Research, Implementation and Application of the SQBC Block Cipher in the Area of Encrypting Images

Zlatka Trajcheska, Vesna Dimitrova

Faculty of Computer Science and Engineering

zlatka.trajcheska@gmail.com, vesna.dimitrova@finki.ukim.mk

Abstract. The application of block ciphers in the modern society is enormous. One branch of this development is getting a new dimension – the application of quasigroups and quasigroup transformations in the block ciphers. This paper contains the results of the research involving the family of block ciphers named SQBC (Small Quasigroup Block Cipher), which is based on quasigroups and quasigroup transformations. Actually, we will discuss the results of encrypting images in different formats using the SQBC block cipher based on quasigroups of order 4.

Keywords: cryptography, block cipher, images, quasigroup, encryption, SQBC

1 Introduction

Cryptography is not only a scientific field, it is a fundamental part of our everyday living. We could not possibly imagine a world without the usage of its benefits. The modern society relies on the cryptographic algorithms for the purpose of simple electronic communications, electronic transactions or military causes. Considering this, the development and improvement of cryptography is essential for today's way of living.

Generally, cryptographic algorithms can be divided into two groups: symmetric and public key (or asymmetric) algorithms. On the other side, symmetric algorithms can be stream or block ciphers. Block ciphers are the ones that we are interested in, as SQBC is a family of block ciphers. In particular, SQBC is based on quasigroups and quasigroup transformations of small order, and we will use quasigroups of order 4. For the purposes of this research, before we continue with the results of the encryption, we will first discuss some basic terms about block ciphers, quasigroups and their classification, and finally the SQBC algorithms for encryption and decryption.

2 Block Ciphers

Encryption of a plaintext message using a block cipher is done by separating the plaintext message into blocks with fixed length, and encrypting them individually, so

S. Markovski, M. Gusev (Editors): ICT Innovations 2012, Web Proceedings, ISSN 1857-7288
© ICT ACT – http://ictinnovations.org/2012, 2012

that every block from the plaintext message is encrypted into a block of the encrypted message that has the same length. Both sides of the communication have the same secret key. The block length varies, but usually we are considering block length of 64, 128, 256 bits and so on. To obtain the final encrypted message from the individually encrypted blocks, we can use different ways to combine them. These are known as modes of operation. Two of them (ECB and CBC) are used in the research, so they will be briefly discussed in addition.

The ECB (Electronic Code Book) mode of operation is the simplest and the least secure of the modes. It simply concatenates the encrypted blocks.

The encryption with CBC (Cipher Block Chaining) mode is done by combining the encrypted messages the following way: at the beginning we have an initial vector (IV) which is of same length as the block. This initial vector is XORed with the first block, and then encrypted, so that is the first encrypted block. Afterwards, we get every new encrypted block by XORing the previous encrypted block and the current plaintext block and encrypting the result.

3 Quasigruops and Their Classification

A groupoid $(G,*)$, where * is a binary operation, is called a quasigroup if for each a and b in G there are unique x and y in G so that:

$$a * x = b = y * a \tag{1}$$

or formally

$$(\forall\, a, b\ \in G)\ (\exists x, y\ \in G)(a * x = b = y * a) \tag{2}$$

For every quasigroup five other quasigroups can be derived, which are called parastrophes. In particular, we are interested in the so called left and right parastrophe which are defined by the following equivalences [1]:

$$x*y=z \qquad y=x\backslash z \qquad x=z/y \tag{3}$$

- There are 576 quasigroups of order 4, and we will use them in the research. Considering the results of a previously conducted research [1], there are several classifications on the quasigroups of order 4. In this paper we consider the classification by fractality (fractal and non-fractal) and classification by Boolean representation of quasigroups (linear, non-linear and purely non-linear). The results from [1] reveal several properties of the quasigroups which will be used in the research. Actually, several representative quasigroups were taken in our research and also their left and right parastrophes. We use the following quasigroups given with its lexicographical number quasigroup 1, 6, 158 and 181 – as a representative of the fractal and linear quasigroups, the non-fractal and linear quasigroups, the non-fractal and non-linear quasigroups and the non-fractal and purely non-linear quasigroups, respectively.

