Quasigroups and Related Systems **24** (2016), 249 − 268

Cryptcodes Based on Quasigroups in Gaussian channel

Daniela Mechkaroska, Aleksandra Popovska-Mitrovikj, Verica Bakeva

Abstract. Cryptcodes based on quasigroups transformation, known as Random Codes Based on Quasigroups (RCBQ) are error-correcting codes defined by using a cryptographic algorithm during the encoding/decoding process. Therefore, they allow not only correction of certain amount of errors in the input data, but they also provide an information security, all built in one algorithm. Standard and Cut-Decoding algorithms for these codes are defined elsewhere. Also, performances of these codes are investigated elsewhere, when a transmission through a binary symmetric channel is used. In this paper, we investigate the performances of RCBQ for transmission through Gaussian channel. We analyze the influence of the code parameters on the performances of RCBQ for code (72,288) with rate 1/4. We present and compare several experimental results obtained with different coding/decoding algorithms for these codes.

1. Introduction

Random Codes Based on Quasigroups (RCBQ) considered in this paper are cryptcodes. In order to provide an information security cryptcodes include an application of some of the known ciphers on codewords, before sending them through an insecure channel ([11, 12]). Usually, in the design of these codes two algorithms are used, one for error-correcting and another for obtaining information security. In the paper [4] authors give one algorithm where a block cipher and an error-correcting code are combined. But, the main application of their design is for cryptographic purposes, although it can be used as an error-correcting code.

RCBQs are proposed in [2] and they are defined by using a cryptographic algorithm during the encoding/decoding process. They allow not only correction of certain amount of errors in the input data, but they also provide an information security, all built in one algorithm. Therefore, these codes are interesting for further investigation. The influence of the code parameters on the performances of these codes are investigated in [6]. In [5] authors compare the performances of RCBQ with Reed-Solomon and Reed-Muller codes. From the results for packet-error and bit-error probabilities given there, authors concluded that RCBQ outperforms Reed-Muller and Reed-Solomon codes significantly for $p \geqslant 0.05$ in binary

2010 Mathematics Subject Classification: 94B35, 94B60, 68P30, 20N05.
Keywords: quasigroup, random code, error-correcting code, cryptcoding, packet-error probability, bit-error probability.

symmetric channel. But, the time efficiency of RCBQ is much lower than time efficiency of these two popular codes. In order to improve the decoding speed and other performances of RCBQ, in [7, 9] authors proposed new coding/decoding algorithms. In all papers for RCBQs, transmission through a binary symmetric channel is considered. Here, we investigate performances of these codes for transmission through Gaussian channel where the noise is a random variable with normal $N(0, N_0)$ distribution. In this channel, the probability of bit-error (see [10]) is given by

$$P_b = \frac{1}{2} erfc\left(\sqrt{\frac{E_b}{N_0}}\right), \tag{1}$$

where E_b is a power constraint.

RCBQs are designed using algorithm for encryption/decryption from the implementation of TASC (Totally Asynchronous Stream Ciphers) by quasigroup string transformation ([1]). These cryptographic algorithms use the alphabet Q and a quasigroup operation $*$ on Q together with its parastrophe " $\backslash$ ". The notions of quasigroups and quasigroup string transformations are given in the previous papers for these codes ([5], [6]). Here, we are using the same terminology and notations as there. Note that in this paper we consider only ability of RCBQ for corrections of errors in the transmitted data. The provided information security is guaranteed from the used quasigroup string transformation E given in [3].

The rest of this paper is organized as follows. In Section 2, we will briefly repeat the coding/decoding algorithms of RSBQ. In Section 3 we explain how the experiments are made. The influence of the code parameters on the performances of RCBQ with Standard algorithm is investigated in Section 4. In Section 5, we present experimental results obtained with Cut-Decoding algorithm and we compare these results with the best results for Standard algorithm. In order to improve the performances of Cut-Decoding algorithm for transmission through Gaussian channel, in Section 6, we define two combinations of the proposed methods for decreasing the number of unsuccessful decodings. At the end, we give some conclusions.

2. Description of RCBQ

Description of coding with Standard and Cut-Decoding algorithms

Let $M = m_1 m_2 \dots m_l$ be a block of $N_{block} = 4l$ bits where $m_i \in Q$ and Q is an alphabet of 4-bit symbols (nibbles). First, we add a redundancy as zero symbols and produce message

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

of $N = 4m$ bits ($m = rs$), where $L_i \in Q$, $L^{(i)}$ are sub-blocks of r symbols from Q. After erasing the redundant zeros from each $L^{(i)}$, the message L will produce the original message M. In this way we obtain (N_{block}, N) code with rate $R = N_{block}/N$. The codeword is produced after applying the encryption algorithm of TASC (given in Figure 1) on the message L. For this aim, a key $k = k_1k_2...k_n \in Q^n$ should be chosen. The obtained codeword of M is

$$C = C_1C_2...C_m,$$

where $C_i \in Q$.

Encryption	**Decryption**
Input: Key $k = k_1k_2 \ldots k_n$ and $L = L_1L_2 \ldots L_m$ **Output**: codeword $C = C_1C_2...C_m$	**Input**: The pair $(a_1a_2 \ldots a_r, k_1k_2 \ldots k_n)$ **Output**: The pair $(c_1c_2 \ldots c_r, K_1K_2 \ldots K_n)$

```
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 $r - 1$
    $X, T \leftarrow a_{j+1}$;
    $temp \leftarrow K_n$;
    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: $(c_1c_2 \ldots c_r, K_1K_2 \ldots K_n)$
```

Figure 1: Algorithms for encryption and decryption

In Cut-Decoding algorithm, instead of using (N_{block}, N) code with rate R, we use together two $(N_{block}, N/2)$ codes with rate $2R$ for coding/decoding the same message of N_{block} bits. Namely, for coding we apply two times the encryption algorithm, given in Figure 1, 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.

