Error-Correcting Codes with Cryptographic Algorithms

Aleksandra Popovska-Mitrovikj, Smile Markovski, and Verica Bakeva, *Member, IEEE*

***Abstract* — Error-correcting codes based on quasigroups are defined elsewhere. These codes are a combination of cryptographic algorithms and error correcting codes. In a paper of ours we succeed to improve the speed of the decoding process by defining new algorithm for coding and decoding, named "cut-decoding algorithm". Here, a new modification of the cut-decoding algorithm is considered in order to obtain further improvements of the code performances. We present several experimental results obtained with different decoding algorithms for these codes.**

***Keywords* — bit-error probability, cryptcoding, error-correcting code, packet-error probability, quasigroup, quasigroup transformation, random code.**

I. INTRODUCTION

THE Random Codes Based on Quasigroups (RCBQ) are defined in [1]. These codes have several parameters and they are a combination of cryptographic algorithms and error correcting codes. Therefore, RCBQ allow not only correction of certain amount of errors in the input data, but also they provide an information security. In [2] we have investigated the influence of the code parameters to the code performances and in [3] we have proposed a cut-decoding algorithm such that the modified decoding process is 4.5 times faster than the original one for code (72,288). In cut-decoding algorithm we use two transformations of the redundant message with different parameters, and the candidates for the decoded message are obtained by using intersection of the corresponding sequences. On this way we obtained a significant reduction in the number of elements in the sets of candidates for decoded message. These results give us an idea to use intersections of more decoding candidate sets in order to obtain greater improvement of the speed of decoding process. Therefore, we make modifications of the cut-decoding algorithm where we use four transformations of the redundant message and we obtain greater improvement of the code performances.

The RCBQ are designed using algorithms for encryption/decryption from the implementation of TASC (Totally Asynchronous Stream Ciphers) by quasigroup string transformations [4]. These cryptographic algorithms use the alphabet Q and a quasigroup operation * on Q together with its parastrophe \. In the next section we will give briefly description of these codes using quasigroups, but from the definition of the algorithms it is clear that in their design can be used other algorithms for encryption and decryption.

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

Aleksandra Popovska-Mitrovikj, Smile Markovski and Verica Bakeva are with the Faculty of Computer Science and Engineering, University "Ss Cyril and Methodius" - Skopje, P.O. Box 393, R. of Macedonia (phone: +389-71-277093; e-mails: {aleksandra.popovska.mitrovikj, smile.markovski, verica.bakeva}@finki.ukim.mk

II. DESCRIPTION OF RCBQ

A. Description of coding with standard algorithm and cut-decoding algorithm

Let $M = m_1m_2\ldots m_l$ be a block of $N_{block}= la$ bits where $m_i \in Q$ and Q is an alphabet of a-bit symbols. First, we add a redundancy as zero symbols and produce message $L=L^{(1)} L^{(2)} \ldots L^{(s)} = L_1L_2\ldots L_m$ of N bits, where $L^{(i)}$ are sub-blocks of r symbols from Q and $L_i \in Q$ (so, $sr = m$). 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 of TASC (given in Fig. 1) on the message L. For that aim, previously, a key $k=k_1k_2\ldots k_n \in Q^n$ should be chosen. The obtained codeword of M is $C=C_1C_2\ldots C_m$, where $C_i \in Q$.

Encryption	Decryption
Input: Key $k = k_1k_2\ldots k_n$ and message $L = L_1L_2\ldots L_m$ **Output**: message (codeword) $C = C_1C_2\ldots C_m$	**Input**: The pair $(a_1 a_2\ldots a_s, k_1k_2\ldots k_n)$ **Output**: The pair $(c_1 c_2\ldots c_s, K_1K_2\ldots K_n)$

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 $s-1$
	  $X, T \leftarrow a_{j+1}$;
	  $temp \leftarrow K_n$;
	  For $i = n$ down 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:$(c_1c_2\ldots c_s, K_1K_2\ldots K_n)$

Fig. 1. TASC algorithm for encryption and decryption.

