ON IMPROVING THE DECODING OF 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 examine the influence of the length of the messages on the performances of error-correcting codes based on quasigroup transformations proposed elsewhere. We made simulations for binary symmetric channel, for different lengths of messages with the same rate and find the packet-error probability (PER) and the bit-error probability (BER). Also we give some ideas for speed-up the decoding process and improving the performances of random codes based on quasigroups.

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

I. Introduction

Here we consider the codes defined in [3] and our task is to investigate the influence of the length of the messages to the code performances with the same rate. First we give some definitions and explanation of the coding and decoding algorithm of these codes.

A. *Quasigroups and Quasigroups string transformation*

A quasigroup $(Q,*)$ is a groupoid, i.e., a set Q with a binary operation $*: Q^2 \to Q$, such that for all $u,v \in Q$, there exist unique $x,y \in Q$, satisfying the equalities $u*x=v$ and $y*u=v$. In the sequel we assume that the set Q is a finite set. The cardinality of this set, $\mid Q \mid$, 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:

$$x*y=z \Leftrightarrow y=x\backslash z. \tag{1}$$

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

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

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

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. Note that $Q^+ = Q \cup Q^2 \cup Q^3 \ldots$. Here, we use two types of quasigroup transformations as explained below. Let $l \in Q$ be a fixed element, called a leader. For every $a_i, b_i \in Q$, $e-$ and $d-$transformations are defined as follows:

$$\begin{aligned} e_l(a_1a_2\ldots a_n) = b_1b_2\ldots b_n \Leftrightarrow b_{i+1} = b_i * a_{i+1}, \\ d_l(a_1a_2\ldots a_n) = b_1b_2\ldots b_n \Leftrightarrow b_{i+1} = a_i \backslash a_{i+1}, \end{aligned} \tag{2}$$

for each $i=0,1,\ldots,n-1$, where $b_0=a_0=l$. By using the identities $x\backslash(x*y)=y$ and $x*(x\backslash y)=y$, we have that $d_l(e_l(a_1a_2\ldots a_n)) = a_1a_2\ldots a_n$ and $e_l(d_l(a_1a_2\ldots a_n)) = a_1a_2\ldots a_n$. This means that e_l and d_l are permutations on Q^n, mutually inverse. In the code design compositions of e_l and d_l are used.

Theorem 1: [4] Consider an arbitrary string $\alpha = a_1a_2\ldots a_n$ where $a_i \in Q_i$, and let β be obtained after k applications of an $e-$transformation on α. If n is an enough large integer then, for each $1 \le t \le k$, the distribution of substrings of β of length t is uniform. (We note that for $t > k$ the distribution of substrings of β of length t is not uniform.)

Code design uses the alphabet $Q = \{0,1,...,9,a,b,c,d,e,f\}$ of nibbles and a quasigroup operation $*$ on Q, together with its parastrophe $\backslash$.

B. *Description of coding*

Let $M = m_1m_2...m_r$ be a block of N_{block} bits, where m_i is a nibble (4-bit letter); hence, $N_{block} = 4r$. We first add redundancy as zero bits and produce block $L = L^{(1)}L^{(2)}...L^{(s)} = L_1L_2...L_m$ of N bits, where $L^{(i)}$ are 4-nibble words, L_i are nibbles, so $m=4s$, $N=16s$. 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 from L after applying the encryption algorithm given in Figure 1. For that aim, previously, a key $k = k_1k_2\ldots k_n$ of length n nibbles should be chosen. The obtained codeword of M is $C = C_1C_2...C_m$, where C_i are nibbles.

C. *Description of decoding*

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 4 nibbles and D_j are nibbles. 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.