Description of decoding with Standard and Cut-Decoding algorithm

After transmission through a noise channel (for our experiments we use Gaussian channel), the codeword C will be received as message $D = D^{(1)} D^{(2)} \ldots D^{(s)} = D_1D_2 \ldots D_m$ where $D^{(i)}$ are blocks of r symbols from Q and $D_i \in Q$. The decoding process consists of four steps: (i) procedure for generating the sets with predefined Hamming distance, (ii) inverse coding algorithm, (iii) procedure for generating decoding candidate sets and (iv) decoding rule.

Let B_{max} be a given integer which denotes the asumed maximum number of errors occur in a block during transmission. The probability that at most t bits in D^i are not correctly transmitted is

$$P(P_b;t) = \sum_{k=0}^{t} \binom{4r}{k} P_b^k (1-P_b)^{4r-k},$$

where P_b is probability of bit-error in a Gaussian channel. Then $P(P_b;B_{max})$ is the probability that at most B_{max} errors occur in a block during transmission. We generate the sets $H_i = \{\alpha | \alpha \in Q^r, \ \ H(D^{(i)},\alpha) \leqslant B_{max}\}$, for $i = 1,2,\dots,s$, where $H(D^{(i)},\alpha)$ is the Hamming distance between $D^{(i)}$ and α.

The decoding candidate sets S_0, S_1, S_2,…,S_s, are defined iteratively. Let $S_0 = (k_1 \dots k_n; \lambda)$, where λ is the empty sequence. Let S_{i-1} be defined for $i \geqslant 1$. Then S_i is the set of all pairs $(\delta, w_1 w_2 \dots w_{4ri})$ obtained by using the sets S_{i-1} and H_i as follows (w_j are bits). For each element $\alpha \in H_i$ and each $(\beta, w_1 w_2 \dots w_{4r(i-1)}) \in S_{i-1}$, we apply the inverse coding algorithm (i.e., algorithm for decryption given in Figure 1) 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 $(\delta, w_1 w_2 \dots w_{4r(i-1)} c_1 c_2 \dots c_r) \equiv (\delta, w_1 w_2 \dots w_{4ri})$ ($c_i \in Q$) is an element of S_i.

The decoding of the received message D is given by the following rule: If the set S_s contains only one element

$$(d_1 \dots d_n, w_1 \dots w_{4rs}),$$

then $L = w_1 \dots w_{4rs}$ is the decoded (redundant) message and we say that we have a *successful decoding*. In the case when the set S_s contains more than one element then the decoding of D is unsuccessful and we say a *more-candidate-error* appears. In the case when $S_j = \emptyset$ for some $j \in \{1,\dots,s\}$, the process will be stopped and we say that a *null-error* appears.

In Cut-Decoding algorithm, after transmitting through a noise channel, we divide the outgoing message $D = D^{(1)} D^{(2)} \dots D^{(s)}$ in two messages $D_1 = D^{(1)} D^{(2)} \dots D^{(s/2)}$ and $D_2 = D^{(s/2+1)}\ D^{(s/2+2)} \dots D^{(s)}$ with equal lengths and we decode them parallel with the corresponding parameters. In this decoding algorithm we make modification in the procedure for generating decoding candidate sets. Let $S_i^{(1)}$ and $S_i^{(2)}$ be the decoding candidate sets obtained in the i^{th} iteration of the two parallel decoding processes, $i = 1,\dots,s$. Then, before the next iteration we eliminate from $S_i^{(1)}$ all elements whose second part does not match with the second part of an element in $S_i^{(2)}$, and vice versa. In the $(i+1)^{th}$ iteration the both processes use the corresponding reduced sets $S_i^{(1)}$ and $S_i^{(2)}$. With Cut-Decoding algorithm the decoding speed is improved and the values of the packet-error probability (PER) and the bit-error probability (BER) for code $(72, 288)$ are smaller.

3. Experiments

The experiments with the random codes based on quasigroup are made on a high performance claster on Faculty of Computer Science and Engineering, UKIM - Skopje. The cluster has 24 GB RAM, a processor with 2.266 GHz and 12 physical cores (24 logical cores) are used.

The experiments are made in the following way:

- Firstly, we extend the message obtained from the source using a pattern for adding redundant zero nibbles. We made experiments with 6 different patterns.
- The extended message is coded using an algorithm for coding (Standard or Cut-Decoding) and blocks of 4 nibbles.
- On the coded message we make $BPSK$ modulation ($0 \to -1$ and $1 \to 1$).
- The signal is transmitted through Gaussian channel and due to the noises, the received output signal can be different from the input signal.
- Then, we make demodulation on the output signal in the following way:
  1) if the received signal is greater than 0, then the receiver assumes that bit 1 was transmitted;
  2) if the received signal is less than 0, then the receiver assumes that bit 0 was transmitted.
- The demodulated message is decoded with the corresponding (previously defined) decoding algorithm.
- We compare the decoded message with the input message and compute BER and PER for different values of SNR in the interval from -3 to 10.

The packet-error probability PER is computed as a ratio of the number of incorrectly decoded packets (messages) and the number of all packets. The incorrectly decoded packets appear in the following cases:

1. If the last decoding candidate set S_s has only one element, then the message in that element (the decoded message) is compared with the input message. If both are equal then we have a correct decoding. If the decoded message differs in at least one bit then we have an uncorrected error.
2. Packet errors appear in other cases of unsuccessful decoding (*more-candidate errors* and *null-errors*).

The bit-error probability BER is computed as a ratio of the number of incorrectly decoded bits and the number of all bits. The incorrectly decoded bits appear in the following cases:

1. When the decoding is successful, we compare the decoded and the input message computing Hamming distance between them. It gives the number of incorrectly decoded bits.