- It is expected that the quasigroups that are linear by Boolean representation or fractal give bad results used for encryption and we want to see if these quasigroups cause some undesirable structures or fractals[1] in the encrypted image.

4 The Algorithm of the SQBC Block Cipher

SQBC is a family of block ciphers that use quasigroups and quasigroup transformations to encrypt the plaintext message in encrypted message, using a working key generated directly from a secret key. Obviously, the cipher includes decrypting the encrypted message, which also is based on quasigroup transformations.

The algorithm of this cipher is described in “SQBC - Block cipher defined by small quasigroups” [2] and in this section the discussed content is referenced to it.

Without going into detail about the algorithms used and their construction, we can more generally say that this cipher is different from most of the common and well known block ciphers, as it uses two algorithms for encryption and two for decryption. Actually, one encryption algorithm is used to encrypt the first block and another one to encrypt all the other blocks. Similarly is for the decryption. The encryption/decryption algorithm uses e transformation or d transformation, respectively. The e and d transformations are defined as:

$$e_{*,l}(\alpha) = b_1 b_2 \dots b_n, \text{so that } b_{i+1} = b_i * a_{i+1} \tag{4}$$

$$d_{*,l}(\alpha) = c_1 c \dots c_n \text{ , so that } c_{i+1} = a_i * c_{i+1} \tag{5}$$

There is also an algorithm for generating working key out of the secret key which should be at least 80 bits long.

In order to examine the avalanche effect[2] of this block cipher, it was implemented in Java [3]. The interface allows the user to enter a number of rounds and the block length. Also, it is required to enter the secret key and generate the working key before we proceed to encrypting and decrypting. The avalanche effect can be calculated both for the encryption and the working key generation algorithm. This implementation can be used in various purposes. In this paper, we will apply it in the area of encrypting images.

5 Application in the Area of Encrypting Images

The modern way of living and the fact that the information technologies are ubiquitous in all scientific fields, the culture and life in general demand paying a lot more attention to the data security, especially when concerning sensible data. Lately, those sensible data are often some images. And this is where the idea for encrypting images

[1] the term fractal here is used as a general term for any recognizable pattern or structure that can appear on the picture

[2] the term avalanche effect represents the percentage of different bits between the encryptions of two very similar plaintext messages (that are differ only in one bit)

comes from. Actually, the SQBC block cipher may be used in several devices like personal identification cards, where there is an obvious need for encrypting images. Before we enclose the results of the experimental research conducted with various image formats, we should mention that the header of the images, which varies from one format to another, is not encrypted, because we want to display the results of the encryption like an image as well. That way we can visually see any fractals that may appear.

5.1 Encryption of *.bmp Images with the SQBC Block Cipher

Using the mentioned implementation an experimental research was conducted in order to see the effect of encrypting images with this cipher. The results are given in details in the paper "Encrypting images with SQBC"[4]. In this paper we will shortly discuss this part too, as we want to compare the results with the ones obtained from our research.

In the previous research given in [4] were used 24-bit Bitmap image format. This format is made up from 4 blocks – Header, InfoHeader, RGBQuadArray and color indexed array. More details about the format are given in Table 1. This image format is encoded with Windows-1251 encoding, so some slight changes were made in the previous implementation. Actually, to make the testing faster and easier, the manual input of the plaintext message was avoided and was replaced from reading the plaintext message from an external file. To avoid problems with the encoding the HxD hexadecimal editor was used. As said before, the first 54 bytes (the header) was not encrypted.

Table 1**.** Bitmap file structure

Name	Size	Description
Header	14 bytes	Windows Structure: Bitmap File Header
InfoHeader	40 bytes	Windows Structure: Bitmap Info Header
RGBQuad array	4 bytes	
color-index array	Varies	

At first, the CBC mode was used for encryption. The results were very similar to each other and all results showed that a fractal does not appear. Figure 3 shows the original *.bmp image, as for 4 and 5 show the results of the encryption using quasigroups number 1 and 181, respectively. The block length used is 128 bits.

