SOME NEW RESULTS FOR RANDOM CODES BASED ON QUASIGROUPS

Aleksandra Popovska-Mitrovikj* Smile Markovski* Verica Bakeva*
UKIM, Faculty of Computer Science and Engineering
{aleksandra.popovska.mitrovikj, smile.markovski, verica.bakeva}@finki.ukim.mk

Abstract

In this paper we continue with our work on Random codes based on quasigroups. Here, some modifications of the encoding algorithm, where quasigroups of order 4 and order 256 are used, instead of quasigroups of order 16. We made simulations for binary symmetric channel, for different quasigroups and different lengths of messages and we find the packet-error probability (PER) and the bit-error probability (BER). Also, we propose a method for reducing the more-candidate-errors.

Key words: error-correcting code, random code, packet-error probability, bit-error probability, quasigroup.

I. Introduction

Here we consider the codes defined in [5] and we investigate the performances of these codes using quasigroups of order 4 and order 256 in the processes of coding and decoding. For that aim we have modified the standard algorithm defined for nibbles to be used for 2-bit letters and bytes. We have made several experiments with the standard method of decoding, but also with the cut-decoding algorithm defined in [1]. A new method for reducing the more-candidate-errors is defined and some improvements are obtained.

The notions of quasigroups and quasigroup string transformations are given in the previous paper of ours [2]. Here, we are using the same terminology and notations as there. We will repeat only the encoding and decoding algorithms for more general case when a-bit letters instead of nibbles are used.

A. *Description of coding with standard algorithm*

Let $M = m_1m_2...m_r$ be a block of N_{block} bits, where $m_i \in Q$ and Q is an alphabet of a-bit symbols. First, we add redundancy as zero symbols and produce block $L = L^{(1)}L^{(2)}...L^{(s)} = L_1L_2...L_m$ of N bits, where $L^{(i)}$ are sub-blocks of r symbols from Q and $L_i \in Q$. After erasing the redundant zeros from each $L^{(i)}$, the message L will produce the original message M. On 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 1) on the block L. For that aim, previously, a key $k = k_1k_2\dots k_n \in Q^n$ should be chosen. The obtained codeword of M is $C = C_1C_2...C_m$, where $C_i \in Q$.

B. *Description of decoding with standard algorithm*

After transmission through a noise channel (for our experiments we use binary symmetric channel), the codeword C will be received as message $D = D^{(1)}D^{(2)}...D^{(s)} = D_1D_2...D_m$, 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.

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**: $(c_1c_2\dots c_s, K_1K_2\dots K_n)$

Figure1: Algorithm for encryption and decryption

The probability that $\leq t$ bits in $D^{(i)}$ 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. Let B_{max} be an integer such that $1 - P(p; B_{max}) \leq q_B$ and $H_i = \{\alpha | \alpha \in Q^r,\ H(D^{(i)}, \alpha) \leq B_{max}\}$, 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 $i \geq 1$. Then S_i is the set of all pairs $(\delta, w_1w_2\dots w_{r\cdot a\cdot i})$ obtained by using the sets S_{i-1} and H_i as follows (w_j are bits). For each $(\beta, w_1w_2\dots w_{r\cdot a\cdot (i-1)}) \in S_{i-1}$ and each element $\alpha \in H_i$, we apply the inverse coding algorithm (i.e. algorithm for decryption given in Figure 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 $(\delta, w_1w_2\dots w_{r\cdot a\cdot (i-1)}c_1c_2\dots c_{r\cdot a}) \equiv (\delta, w_1w_2\dots w_{r\cdot a\cdot i})$ is an element of S_i.

The decoding of the received codeword D is given by the following rule: If the set S_s contains only one element $(d_1 \dots d_n, w_1 \dots w_{r\cdot a\cdot s})$ then $L = w_1 \dots w_{r\cdot a\cdot s}$. In this case, we say that we have a *successful decoding*. 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-errors*). In the case when $S_j = \emptyset$ for some $j \in \{1, \dots, s\}$, the process will be stopped (*null-error* appears). We conclude that for some $i \leq j$, $D^{(i)}$ contains more than B_{max} errors, resulting with $C_i \notin H_i$.

Theorem 1: [5] The packet-error probability of these codes is $q = 1 - (1 - q_B)^s$.