2. When *a null-error* appears, i.e., $S_i = \emptyset$ for some $0 \leqslant j \leqslant s$, we take all elements from the set S_{i-1} and we find their maximal common prefix substring. If this string has k bits and the length of the input message is $4l$ bits then we compare this substring with the first k bits of the input message. If they differ in t bits then the number of incorrectly decoded bits is $4l - k + t$.

3. If *a more-candidate-error* appears we take all elements from the set S_s and we find their maximal common prefix substring. The number of incorrectly decoded bits is computed as previously.

4. Experimental results for Standard algorithm

In this section we present and analyze the results obtained using Standard coding/decoding algorithm for RCBQ. We investigate the influence of the code parameters on the code performances.

The influence of the pattern on the code performances

In order to check the influence of the pattern on the code performances, we made experiments with 6 different patterns for redundant zero nibbles for code $(72, 288)$ with rate $R = 1/4$. This means that the alphabet

$$Q = \{0, 1, 2, 3, 4, 5, 6, 8, 9, \mathrm{a}, \mathrm{b}, \mathrm{c}, \mathrm{d}, \mathrm{e}, \mathrm{f}\}.$$

In these experiments we have used the quasigroup $(Q, *)$ and its parastrophe $(Q, \backslash)$ given in Table 1, the initial key $k = 0123456789$ and 6 patterns given in Table 2. In the patterns, we denote the message (information) symbol with 1 and the redundant zero symbol with 0. The experiments for different values of SNR in the interval from -3 to 10 dB are made. In this section we present the experimental results for bit-error probability (BER) and packet-error probability (PER).

Firstly, we made experiments for $B_{max} = 3$ with 13888 messages. The obtained results for BER are given in Table 3 and presented in Figure 2, while the appropriate values of PER are presented in Table 4 and Figure 3.

The experimental results for BER are compared with a probability P_b for bit-error in Gaussian channel. It is obvious that the values of BER and PER increase as the values of SNR decrease (smaller values of SNR mean larger noise). Therefore, we made experiments starting from $SNR = 10$ and decreasing the values of SNR by 1. We stoppped with experiments when we get $BER > P_b$. In this case the codes does not have sense since the bit-error probability obtained using the code is greater than the bit-error probability P_b without coding. All experimental results for BER obtained using pattern 1 and pattern 5 were greater

*	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

\	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 1: Quasigroup of order 16 used in the experiments and corresponding parastrophe

pattern 1	**pattern 2**	**pattern 3**	**pattern 4**	**pattern 5**	**pattern 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: Patterns for redundant zero nibbles

than P_b and therefore these results are not present in the following tables and figures.

From all experimental results given in the tables and figures we can see that when the value of SNR increases, the bit-error and packet-error probabilities decrease.

SNR	**pattern 2**	**pattern 3**	**pattern 4**	**pattern 6**
1	0.08637	0.08737	0.10262	0.08838
2	0.02054	0.02075	0.02418	0.02057
3	0.00387	0.00408	0.00398	0.00321
4	0.00055	0.00046	0.00030	0.00016
5	0	0	0.00007	0
6	0	0	0	0.00004
7	0	0	0	0
8	0	0	0	0
9	0	0	0	0
10	0	0	0	0

Table 3: Experimental results for BER for different patterns and $B_{max} = 3$

From Figure 2 and Figure 3 we can notice that none of the considered patterns

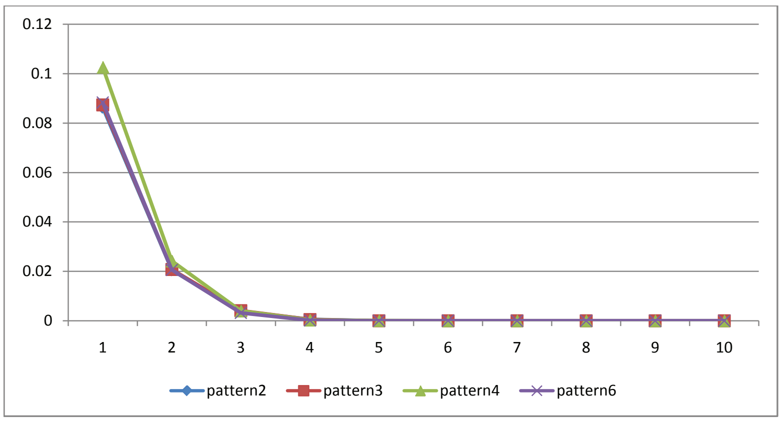

Figure 2: Experimental results for BER for different patterns and $B_{max} = 3$

SNR	pattern 2	pattern 3	pattern 4	pattern 6
1	0.176915	0.17519	0.171659	0.175403
2	0.043707	0.04435	0.040971	0.043131
3	0.007488	0.00801	0.007056	0.007272
4	0.001008	0.00086	0.000648	0.000432
5	0	0	0.00007	0
6	0	0	0	0.00007
7	0	0	0	0
8	0	0	0	0
9	0	0	0	0
10	0	0	0	0

Table 4: Experimental results for PER for different patterns and $B_{max} = 3$

stands out as the best or as the worst, i.e., for all patterns the values of BER (PER) are very close. In these experiments we did not obtain any unsuccessful decoding with *more-candidate-error*. So, all unsuccessful decodings are *null-errors*, whose number decreases when SNR increases.

Further on, we made experiments with the same parameters, but for $B_{max} = 4$. The decoding process in these experiments is much slower than for $B_{max} = 3$ since the number of elements in the sets S_i are greater. The obtained results for BER are presented in Table 5 and Figure 4, and the results for PER in Table 6 and Figure 5.

From Figure 4, we can see that pattern 6 is the worst pattern for this value of B_{max}, while the other patterns give almost equal values of BER (the differences are in the fourth decimal). Only for smaller values of SNR pattern 3 gives better results than pattern 2 and pattern 4.