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

B. Description of decoding with standard algorithm and cut-decoding 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.

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 $H_i = \{\alpha \mid \alpha \in Q^r, H(D^{(i)}, \alpha) \leq B_{max}\}$, for $i = 1, 2, \ldots, s$, where $H(D^{(i)}, \alpha)$ is the Hamming distance between $D^{(i)}$ and α.

The decoding candidate sets $S_0, S_1, \ldots, S_s$ are defined iteratively. Let $S_0 = (k_1k_2\ldots k_n; \lambda)$, 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 $(\delta, w_1w_2\ldots w_{rai})$ obtained by using the sets S_{i-1} and H_i as follows (w_j are bits). For each element $\alpha \in H_i$ and each $(\beta, w_1w_2\ldots w_{ra(i-1)}) \in S_{i-1}$, we apply the inverse coding algorithm (i.e. algorithm for decryption given in Fig. 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 $(\delta, w_1w_2\ldots w_{ra(i-1)}\, c_1c_2\ldots c_{ra}) \equiv (\delta, w_1\, w_2\ldots w_{rai})$ 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 $(d_1\ldots d_n, w_1w_2\ldots w_{ras})$ then $L = w_1\, w_2\ldots w_{ras}$ is the decoded (redundant) message (then we say 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-error*). In the case when $S_j = \emptyset$ for some $j \in \{1, \ldots, s\}$, the process will be stopped (we say that a *null-error* appears).

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

In the cut-decoding algorithm, after transmitting through a noise channel, we divide the outgoing message $D = D^{(1)}D^{(2)}...D^{(s)}$ 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 this method of decoding we make modification in the procedure for generating decoding candidate sets. In this algorithm we generate these sets in the following way.

Step 1. Let $S_0^{(1)} = (k_1^{(1)}k_2^{(1)}\ldots k_n^{(1)}; \lambda)$ and $S_0^{(2)} = (k_1^{(2)}k_2^{(2)}\ldots k_n^{(2)}; \lambda)$ where λ is the empty sequence, and let $k_1^{(1)}k_2^{(1)}\ldots k_n^{(1)}$ and $k_1^{(2)}k_2^{(2)}\ldots k_n^{(2)}$ be the initial keys used for obtaining the two codewords.

Step 2. Let $S_{i-1}^{(1)}$ and $S_{i-1}^{(2)}$ be defined for $i \geq 1$.

Step 3. Let two decoding candidate sets $S_i^{(1)}$ and $S_i^{(2)}$ be obtained in the both decoding processes, on the same way as in the standard algorithm of RCBQ.

Step 4. Let $V_1 = \{w_1w_2\ldots w_{rai} \mid (\delta, w_1w_2\ldots w_{rai}) \in S_i^{(1)}\}$, $V_2 = \{w_1w_2\ldots w_{rai} \mid (\delta, w_1w_2\ldots w_{rai}) \in S_i^{(2)}\}$ and $V = V_1 \cap V_2$.

Step 5. For each $(\delta, w_1w_2\ldots w_{rai}) \in S_i^{(1)}$, if $w_1w_2\ldots w_{rai} \notin V$ then $S_i^{(1)} \leftarrow S_i^{(1)} \setminus \{(\delta, w_1\, w_2\ldots w_{rai})\}$. Also, for each $(\delta, w_1\, w_2\ldots w_{rai}) \in S_i^{(2)}$, if $w_1w_2\ldots w_{rai} \notin V$ then $S_i^{(2)} \leftarrow S_i^{(2)} \setminus \{(\delta, w_1\, w_2\ldots w_{rai})\}$.

(We note that 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.

If, after the last iteration, the reduced sets $S_{s/2}^{(1)}$ and $S_{s/2}^{(2)}$ have only one element with same second component $w_1\ldots w_{ras/2}$, then $L = w_1\ldots w_{ras/2}$ and we have *successful decoding*. If, after the last iteration, the reduced sets have more than one element 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 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).