C. *Description of cut-decoding algorithm*

In [1] we defined a new cut-decoding algorithm. In this algorithm, for coding we apply two transformations on the same redundant message using different parameters and obtain the codeword C as concatenation of two produced codewords. Therefore, we use two codes with rate 1/2 in order to obtain a code with rate 1/4. In the decoding process, we divide the received message D on two messages with same length and we apply the standard decoding algorithm on both sub-messages. After each iteration we obtain the decoding candidate set with elimination of all elements in the first decoding candidate set whose second part does not matches with the second part of an element in the second set, and vice versa. With this algorithm we obtained improvement in the speed of decoding and the values of PER and BER for code (72,288) using nibbles.

II. Experiments with quasigroups of order 4 and order 256

In [2], we investigated the performances for code (72,288) with rate R=1/4, for binary symmetric channel. There we made experiments using alphabet $Q = \{0, 1, \dots 9, a, b, c, d, e, f\}$ of nibbles and different quasigroups of order 16. The best results we obtained for quasigroup given in [2], the 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 [1] we proposed a new cut-decoding algorithm in order to improve the decoding speed and decrease packet-error and bit-error probabilities of these codes.

Now, we made experiments with the random codes based on quasigroups (with standard algorithm and with cut-decoding algorithm) in which we use quasigroups of order 4 and order 256 instead of quasigroups of order 16. Then the messages and codewords are strings of 2-bit symbols or 8-bit symbols, correspondingly. We have made experiments 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.

Using image pattern, Dimitrova and Markovski give a classification of quasigroups of order 4 in [6] as fractal and non-fractal and in [3] authors give a classification of these quasigroups as linear and nonlinear by Boolean representation. Using these classifications of quasigroups of order 4 in the experiments with messages of two-bit symbols, we obtained the worst results using fractal quasigroups. Namely, if we use fractal quasigroup then we have many unsuccessful decoding with more-candidate-error even if $B_{max} = 3$. With non-fractal non-linear quasigroups and non-fractal linear quasigroups the values for packet-error probability (PER) and bit-error probability (BER) are similar, with slightly better results for non-fractal linear quasigroup. Although the codes with nibbles and the codes with 2-bit symbols are two different codes, we obtained better results for $B_{max} = 3$ in the experiments with 2-bit symbols, compared with experiments with nibbles. But, for larger values of B_{max}, we have much unsuccessful decoding 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 were obtained for the pattern: 111000 111000 110000 000000 111000 110000 110000 000000 110000 110000 000000 000000 111000 111000 110000 000000 111000 110000 110000 000000 110000 000000 000000 000000, key $k = 012301230123213023103210$ and the quasigroup (1).

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

(1)

In Table 1 the best results for PER and BER for the both codes (72,288) (with 2-bit symbols and nibbles) using standard algorithm for different values of bit-error probability p of binary symmetric channel and $B_{max} = 3$. There, PER_2 and BER_2 are packet-error and bit-error probabilities for the code with 2-bit symbols and PER_4 and BER_4 - for the code with nibbles.

Table1: Experimental results for packet-error and bit-error probability for $B_{max} = 3$

p	PER_2	PER_4	BER_2	BER_4
0.02	0.00171	0.00475	0.00121	0.00209
0.03	0.00849	0.01843	0.00486	0.00849
0.04	0.02429	0.05559	0.01491	0.02629
0.05	0.05429	0.11758	0.03467	0.05488
0.06	0.09886	0.21314	0.05917	0.10655
0.07	0.15743	0.32971	0.09941	0.16738

From the Table 1 we can conclude that for $B_{max} = 3$ the values of PER_2 are approximately twice better than the values of PER_4. The same conclusion is true for the values BER_2 and BER_4. The better results for number of null-errors for the code with 2-bit symbols follows from the formula for theoretical packet-error probability given in Theorem1, which provide 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 for PER_2 may be greater than the theoretical PER given in the Theorem1.

Now, we made experiments for the code (72,288) using 2-bit symbols with the cut-decoding algorithm. In these experiments we use the quasigroup (1), the keys $k_1 = 012301230123213023103210$, $k_2 = 321023102130012301230123$ and the pattern: 111100 111100 111000 111000 111100 111000 111000 111100 111000 111000 110000 000000. The obtained results for PER and BER for different values of bit-error probability p of binary symmetric channel and $B_{max} = 3$ are presented in Table 2.

Table2: Experimental results for cut-decoding algorithm for $B_{max} = 3$

p	PER_2	BER_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

Comparing the suitable probabilities in Table 1 and Table 2 we can conclude that the cut-decoding algorithm gives slightly better results than the standard decoding algorithm for PER and BER for codes with 2-bit symbols. Also, for these codes the decoding process using cut-decoding algorithm is two times faster than the standard algorithm.

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, we give an example of it.