Fig. 1. The original *.bmp image (1)

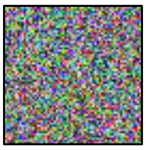


Fig. 2. Encryption of (1) with quasigroup number 1 and CBC mode

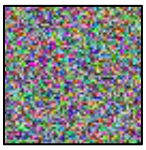


Fig. 3. Encryption of (1) with quasigroup number 181 and CBC mode

Also, the same experiment was conducted using ECB mode to see more clearly the direct impact of the quasigroups used and its properties. Another change was made, which was to encrypt all blocks with the same algorithm. The testing was done with 8, 16 and 24 bits, because of the structure of *.bmp images. Actually, by encrypting 8 bits long blocks every color in the pixel is encrypted as one block. Encrypting blocks of 16 bits is actually encrypting two colors of the pixel as one block, and finally encrypting blocks which are 24 bits long result in encrypting exactly one pixel.

Using the ECB mode as described some fractal structures started to appear. They are not that obvious at first, but we must consider that the algorithm of block cipher itself has a great impact on the fact that the encrypted image is not periodic. When using ECB mode, as expected, whenever the input was periodic, the encryption was periodic, too. The images below show the results of the encryptions with ECB mode, using blocks which are 8, 16 and 24 bits long, and the quasigroups 1, 6, 158 and 181 for the properties mentioned above.

The Figures 6, 7 and 8 show the results of the encryption of the original image (1) using the quasugroup 1, ECB mode and block length of 8, 16 and 24 bits.

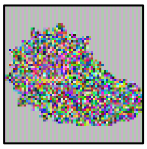


Fig. 4. Encryption of (1) with quasigroup 1, 8 bit block

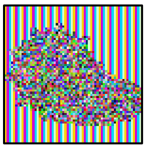


Fig. 5. Encryption of (1) with quasigroup 1, 16 bit block

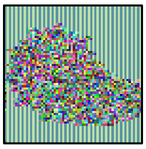


Fig. 6. Encryption of (1) with quasigroup 1, 24 bit block

The Figures 9, 10 and 11 show the results of same experiments using the quasigroup 6, the Figures 12, 13 and 14 using the quasigroup 158 and the Figures 15, 16 and 17 using the quasigroup 181.

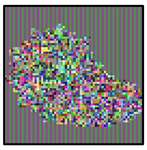


Fig. 7**.** Encryption of (1) with quasigroup 6, 8 bit block

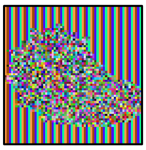


Fig. 8**.** Encryption of (1) with quasigroup 6, 16 bit block

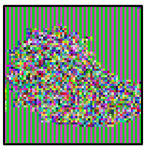


Fig. 9**.** Encryption of (1) with quasigroup 6, 24 bit block

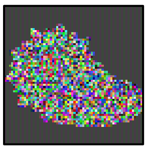


Fig. 10**.** Encryption of (1) with quasi. 158, 8 bit block

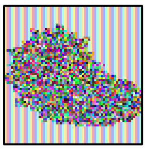


Fig. 11**.** Encryption of (1) with quasi. 158, 16 bit block

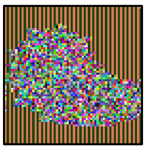


Fig. 12**.** Encryption of (1) with quasi. 158, 24 bit block

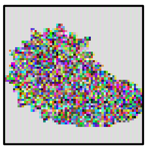


Fig. 13**.** Encryption of (1) with quasi. 181, 8 bit block

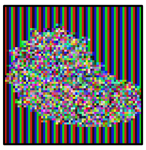


Fig. 14**.** Encryption of (1) with quasi. 181, 16 bit block

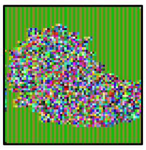


Fig. 15**.** Encryption of (1) with quasi. 181, 24 bit block

The encryption of images with SQBC is more elaborated in [4] and in the next sections, we will discuss more about this research – encrypting images of other formats.