With the cut-decoding algorithm, we noticed a significant reduction of the number of elements in the sets S and we achieve big improvement of the decoding speed (4.5 times faster for code (72,288)). The problem in the cut-decoding algorithm is that for obtaining code with rate R we need a pattern for code with rate $2R$. 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 we can see that when the decoding process ends with more elements in the last reduced decoding candidate sets, almost always in these sets is contained the correct message. So, in this case we can randomly select a message from the one of the 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 the bit-error probability (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.

III. New 4-Sets-Cut-Decoding Algorithm

In this paper we propose a new modification of the cut-decoding algorithm where we use cuts of four decoding candidate sets. In this modification of cut-decoding algorithm instead of using a (N_{block}, N) code with rate R, we use together 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 Fig. 1, 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. After transmitting through a noise 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 parallel with the corresponding parameters.

Similarly, as in the cut-decoding algorithm with two sets, we reduce the decoding candidate sets obtained in the four decoding processes (in all iterations of the decoding process). In the initial experiments with this modification, for reduction we have used intersection of all four decoding candidate sets. But, in these experiments, 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 the four non-reduced sets. Therefore, we introduced heuristic, i.e., an additional step in the algorithm, for the cases when the intersection of all four sets is empty. So, in the new 4-Sets-Cut-Decoding algorithm for RCBQ we generate decoding candidate sets in the following way.

Step 1. Let $S_0^{(1)} = (k_1^{(1)}\ldots k_n^{(1)}; \lambda),\ldots, S_0^{(4)} = (k_1^{(4)}\ldots k_n^{(4)}; \lambda)$ where λ is the empty sequence, $k_1^{(1)}\ldots k_n^{(1)},\ldots, k_1^{(4)}\ldots k_n^{(4)}$ are the initials keys used for obtaining the four codewords.

Step 2. Let $S_{i-1}^{(1)},\ldots,S_{i-1}^{(4)}$ be defined for $i \geq 1$.

Step 3. Let four decoding candidate sets $S_i^{(1)},\ldots,S_i^{(4)}$ be obtained in the four decoding processes, on the same way as in the standard algorithm of RCBQ.

Step 4. Let $V_1 = \{w_1w_2\ldots w_{rai} | (\delta, w_1w_2\ldots w_{rai}) \in S_i^{(1)}\},\ldots, V_4 = \{w_1w_2\ldots w_{rai} | (\delta, w_1w_2\ldots w_{rai}) \in S_i^{(4)}\}$ and $V = V_1 \cap V_2 \cap V_3 \cap V_4$.

If $V = \varnothing$ then $V = (V_1 \cap V_2 \cap V_3) \cup (V_1 \cap V_2 \cap V_4) \cup (V_1 \cap V_3 \cap V_4) \cup (V_2 \cap V_3 \cap V_4)$.

Step 5. For each j = 1, 2, 3, 4 and for each $(\delta, w_1 w_2\ldots w_{rai}) \in S_i^{(j)}$, if $w_1 w_2\ldots w_{rai} \notin V$ then $S_i^{(j)} \leftarrow S_i^{(j)} \setminus \{(\delta, w_1 w_2\ldots w_{rai})\}$.

(Note that 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.

If, after the last iteration, all reduced 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 $w_1 w_2\ldots w_{ras/4}$, then $L = w_1 w_2\ldots w_{ras/4}$ is the decoded (redundant) message and we have *successful decoding*. If, after the last iteration, the reduced sets $S_{s/4}^{(1)}$, $S_{s/4}^{(2)}$, $S_{s/4}^{(3)}$, $S_{s/4}^{(4)}$ have more than one element we have *more-candidate-error*. In this case we apply the same heuristic as in the cut-decoding algorithm (we randomly select a message from the reduced sets in the last iteration). If we obtain $S_{s/4}^{(1)} = S_{s/4}^{(2)} = S_{s/4}^{(3)} = S_{s/4}^{(4)} = \varnothing$ in some iteration, then the process will be stopped (*null-error* appears). But, if we obtain only one (or two) empty decoding candidate set (in Step 3) then the decoding continues with the three (or two) nonempty sets. If, in some iteration, we obtain only one nonempty set then the decoding continues with the nonempty set using the standard RCBQ decoding algorithm.

