups*, Quasigroups and Related Systems, Vol. 18, No. 2, (2010) pp. 149-164.

[46] 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.

[47] 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.

[48] Markovski S., Gligoroski D., Bakeva V.: *Random walk tests for pseudo random number generators*, Mathematical Communications, Vol. 6, No.2, Osijek, Croatia (2001), pp. 135-143.

[49] Markovski S., Gligoroski D., Bakeva V.: *Quasigroup Hash Functions*, Proceedings of Workshop of Education of Informaticians and Industrial Mathematicians: New Challenges and Needs, Ohrid, Macedonia, 2001, pp. 123-131.

[50] 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.

[51] 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.

[52] 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.

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

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

[55] Markovski S., Mileva A.: *Generating huge quasigroups from small nonlinear bijections via extended Feistel function*, Quasigroups and Related Systems 17, 91-106, 2009.

[56] 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.

[57] Mathur C. N., Narayan K., Subbalakshmi i K.P: *High Diffusion Codes: A Class of Maximum Distance Separable Codes for Error Resilient Block Ciphers* 2nd IEEE International Workshop on Adaptive Wireless Networks (AWiN), Globecom, (2005).

[58] McEliece R.J.: *A Public-Key Cryptosystem Based on Algebraic Coding Theory*, DSN Progress Report, Jet Propulsion Laboratory, Calif., (1978), pp. 114–116.

[59] McKay B. D.: *Latin squares*, A Web Resourse $http : //cs.anu.edu.au /bdm/data/latin.html$.

[60] McKay B., Meynert A., Myrvold W: *Small Latin Squares, Quasigroups and Loops*, J. Combinatorial Designs 15, 2007, 98-119.

[61] Menezes A. J., P. C. Van Orchort P. C, Vanstone S. A.: *Handbook of Applied Cryptography*, CRC Press, 1997.

[62] Mileva A., Markovski S.: *Quasigroups String Transformations and Hash Function Design. A Case Study: The NaSHA Hash Function*, ICT Innovations conference 2009, Ohrid, pp 367-376.

[63] Moreira J. C., Farrell P. G.: *Essentials of Error-Control Coding*, John Wiley and Sons, Ltd, 2006.

[64] Nanjunda C., Haleem M., Chandramouli R.: *Robust Encryption for Secure Image Transmission over Wireless Channels*, IEEE International Conference on Communications (ICC 2005), 2005, Seoul, Korea, pp. 1287-1291.

[65] Pless V. S. and Huffman W. C. (editors): *Handbook of Coding Theory*, Elsevier Science B.V., Amsterdam, The Netherlands, 1998.

[66] Papoulis A.: *Probability, Random Variables and Stochastic Processes*, McGraw-Hill, Inc., New York, 1965.

[67] 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.

[68] 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.

[69] 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.

[70] 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.

[71] 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, pp. 178-181.

[72] Popovska-Mitrovikj A., Markovski S., Bakeva V.: *Error-correcting codes with cryptographic algorithms*, Proceedings of 21st Telecommunications Forum (TELFOR), 2013, Belgrade, Serbia, pp. 327-330, $http$: $//ieeexplore.ieee.org/xpl/articleDetails.jsp?tp = \&arnumber = 6716236\&queryText\%3DPopovska - Mitrovikj.$

[73] Popovska-Mitrovikj A., Markovski S., Bakeva V.: *4-Sets-Cut-Decoding algorithms for random codes based on quasigroups*, submitted to Advances in Mathematics of Communications (AMC)

[74] Popovska-Mitrovikj A., Mechkaroska D., Bakeva V.:, *Applying error-correcting codes based on quasigroups for image coding*, Proceedings of the 11th International Conference on Informatics and Information Technology, Faculty of Computer Science and Engineering, University "Ss.Cyril and Methodius" (Macedonia), Bitola, Macedonia, April 2014, (in print).

[75] Purser M.: *Introduction of Error-Correcting Codes*, Artech House, Boston, 1995.

[76] Raaphorst S.: *Reed-Muller Codes*, Carleton University, 2003.

[77] 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.

[78] Sade A.: *Quasigroupes parastrophiques. Expressions et identites*, Math. Nachr. 20, 1959, 73 - 106.

[79] Satti M.: *A Quasigroup Based Cryptographic System*, arXiv:cs/0610017v1 [cs.CR], 2006.

[80] Schneier B.: *Applied Cryptography*, John Wiley and Sons, Inc, 1996.

[81] Smith J. D. H.: *An introduction to quasigroups and their representations*, Academic Press, Inc., 1974.

[82] Stein S. K.: *On the foundations of quasigroups*, Trans. Amer. Math. Soc. 85, 1957, 228-256.

[83] Stinson D.R.: *Cryptography: Theory and Practice*, CRC Press, Inc., 1995.

[84] Tzonelih H., Rao T.R.N.: *Secret error-correcting codes*, Goldwasser, R. (Ed.): Advances in Cryptology - CRYPTO '88, LNCS 403, (1990), pp.540-563.

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

[86] Бакева В.: *Прилог кон примената на веројатносни модели во системите за масовно опслужување, криптографијата и кодирањето*, Докторска дисертација, Универзитет "Св. Кирил и Методиј", Скопје, 2002.

[87] Димитров Б.: Вериги на Марков, Издателство "Наука и искуство", Софија, 1974.

[88] Димитрова В.: *Квазигрупно процесирани низи, нивна булова презентација и примена во криптографијата и кодирањето*, Докторска дисертација, Универзитет "Св. Кирил и Методиј", Скопје, 2010.

[89] Поповска-Митровиќ А.: *Споредба на перформансите на случајните кодови базирани на квазигрупи и кодовите на Рид-Милер и Рид-Соломон*, магистерски труд, Универзитет "Св. Кирил и Методиј", Скопје, 2009.