As we concluded before, for $B_{max} = 3$, *more-candidate-error* does not appear with none of the patterns. But, from the experiment for $B_{max} = 4$ we can conclude the following:

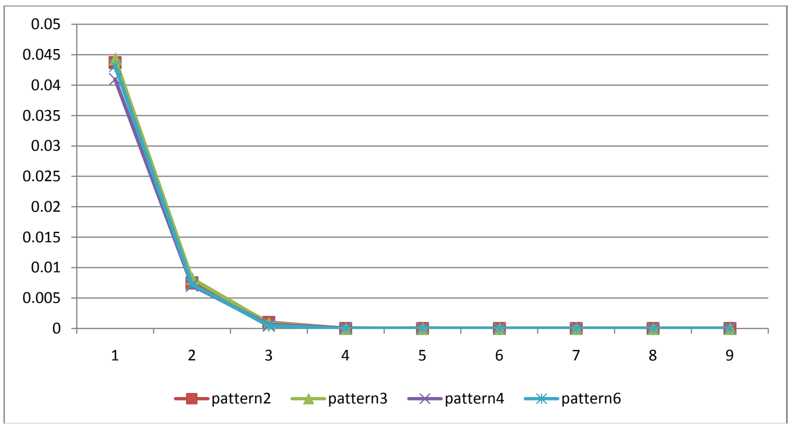

Figure 3: Experimental results for PER for different patterns and $B_{max} = 3$

SNR	pattern 2	pattern 3	pattern 4	pattern 6
0	0.10582	0.08343	0.09630	0.94627
1	0.02411	0.01928	0.02336	0.24562
2	0.00361	0.00400	0.00424	0.04602
3	0.00049	0.00068	0.00047	0.00788
4	0.00024	0.00042	0.00014	0.00202
5	0.00004	0.00066	0.00016	0.00136
6	0.00032	0.00054	0.00016	0.00374
7	0.00025	0.00046	0.00011	0.00354
8	0.00025	0.00053	0.00004	0.00372
9	0.00025	0.00053	0.00004	0.00372
10	0.00025	0.00053	0.00004	0.00372

Table 5: Experimental results for BER for different patterns and $B_{max} = 4$

SNR	pattern 2	pattern 3	pattern 4	pattern 6
0	0.10750	0.10829	0.10808	0.10707
1	0.02448	0.02520	0.02635	0.02758
2	0.00367	0.00554	0.00482	0.00511
3	0.00050	0.00137	0.00065	0.00086
4	0.00029	0.00101	0.00014	0.00029
5	0.00007	0.00144	0.00004	0.00014
6	0.00036	0.00115	0.00029	0.00043
7	0.00029	0.00108	0.00022	0.00043
8	0.00029	0.00122	0.00001	0.00043
9	0.00029	0.00122	0.00001	0.00043
10	0.00029	0.00122	0.00001	0.00043

Table 6: Experimental results for PER for different patterns and $B_{max} = 4$

- for $SNR \leqslant 3$, we obtain more *null-errors* than *more-candidate-errors*;
- for $SNR > 3$, we do not have *null-errors*, but we have *more-candidate-errors*.

In order to reduce the number of unsuccessful decodings with *more-candidate-error*, we use the heuristic introduced in [7] in the experiments with $B_{max} = 4$.

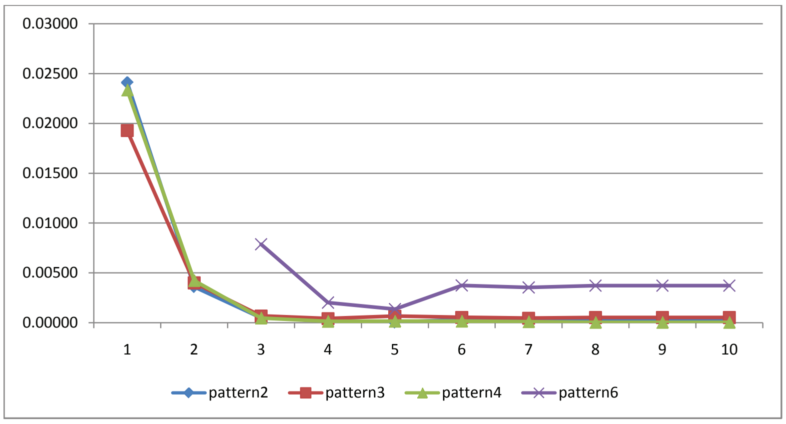

Figure 4: Experimental results for BER for different patterns and $B_{max} = 4$

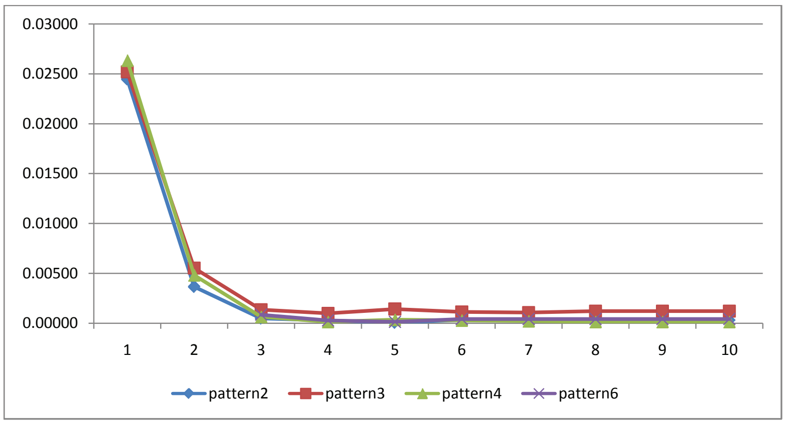

Figure 5: Experimental results for PER for different patterns and $B_{max} = 4$

According to this heuristic, in the case of *more-candidate-error* we randomly select a message from the last decoding candidate set and it is taken as decoded message. The results for BER and PER obtained using the heuristic are presented in Table 7 and Table 8 (Figure 6 and Figure 7), correspondingly. From the results, we can conclude that using this heuristic, the values of BER and PER are slightly better.