IV. Experimental results

In this section we give the experimental results for the probabilities for packet-error (PER) obtained with the new 4-Sets-Cut-Decoding algorithm and we compare them with the results obtained with the standard decoding algorithm and the cut-decoding algorithm with two sets. Also, we will compare the decoding speeds obtained from the experiments for all algorithms.

In previous papers of ours [2], [3], we have given experimental results for the code (72,288) with rate ¼ obtained with the standard and the cut-decoding algorithm. But, for obtaining the code (k, n) with rate R in the proposed 4-Sets-Cut-Decoding algorithm, we use four (k, $n/4$) codes with rate $4R$. Therefore, for code with rate ¼ we do not have redundancy (if R = ¼ then $n = k$, i.e., the length of the codeword is equal to the length of the message). So, we made experiments for code (72,576) with rate R = 1/8. In these experiments we used alphabet Q={0, 1,…, 9, a, b, c, d, e, f} of nibbles with the quasigroup operations * and \ on Q given in [2] and blocks of 4 nibbles in the decoding process.

We obtained the best results for code (72,576) with the standard decoding algorithm for the pattern: 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 initial key of 10 symbols. (Here, 1 denotes the place of a message symbol, and 0 is the redundant symbol of *a* zero bits.) With cut-decoding algorithm with 2 sets, we obtained the best results with the redundancy pattern: 1100 1100 1000 0000 1100 1000 1000 0000 1100 1100 1000 0000 1100 1000 1000 0000 0000 0000, for rate ¼ and two different keys of 10 symbols. In the experiments with the new 4-Sets-Cut-Decoding algorithm we used the pattern: 1100 1110 1100 1100 1110 1100 1100 1100 0000 for rate ½ and four different keys of 10 symbols. In all experiments we used the same quasigroup on Q.

Experimental results for packet-error probabilities for B_{max} = 4 and different values of bit-error probability p of binary symmetric channel are presented in Fig. 2. In this figure the *PER*s are the packet-error probabilities obtained with the standard algorithm, *PER*c with the cut-decoding algorithm with 2 sets and *PERc4* the packet-error probabilities obtained with 4-Sets-Cut-Decoding algorithm. For all considered algorithms we made experiments until we get $PER > 0.1$.

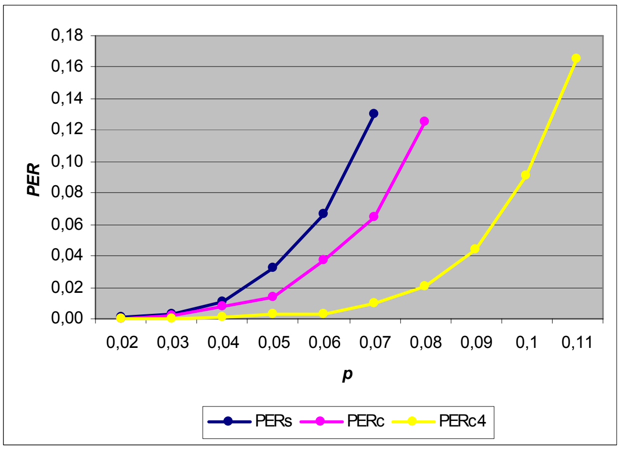

Fig. 2. Experimental results for packet-error probabilities for $B_{max} = 4$ for code (72, 576)

From the results obtained for *PER* given in the Fig. 2 we can derive the following conclusions. Using the cut-decoding algorithm with 2 sets instead of the standard algorithm we obtain great improvement of the probabilities for packet-error (for $p > 0.04$, *PERc* are approximately twice smaller than *PERs*). Also, analyzing the average times of the experiments we can conclude that for $B_{max} = 4$, the cut-decoding algorithm is more than 2 times faster than the standard algorithm. From the values for *PERc4* we can see that with this new modification we obtain better results for *PER* for all values of p compared with the values obtained with the standard and the cut-decoding with 2 sets. Also, this algorithm is 1.4 times faster than the cut-decoding with 2 sets and more than 3 times faster than the standard algorithm.