5.2 Encryption of *.jpg images with the SQBC block cipher

The *.jpg image format is massively accepted and used nowadays. It provides a very powerful compression which contributes to the small size of the *.jpg image file. But, this reflects on the quality of the picture.

The *.jpg format is very different from other image formats. While *.png, *.bmp and even *.gif have the so called lossless compression and allow the image to be fully restored, the *.jpg format is intended to have loss of quality in order to obtain small size of the file [5].

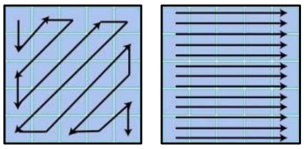
The structure of a *.jpg file according to its official documentation is complex and may contain a lot of markers. But, most often the *.jpg images are saved as JFIF (JPEG File Interchange Format). Table 2 shows the structure of a JFIF segment.

Table 2**.** Structure of a JFIF segment

Name	Size	Description
APP0 marker	2 bytes	It is always the same value: 0xFFE0
Length	2 bytes	The segment length including the APP0

		marker including the APP0 marker
Identifier	5 bytes	Always has the same value 0x4A46494600, which is JFIF0 in ASCII
Version	2 bytes	The first byte is for the major version, and the second one for the minor version
density units	1 bytes	Holds the information about the units for pixel density fields (0 – only aspect ratio, 1 – pixels per inch, 2 – pixels per centimeter)
X density	2 bytes	horizontal pixel density
Y density	2 bytes	vertical pixel density
thumbnail width (tw)	1 bytes	thumbnail width in pixels
thumbnail height (th)	1 bytes	thumbnail height in pixels
thumbnail data	3 x tw x th	uncompressed 24-bit RGB rasterised thumbnail

Without discussing about all the details of this format, we will consider only the aspects which are important about the encryption of the images. Actually, while encoding the image, a chromatic subsampling occurs. In simple words, this means that the encoding uses the flaws of the human eye to compress the image and save it in smaller resolution, so that it wouldn't be noticeable on first glance. After the subsampling, the image channels are divided on blocks of 8x8 bits. They undergo several operations, which finally leads to dependencies between the blocks. Figure 18 a) shows the so called zigzag encoding.

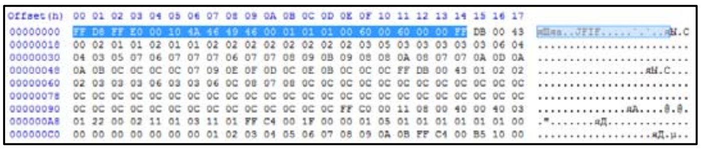


a) b)

Fig. 16. a) JFIF zigzag encoding b) Encrypting the image with SQBC

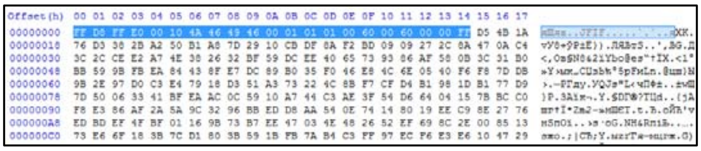
Now that the picture is divided into dependent blocks it will be encrypted sequentially, bit by bit, as shown on Figure 18 b). So, with the encryption of the image, the dependencies are lost and the output cannot be shown as a *.jpg image. This does not mean that the images of *.jpg format cannot be encrypted or decrypted with SQBC, but the result will have to be stored textually.

Because the output of the encryption cannot be shown as an image, Figures 19 and 20 show the original and the encrypted image in textual and hexadecimal representation.



```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10 11 12 13 14 15 16 17
00000000  FF D8 FF E0 00 10 4A 46 49 46 00 01 01 01 00 60 00 60 00 00 FF DB 00 43  яШяа..JFIF.....`.`..яЫ.C
00000018  00 02 01 01 02 01 01 02 02 02 02 02 02 02 02 03 05 03 03 03 03 03 06 04  ........................
00000030  04 03 05 07 06 07 07 07 06 07 07 08 09 0B 09 08 08 0A 08 07 07 0A 0D 0A  ........................
00000048  0A 0B 0C 0C 0C 0C 07 09 0E 0F 0D 0C 0E 0B 0C 0C 0C FF DB 00 43 01 02 02  .................яЫ.C...
00000060  02 03 03 03 06 03 03 06 0C 08 07 08 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C  ........................
00000078  0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C  ........................
00000090  0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C FF C0 00 11 08 00 40 00 40 03  ..............яА....@.@.
000000A8  01 22 00 02 11 01 03 11 01 FF C4 00 1F 00 00 01 05 01 01 01 01 01 01 00  ."........яД............
000000C0  00 00 00 00 00 00 00 01 02 03 04 05 06 07 08 09 0A 0B FF C4 00 B5 10 00  ..................яД.µ..
```