Table3: Experimental results for packet-error and bit-error probability for code $(72, 288)$, $p = 0.08$ and $B_{max} = 4$

algorithm	PER_4	PER_8	BER_4	BER_8
standard	0.1131	0.1100	0.0649	0.0749
$\text{cut} - \text{decoding}$	0.1080	0.1029	0.0751	0.0838

III. Method for reducing the more-candidate-errors

In our initial experiments with the cut-decoding algorithm we obtained worse results in the number of unsuccessful decoding of type more-candidate-error, but the number of unsuccessful decoding with null-error was smaller. To resolve this problem of greater number of more-candidate-errors, in [1] we proposed one heuristic in the decoding rule for elimination of this type of errors. Namely, from the experiments we can see that when the decoding process ends with more elements in the last set S_s, the correct message is almost always in this set. So, in these cases we randomly select a message from the set S_s and we take it as the decoded message. In the experiments which we made with this modification we got that in around half of the cases the correct message is selected. In [2] we proposed a method for decreasing the number of null-errors by backtracking. Namely, if we obtain $S_i = \emptyset$ in the i-th iteration of the decoding process then 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}. Now, we propose a similar modification with backtracking, in the cut-decoding algorithm, for decreasing the number of more-candidate-errors. If the decoding process ends with more elements in the last set S_s 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 have made experiments with this modification using the cut-decoding algorithm for the code (72, 288) with nibbles. We obtain the best results, if we cancel the last two iterations and using $B_{max} - 1$ in the first of the cancelled iterations. In Table 4 we compare the values of PER and BER obtained without this backtracking and values of PER_{back_1} and BER_{back_1} obtained with backtracking. These results are obtained for the pattern: 11001110110011001110110011 0011000000, the keys $k_1 = 01234$, $k_2 = 56789$ and $B_{max} = 4$.

Table4: Experimental results for PER and BER with and without backtracking

p	PER	PER_{back_1}	BER	BER_{back_1}
0.02	0.00171	0.00029	0.00011	0.000004
0.03	0.00257	0.00086	0.00083	0.000507
0.04	0.00514	0.00343	0.00302	0.002531
0.05	0.01714	0.01543	0.01134	0.010845
0.06	0.02657	0.02600	0.01943	0.019317
0.07	0.06171	0.05971	0.04243	0.042075
0.08	0.10800	0.10543	0.07512	0.074948
0.09	0.15886	0.15743	0.11621	0.115833

Table5: Experimental results for PER and BER with backtracking for both type of errors

p	PER_{back_1}	BER_{back_1}	PER_{back_2}	BER_{back_2}
0.02	0.00029	0.000004	0.00029	0.000004
0.03	0.00086	0.000507	0.00029	0.000238
0.04	0.00343	0.002531	0.00314	0.001421
0.05	0.01543	0.010845	0.01200	0.005571
0.06	0.02600	0.019317	0.02000	0.008698
0.07	0.05971	0.042075	0.04486	0.020171
0.08	0.10543	0.074948	0.08114	0.034599
0.09	0.15743	0.115833	0.12486	0.054984

From the results in Table 4 we can see that we have improvement of the values for PER and BER using this modification for elimination of the more-candidate-errors. Also, we have made experiments using the both methods with backtracking, for null-errors and for more-candidate-errors. In these experiments, if we obtain null-error, i.e., empty set in some iteration of the decoding process then we cancel two iterations and we reprocess the first of cancelled iterations using $B_{max}+2=6$, and if the decoding process ends with more elements in the last set S_s, then we go two iterations back and we reprocess the $(s-1)$-th iteration with $B_{max}-1=3$. Let denote by PER_{back_2} and BER_{back_2} the probabilities of packet-error and bit-error obtained in this case. In Table 5 we compare PER_{back_2} and BER_{back_2} with the values PER_{back_1} and BER_{back_1} obtained with backtracking only for more-candidate-errors. We can conclude that with the proposed combination of the backtracking for both types of errors we obtained improvement for the values of PER and BER.

Acknowledgment

This work was partially financed by the Faculty of Computer Science and Engineering at the "Ss. Cyril and Methodius University", Skopje, Macedonia.

References

[1] **A. Popovska-Mitrovikj, S. Markovski, 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.

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

[3] **D. Gligoroski, V. Dimitrova, S. Markovski**, *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

[4] **D. Gligoroski, S. Markovski, 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) 446 – 451

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

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

[7] **S. Markovski, D. Gligoroski, V. Bakeva**, *Quasigroup string processing: Part 1*, Maced. Acad. of Sci. and Arts, Sec. Math. Tech. Scien. **XX** 1-2 (1999) 13 – 28