Also, we made experiments for the same codes using $B_{max} = 5$ in the decoding process and in Fig. 3 we present the obtained results for packet-error probabilities for different values of bit-error probability p of binary symmetric channel.

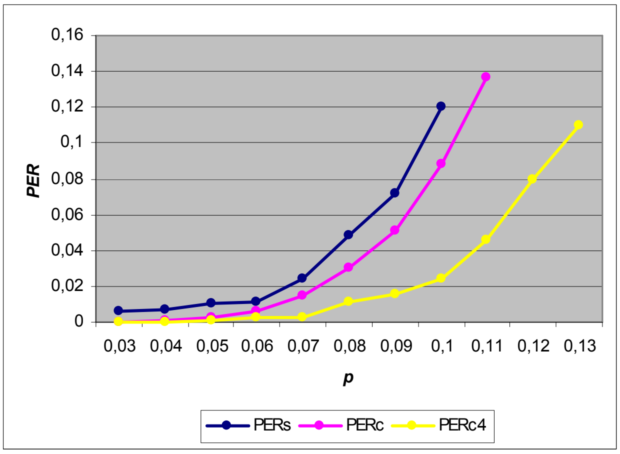

Fig. 3. Experimental results for packet-error probabilities for $B_{max} = 5$ for code (72, 576)

From the values for *PERs* and *PERc* in Fig. 3 we can conclude that using the cut-decoding algorithm with 2 sets for rate 1/8 we obtain better results than with the standard algorithm. On the other side, in terms of the decoding speed, for $p < 0.06$ the cut-decoding algorithm is 5.2 times faster than the standard one, but for $p \geq 0.06$, in some experiments with cut-decoding algorithm, we obtained a very large cardinality of the decoding candidate sets (after some iteration), so the decoding speed decreases (but it is still better than the speed obtained with standard algorithm). From the results in Fig. 3 we can see that with the 4-Sets-Cut-Decoding algorithm, we obtain better values for *PER* for all values of p (for $p \geq 0.07$, *PERc4* is more than 3 times smaller than *PERc* and for all p *PERc4* is more than 4 times smaller than *PERs*). Also, this algorithm is more than 6 times faster than the standard algorithm and from 1.16 to 5.6 times (for different values of p) faster than the cut-decoding with 2 sets.

In this paper we do not present the results obtained for bit-error probabilities (*BER*), but for this probabilities we can derive the same conclusions as for *PER* (since, for all algorithms and for all p, *BER* is approximately *PER*/2).

V. Conclusion

In this paper, in order to improve the decoding speed of random codes based on quasigroups we have defined new 4-Sets-Cut-Decoding algorithm. With this algorithm we obtained greater reduction of the cardinality of decoding candidate sets in all iterations. Also, we have introduced an additional heuristic (when decoding ends with *null-error*) in the proposed decoding algorithm, and we obtained improving of the packet-error and bit-error probabilities. Several experiments for different decoding algorithms for code (72,576) were presented and compared. From the comparison we can conclude that for this code with the new 4-Sets-Cut-Decoding algorithm we obtain great improvement of the decoding speed and much better values for packet-error and bit-error probabilities.

References

[1] D. Gligoroski, S. Markovski, Lj. Kocarev, "Error-correcting codes based on quasigroups," in *Proc. 16th International Conference on Computer Communications and Networks* (ICCCN 2007), 2007, pp.165-172.

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

[3] A. Popovska-Mitrovikj, S. Markovski, V. Bakeva,, "Increasing the decoding speed of random codes based on quasigroups," in *Web Proc. ICT Innovations2012*, ISSN 1857-7288, Ohrid, 2012, pp. 93-102.

[4] D. Gligoroski, S. Markovski, Lj. Kocarev, "Totally asynchronous stream ciphers + Redundancy = Cryptcoding," in *Proc. of the 2007 International Conference on Security and management, SAM 2007* S. Aissi, H.R. Arabnia , Eds., CSREA Press, Las Vegas, 2007, pp. 446 – 451.