From the previous experiments, we can conclude that the chosen pattern has a great influence on the code performances. For some patterns (for example, **pattern 1** and **pattern 5** given in Table 2) the coding does not have a sense.

SNR	pattern 2	pattern 3	pattern 4	pattern 6
0	0.10582	0.08301	0.09609	0.94627
0.5	0.05395	0.04187	0.04913	0.48607
1	0.02397	0.01919	0.02336	0.24562
1.5	0.01169	0.00923	0.00960	0.08374
2	0.00361	0.00368	0.00424	0.04602
3	0.00042	0.00052	0.00047	0.00788
4	0.00011	0.00028	0.00014	0.00202
5	0.00004	0.00031	0.00016	0.00136
6	0.00025	0.00025	0.00016	0.00374
7	0.00018	0.00032	0.00011	0.00354
8	0.00018	0.00035	0.00004	0.00372
9	0.00018	0.00035	0.00004	0.00372
10	0.00018	0.00035	0.00004	0.00372

Table 7: Experimental results for BER for $B_{max} = 4$ using the heuristic for decreasing of *more-candidate-errors*

SNR	pattern 2	pattern 3	pattern 4	pattern 6
0	0.10750	0.10829	0.10808	0.10707
1	0.02434	0.02506	0.02635	0.02758
2	0.00367	0.00490	0.00482	0.00511
3	0.00043	0.00058	0.00065	0.00086
4	0.00014	0.00058	0.00014	0.00029
5	0.00007	0.00065	0.00036	0.00014
6	0.00029	0.00043	0.00029	0.00043
7	0.00022	0.00058	0.00022	0.00043
8	0.00022	0.00065	0.00014	0.00043
9	0.00022	0.00065	0.00014	0.00043
10	0.00022	0.00065	0.00014	0.00043

Table 8: Experimental results for PER for $B_{max} = 4$ using the heuristic for decreasing of *more-candidate-errors*

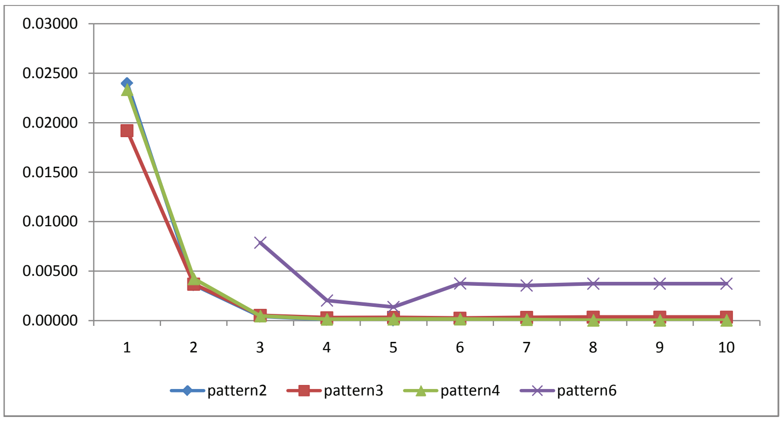

Figure 6: Experimental results for BER for $B_{max} = 4$ using the heuristic for decreasing of *more-candidate-errors*

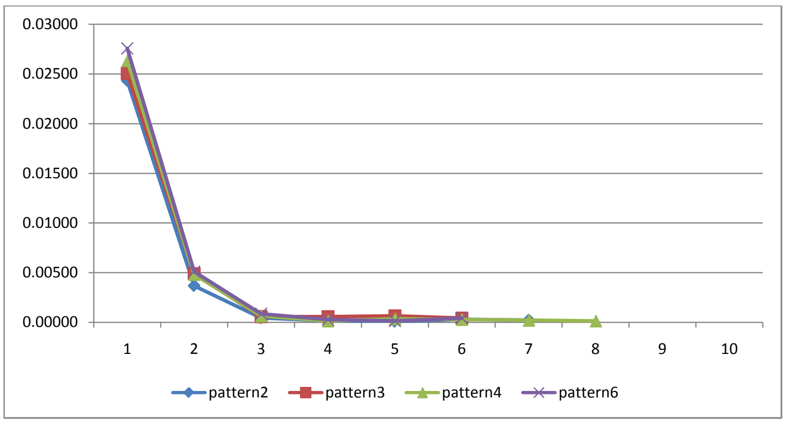

Figure 7: Experimental results for PER for $B_{max} = 4$ using the heuristic for decreasing of *more-candidate-errors*

Influence of the key length to the code performances

In order to check the influence of the key length to the code performances, we have made experiments using keys with different lengths. The results obtained for key lengths of 5, 10 and 15 nibbles are represented in Table 9, Figure 8 (for BER) and Table 10, Figure 9 (for PER).

SNR	key length 5	key length 10	key length 15
1	0.02838	0.01928	0.02090
2	0.01135	0.00400	0.00401
3	0.01026	0.00068	0.00091
4	0.00899	0.00042	0.00041
5	0.00952	0.00066	0.00032
6	0.00910	0.00054	0.00081
7	0.00931	0.00046	0.00067
8	0.00922	0.00053	0.00067
9	0.00927	0.00053	0.00066
10	0.00929	0.00053	0.00066

Table 9: Experimental results for BER for key lengths of 5, 10 and 15 nibbles

Analyzing the presented results we can see that results obtained using keys with length 10 and 15 are almost identical. Also, in these cases, the decoding speeds are same. In the experiments with a key of length 5, the worse results for BER and PER are obtained. For this key length and $SNR \geqslant 3$, the bit-error probability is about 15 to 20 times greater than the corresponding probability obtained with the key length 10, and it is up to 29 times greater than the probability obtained with the key length 15.

From the experiments we can conclude that the key length also has a great influence on the performances of these codes.