The probability that $\le t$ bits in $D^{(i)}$ are not correctly transmitted is $P(p;t) = \sum_{k=0}^{t}\binom{16}{k}p^k(1-p)^{16-k}$, where p is probability of bit-error in a binary symmetric channel. Let B_{max} be an integer such that $1-P(p;B_{max}) \le q_B$ and $H_i = \{\alpha | \alpha \in Q^4,\ \ H(D^{(i)},\alpha) \le B_{max}\}$, for $i=1,2,\ldots,s$, where $H(D^{(i)},\alpha)$ is the Hamming distance between $D^{(i)}$

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

Encryption:

```
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**: $C_j \leftarrow X$
```

Decryption:

```
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

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 $(\delta, w_1w_2\dots w_{16i})$ obtained by using the sets S_{i-1} and H_i as follows (Here, w_j are bits). For each $(\beta, w_1w_2\dots w_{16(i-1)}) \in S_{i-1}$ and each element $\alpha \in H$, 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 nibbles in the same positions, then the pair $(\delta, w_1w_2\dots w_{16(i-1)}c_1c_2\dots c_{16}) \equiv (\delta, w_1w_2\dots w_{16i})$ 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 $(d_1\dots d_n, w_1\dots w_{16s})$ then $L = w_1\dots w_{16s}$. 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 $m \leq j$, $D^{(m)}$ contains more than B_{max} errors, resulting with $C_m \notin H$.

Theorem 2: [3] The packet-error probability of these codes is $q = 1 - (1 - q_B)^s$.

II. The influence of the length of the messages to the code performances

In the known error-correcting codes, the probabilities of bit-error and packet error decrease when the length of the message increase. We will show that it is not case for random codes based on quasigroups. From the formula for theoretical packet-error probability given in Theorem 2, it is clear that for larger number of blocks in the codeword the packet-error probability is larger. This follow from the fact that in this case we have more iteration in the process of decoding and in each of them the correct block should be in the appropriate (corresponding) set S. When the message M is longer, then for obtaining the same rate R, the number of blocks in the redundant message must be larger and codeword is longer than for shorter messages. So, for longer codeword we have larger packet-error probability. We check this result experimentally.

In [5], we gave experimental results, for different patterns of redundancy, for code (72,288) with rate R=1/4 for binary symmetric channel. For comparison we have made experiments with same quasigroup, key and rate for code (144,576) with twice longer messages. We compare the best results for two codes. For the code (72,288) the best results are obtained for the pattern:
110011001000000011001000100000001100110010000000
110010001000000000000000,
and for (144,576) for the pattern:
110011001000000011001000100000001100110010000000
110010001000000000001100110010000000110010001000
0000110011001000000001100100010000000000000000000.

In the experiments we use key $k = 0123456789$ and the quasigroup given in [5]. 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 1 and Table 2 .

Table 1: Experimental results for packet-error probability

p	$RCBQ(72,288)$	$RCBQ(144,576)$
0.03	0.00171	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: Experimental results for bit-error probability

p	$RCBQ(72,288)$	$RCBQ(144,576)$
0.03	0.00089	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

From the results in Table 1 and Table 2 we can conclude that the packet-error probability and bit-error probability for the longer codewords are approximately twice larger than for the shorter codewords. In Table 3 the

Table 3: Probabilities of more-candidate-errors and null-errors

	$RCBQ(72,288)$		$RCBQ(144,576)$	
p	p_{null}	p_{more}	p_{null}	p_{more}
0.03	0.00114	0.00057	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

probabilities p_{more} of more-candidate-errors and probabilities p_{null} of null-errors for these two codes are given ($p_{more} + p_{null} = PER$).

From the results presented in Table 3, 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 smaller number of redundant zero nibbles for the code (72,288). Actually, we do not have enough redundant zero blocks in the end of the redundant message L needed for putting off the incorrect blocks from the sets S in the last iterations of the decoding process.

On the other side, the number of unsuccessful decoding with null-error, which is provided in theoretical probability, is approximately two time 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 (40,160). For these short codes the number of more-candidate-errors is very large due to the small number of redundant zero blocks in the patterns.

Considering these results, we have to note that the larger number of zeros that appear in the pattern of the code (144,576) results with almost no more-candidate-errors. We discuss in Subsection B of Section III how to reduce the null-errors of the code (144,576). After making the new experiments with reduced null-errors, we are going to get a quite new results of the code (144,576).

The above results also rise an open problem. Namely, if we suppose that we have a complete random process, the redundant zero nibbles will be guest with a probability $g = 1/16$. Then, since a (144,576) code have 108 redundant zero nibbles, almost always only null-errors appears, although the decoding process is quite far to be random. On the other side, by Theorem 2, the packet-error probability of the code (144,576) is $q = 1-(1-q_B)^{36}$. Now, the mentioned problem can be stated as follows: *Find a formula that will connect the packet-error probability and the probability of guessing the redundant zeros.* The solving of this problem is hard since the two mentioned probabilities are not independent, and it is not clear, at least in this stage, how their dependencies appear.

III. Some new ideas for improving decoding process

A. The decoding speed

As it is almost always the case, the speed of the decoding process is one of the biggest problem for these random codes based on quasigroups as well. Depending on the patterns, the decoding process is slow in some iterations since the number of elements in the sets S is very large. In fact, if we distribute the redundant zeros more uniformly, the sets S are not going to be very large, but then many more-candidate-errors will appear. So, we need to put more redundant zeros at the end of the patterns. In fact, finding good equilibrium for placing the redundant zeros is an open problem and by experiments several enough satisfactory patterns are discovered.

In order to improve the decoding speed, we have experimented with another idea for decoding. Namely, instead of using a (72,288) code, we have used together two (72,144) codes, that decode a same message of length 72 bits. Then, for the obtained sets $S^{(1)}$ and $S^{(2)}$ we have made its intersection $S = S^{(1)} \cap S^{(2)}$ in each iteration step. Note that the code rate in this case is again $1/4$, since we produce two codewords of length 144 bits.

We have analyzed the number of elements in S, when we have two processes of decoding for different parameters. First we apply the coding algorithm on the same redundant message twice with different key or quasigroups and we obtain twice longer codeword (obtained as concatenation of two codewords). The codeword is transmitted through a binary symmetric channel. The outgoing message is divided in two messages with equal lengths and we decode them parallel with the corresponding parameters. After each iteration we take an intersection of the two sets $S^{(1)}$ and $S^{(2)}$ obtained in both decoding processes. In the next iteration the both processes use the obtained intersection S. The preliminary experiments show that this modification significantly reduces the number of elements in the sets S. The problem here is that for obtaining code with rate $1/4$ we need pattern for rate $1/2$, but it is hard to make good pattern for this rate. In the initial experiments, with this modified algorithm, we obtained similar values for PER and BER as for (72,288) code. What is very promising for this modified algorithm is that we have big improvement in the speed of the decoding process, which is in average more than four times faster.

B. Reducing the null-errors

In the case when null-errors appear, i.e., when the set S is empty, we are going to resolve this problem by backtracking. Namely, if in the i-th iteration we obtain $S = \emptyset$, it means that in some previous step $j < i$ we have lost the right word that had to be processed. This happened because B_{max} is too small. So, we have to go back to the $(i-1)$-th or $(i-2)$-th step and there to work with increased value of B_{max}, i.e., we take $B_{max}+1$ or $B_{max}+2$. The experiments of this kind are under preparation and we expect improving of the PER and BER.

Anyhow, we expect the decoding speed to be increased, but not significantly.

C. Reducing the more-candidate-errors

Another idea that we consider for elimination the unsuccessful decoding with more-candidate-error is the following one. From the experiments we can see that when the decoding process ends with more elements in the last set S, almost always in this set is the correct message. So, in these cases we can randomly select a message from the set S in the last iteration and it can be taken as the decoded message. If the selected message is the correct then the bit-error is 0, so the BER will also be reduced. In the experiments we have made with this modification we got that in around half of the cases, when the last set S contained more than one element, the correct message is selected.

References

[1] **V. Bakeva, V. Dimitrova**, *Some probabilistic properties of quasigroup processed strings useful in cryptanalysis*, M.Gushev, P.Mitreski (Eds.): ICT-Innovations 2010, Springer (2010) 61 – 70

[2] **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

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

[4] **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

[5] **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