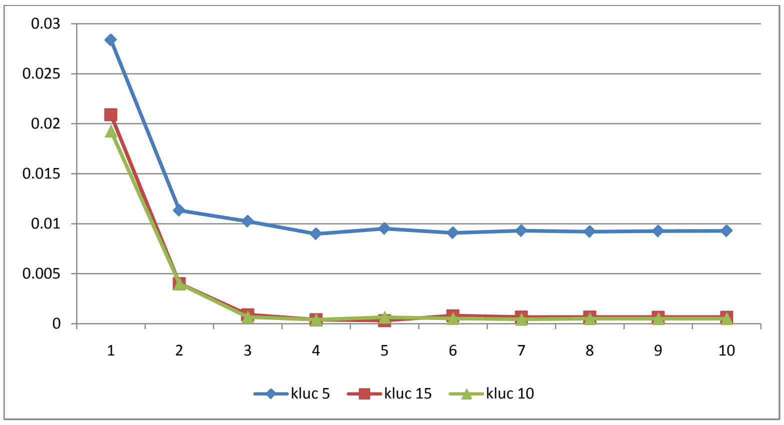

Figure 8: Experimental results for BER for key lengths of 5, 10 and 15 nibbles

SNR	key length 5	key length 10	key length 15
1	0.03816	0.02520	0.02772
2	0.01533	0.00554	0.00583
3	0.01360	0.00137	0.00151
4	0.01202	0.00101	0.00093
5	0.01245	0.00144	0.00079
6	0.01216	0.00115	0.00144
7	0.01252	0.00108	0.00129
8	0.01245	0.00122	0.00129
9	0.01245	0.00122	0.00129
10	0.01250	0.00122	0.00129

Table 10: Experimental results for PER for key lengths of 5, 10 and 15 nibbles

Influence of the chosen quasigroup to the code performances

In order to check the influence of the choice of a quasigroup on the code performances, we have made experiments with a cyclic quasigroup of order 16 using a key of 10 nibbles. Firstly, the experiments were made by using the third pattern. But, in these experiments we obtained a great number of messages in the decoding candidate sets. So, the decoding process was very slow and it did not finish in reasonable time.

Therefore, we made experiments using the first pattern and $B_{max} = 4$. The decoding was faster than in the previous case, but not enough. Also, we obtained a great number of *more-candidate-errors*. For example, for $SNR = 1$, the probability for this type of error is 0.98.

From these experiments, we can conclude that the choice of the quasigroup has an enormous influence on the performances of these codes.

From all experimental results obtained with Standard algorithm for coding/decoding messages transmitted through a Gaussian channel, we can conclude that the best results are obtained using the third pattern, the key length equal to 10

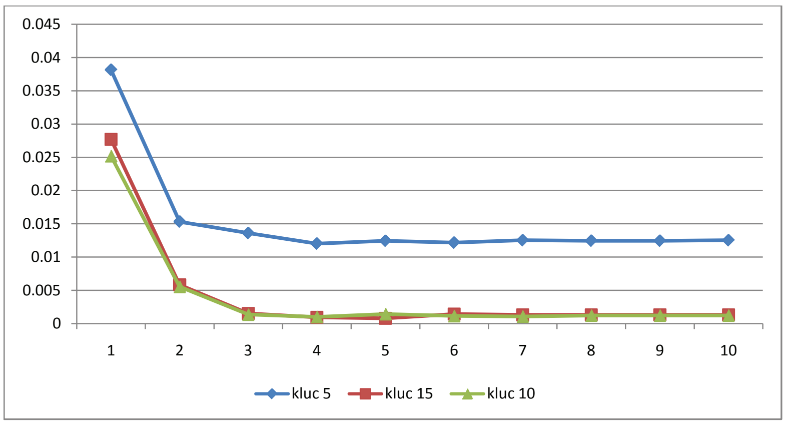

Figure 9: Experimental results for PER for key lengths of 5, 10 and 15 nibbles

(or 15), the quasigroup given in Table 1 and $B_{max} = 4$.

5. Experimental results for Cut-Decoding algorithm

In the experiments made with Cut-Decoding algorithm, in both processes of coding/decoding we used a same quasigroup and different keys. The best results are obtained for the following parameters:

- redundancy pattern: 1100 1110 1100 1100 1110 1100 1100 1100 0000;
- two different keys with length 5: $k_1 = 01234$ and $k_2 = 56789$;
- the quasigroup given in Table 1;
- $B_{max} = 4$.

Also, experiments with keys of 10 nibbles were made, but the results were similar.

Let PER_c be the packet-error probability and BER_c the bit-error probability obtained with Cut-Decoding algorithm. We will compare these probabilities with PER_s and BER_s obtained with Standard algorithm using the best parameters (pattern 3, quasigroup given in Table 1 and key $k = 0123456789$). In Table 11 and Figure 10, BER_s and BER_c for different values of SNR are given, and in Table 12 and Figure 11 - the corresponding results for PER_s and PER_c. In the tables we present the results for $SNR \geqslant 0$, since the decoding does not have sense ($BER > P_b$) for smaller values of SNR.

SNR	BER_s	BER_c
0	0.08343	0.07153
1	0.01928	0.01831
2	0.00400	0.00249
3	0.00068	0.00073
4	0.00042	0.00052
5	0.00066	0.00047
6	0.00054	0.00060
7	0.00046	0.00049
8	0.00053	0.00046
9	0.00053	0.00046
10	0.00053	0.00046

Table 11: Comparison of experimental results for BER for both algorithms

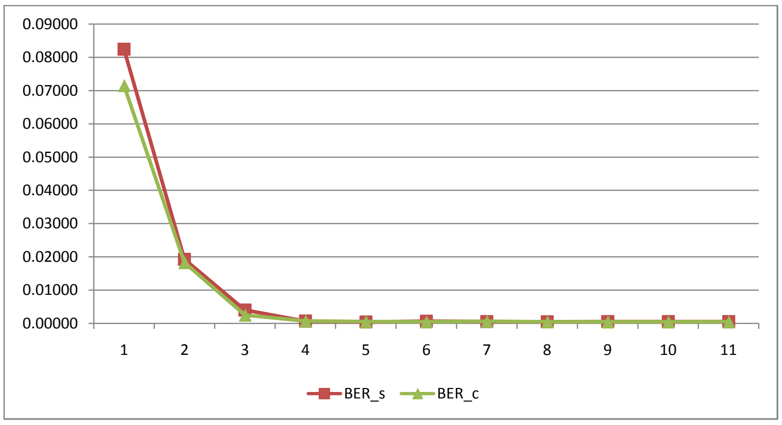

Figure 10: Comparison of BERs

From the results given in Table 11 and Table 12, we can conclude that for both algorithms the results for PER and BER are approximately equal (they differ in the third or higher decimal). But the decoding process with Cut-Decoding algorithm is about 4 times faster than with Standard algorithm.

SNR	PER_s	PER_c
0	0.10829	0.10001
1	0.02520	0.02722
2	0.00554	0.00410
3	0.00137	0.00230
4	0.00101	0.00252
5	0.00144	0.00230
6	0.00115	0.00252
7	0.00108	0.00238
8	0.00122	0.00223
9	0.00122	0.00223
10	0.00122	0.00223

Table 12: Comparison of experimental results for PER for both algorithms

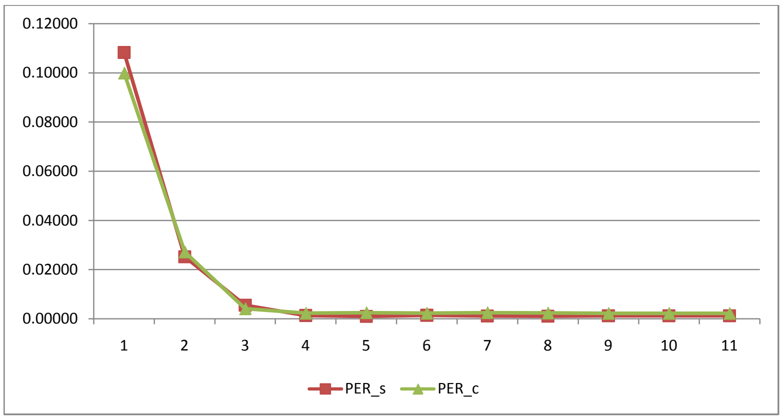

Figure 11: Comparison of PERs

Methods for decreasing of number of unsuccessful decodings with Cut-Decoding algorithm

In order to reduce the number of unsuccessful decodings with *null-error* and *more-candidate-error*, several modifications of Cut-Decoding algorithm are defined (in [7]). To improve the performances of Cut-Decoding algorithm (for transmission through Gaussian channel) we use the following two combinations of the proposed modifications.

In the both combinations with backtracking, if the decoding ends with *null-error*, then the last two iterations are canceled and the first of them is reprocessed with $B_{max} + 2 = 6$ (the next iterations use the previous value of B_{max}). If the decoding ends with *more-candidate-error*, then the last two iterations of the decoding process are canceled and the penultimate iteration is reprocessed with $B_{max} - 1 = 3$. In the decoding of a message only one backtracking is made, except when after the backtracking for *null-error*, *more-candidate-error* appears. In Combination 1, we make one more backtracking for *more-candidate-error* if both decoding candidate sets are non-empty. In Combination 2, we make one more backtracking for *more-candidate-error*, if at least one of decoding candidate sets is non-empty.

The bit-error and packet-error probabilities obtained with Combination 1 are denoted by BER_{c-back} and PER_{c-back}. The bit-error and packet-error probabilities obtained with Combination 2 are denoted by $BER_{c-back-2}$ and $PER_{c-back-2}$.

In Table 13 and Figure 12, we compare the values of packet-error probabilities PER_s, PER_c, PER_{c-back} and $PER_{c-back-2}$ and in Table 14 and Figure 13 - corresponding bit-error probabilities BER, BER_c, BER_{c-back} and $BER_{c-back-2}$.

SNR	PER_s	PER_c	PER_{c-back}	$PER_{c-back-2}$
0	0.10829	0.10001	0.07431	0.07496
1	0.02520	0.02722	0.01944	0.01743
2	0.00554	0.00410	0.00259	0.00331
3	0.00137	0.00230	0.00043	0.00036
4	0.00101	0.00252	0.00014	0.00014
5	0.00144	0.00230	0.00014	0.00007
6	0.00115	0.00252	0.00022	0.00014
7	0.00108	0.00238	0.00029	0.00029
8	0.00122	0.00238	0.00029	0.00029
9	0.00122	0.00223	0.00029	0.00029
10	0.00122	0.00223	0.00029	0.00029

Table 13: Comparison of experimental results for PER

SNR	BER_s	BER_c	BER_{c-back}	$BER_{c-back-2}$
0	0.08343	0.07153	0.04701	0.04815
1	0.01928	0.01830	0.01163	0.01137
2	0.00400	0.00249	0.00146	0.00227
3	0.00068	0.00073	0.00018	0.00013
4	0.00042	0.00052	0.00009	0.00010
5	0.00066	0.00047	0.00005	0.00003
6	0.00054	0.00060	0.00007	0.00006
7	0.00046	0.00049	0.00010	0.00010
8	0.00053	0.00046	0.00010	0.00010
9	0.00053	0.00046	0.00010	0.00010
10	0.00053	0.00046	0.00010	0.00010

Table 14: Comparison of experimental results for BER

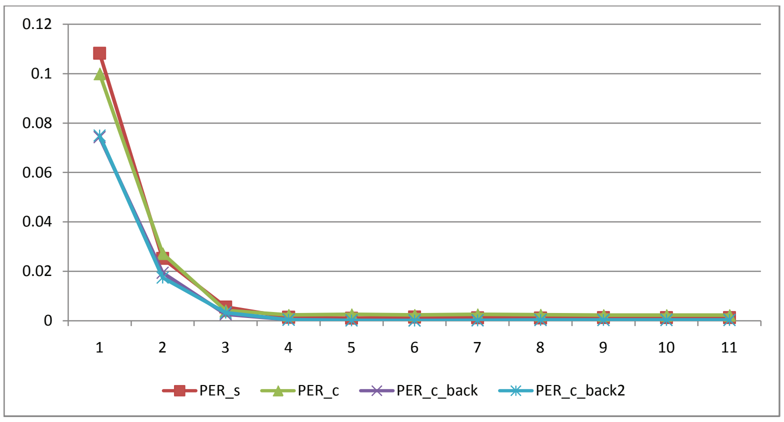

Figure 12: Comparison of PERs

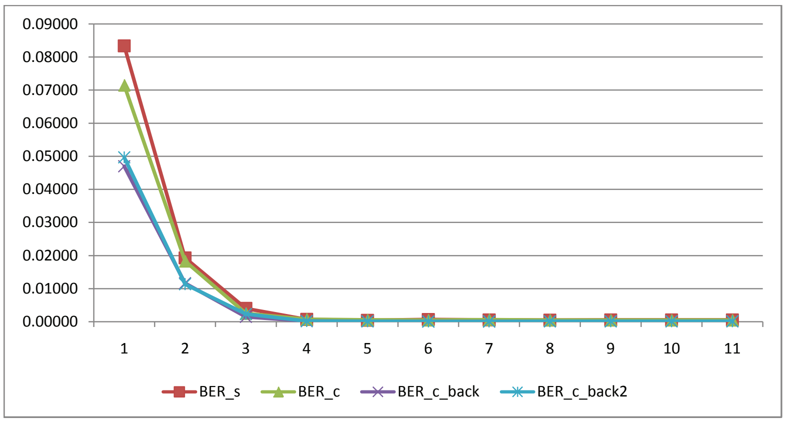

Figure 13: Comparison of BERs

Analyzing the results we can see that Cut-Decoding algorithm with the both proposed combinations for backtracking gives better (and almost identical) results than Cut-Decoding algorithm without backtracking and Standard algorithm.

Also, we calculate the percentage of the eliminated unsuccessful decodings by using the both combinations for backtracking. These results are given in Table 15.

We can notice that the obtained percentages of the eliminated unsuccessful decodings are same with both proposed combinations for backtracking, i.e, we have the same number of eliminated unsuccessful decodings. But, for $SNR \leqslant 2$, the first combination gives better elimination of *null-errors* and the second one -

SNR	% of elimination 1st combination	% of elimination 2st combination
0	22.75%	22.85%
1	35.19%	35.19%
2	35.09%	35.09%
3	84.38%	84.38%
4	80.00%	80.00%
5	90.63%	90.63%
6	91.43%	91.43%
7	87.88%	87.88%
8	87.10%	87.10%
9	83.87%	83.87%
10	87.10%	87.10%

Table 15: Percent of eliminated unsuccessful decodings

better elimination of *more-candidate-errors.*

Conclusions

From the experiments made for investigation of the performances of RCBQ for transmission through Gaussian channel, we can conclude that all parameters (a pattern for adding redundancy, a key length and a chosen quasigroup) have influence to the performances of these codes. Also, the pattern and the quasigroup have a great influence to the decoding speed. Namely, with inappropriate choice of these parameters, the decoding process does not finish in real time and the packet-error and bit-error probabilities are very large. Two combinations with backtracking of Cut-Decoding algorithm are proposed and they give a good percentage of eliminated unsuccessful decodings.

These conclusions for influence of parameters on the performances of RCBQ for transmission through Gaussian channel are very similar with corresponding conclusions for transmission through a binary symmetric channel (see, [6, 7, 8, 9]).

Acknowledgment

This research was partially supported by Faculty of Computer Science and Engineering at the University "Ss Cyril and Methodius" in Skopje.

References

[1] **D. Gligoroski, S. Markovski and Lj. Kocarev**. *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.

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

[3] **S. Markovski, D. Gligoroski and V. Bakeva**, *Quasigrouop string processing: Part* 1, Contributions, Sec. Math. Tech. Sci., MANU, **20** (1999), 13 – 28.

[4] **C.N. Mathur, K. Narayan and K.P. Subbalakshmi**, *High Diffusion Cipher: Encryption and Error Correction in a Single Cryptographic Primitive*, Lecture Notes Comput. Sci. **3989** (2006), 309 – 324.

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

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

[7] **A. Popovska-Mitrovikj, S. Markovski anf V. Bakeva**, *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.

[8] **A. Popovska-Mitrovikj, S. Markovski and V. Bakeva**, *Some New Results for Random Codes Based on Quasigroups*, Proc. 10th Conf. Informatics and Information Technology with International Participants, Bitola (2013), pp. 178 – 181.

[9] **A. Popovska-Mitrovikj, S. Markovski and V. Bakeva**, 4-*Sets-Cut-Decoding algorithms for random codes based on quasigroups*, Intern. J. Electronics Commun. **69** (2015), 1417 – 1428.

[10] **J.G. Proakis and M. Salehi**, *Digital Communications*, Fifth Edition, McGrawHill Higher Education (2008)

[11] **H. Tzonelih and T.R.N. Rao**, *Secret error-correcting codes*, Lecture Notes Comput. Sci. **403** (1990), 540 – 563.

[12] **N. Zivic and C. Ruland**, *Parallel Joint Channel Coding and Cryptography*, Intern. J. Electrical and Electronics Engineering, **4(2)** (2010), 140 – 144.

Received March 19, 2016

University "Ss Cyril and Methodius" - Skopje,
Faculty of Computer Science and Engineering,
P.O. Box 393, Republic of Macedonia
E-mails: daniela-mec@hotmail.com, aleksandra.popovska.mitrovikj@finki.ukim.mk, verica.bakeva@finki.ukim.mk