Fig. 17**.** The original *.jpg image in textual/hexadecimal representation



```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10 11 12 13 14 15 16 17
00000000  FF D8 FF E0 00 10 4A 46 49 46 00 01 01 01 00 60 00 60 00 00 FF D5 4B 1A  яШяа..JFIF.....`.`..яХK.
00000018  76 D3 38 2B A2 50 B1 A8 7D 29 10 CB DF 8A F2 BD 09 09 27 2C 8A 47 0A C4  vУ8+ўP±Ё}).ЛЯЉт..',ЉG.Д
00000030  3C 2C CE E2 A7 4E 38 26 32 BF 59 DC EE 40 65 73 93 86 AF 58 0B 3C 31 B0  <,Ов§N8&2їYЬо@es"†ЇX.<1°
00000048  BB 59 9B FB EA 84 43 8F E7 DC 89 B0 35 F0 46 E8 4C 6E 05 40 F6 F8 7D DB  »Y›ык„CЏзЬ‰°5рFиLn.@цш}Ы
00000060  9B 2E 97 D0 C3 E4 79 18 D3 51 A3 73 22 4C 8B F7 CF D4 B1 98 1D B1 77 D9  ›.—РГдy.УQЈs"L‹чПФ±..±wЩ
00000078  7D 50 06 33 41 BF EA AC 0C 59 10 A7 44 C3 AE 3F 54 D6 64 04 15 7B BC C0  }P.3Aїк¬.Y.§DГ®?TЦd..{јА
00000090  F8 E3 86 AF 2A 5A 9C 32 96 BB ED D8 AA 54 0E 74 14 80 19 EE C9 8E 27 76  шг†Ї*Zњ2–»нШЄT.t.Ђ.оЙЋ'v
000000A8  ED BD EF 4F BF 01 16 9B 73 B7 EE 47 03 4E 48 26 52 EF 69 8C 2E 00 85 13  нЅпOї..›s·оG.NH&RпiЊ.....
000000C0  73 E6 6F 18 3B 7C D1 80 3B 59 1B FB 7A B4 C3 FF 97 EC F6 E3 E6 10 47 29  sжo.;|СЂ;Y.ыzґГя—мцгж.G)
```

Fig. 18**.** The encrypted *.jpg image in textual/hexadecimal representation

5.3 Encryption of *.png Images with the SQBC Block Cipher

Unlike the *.jpg image format, the *.png image format is bitmapped and enables lossless compression. There are two aspects of its structure that are important for the encryption and that are the *.png file header and several markers called chunks that the file might contain .

The *.png file header is represented by the fixed 8 bytes 89 50 4E 47 0D 0A 1A 0A hexadecimal. The details are given in Table 3.

Table 3. Structure of a *.png file header

Hexadecimal bytes	Purpose
89	For the systems that don’t support 8-bit data, this is a marker to distinguish the *.png file from any other text file
50 4E 47	This is PNG in ASCII, so that it would be recognizable in text editor too
0D 0A	Marking the end of conversion in DOS-Unix style.
1A	Stops the preview under DOS when the command type is used
0A	Marking the end of conversion in Unix style.

As said before, a *.png file can contain several chunks which should not be changed during encryption if we want the output as an image. The most important are so called critical chunks. The chunks are made of four fields: length, chunk type, chunk data and CRC. The critical chunks are: IHDR – the first chunk, PLTE – marks the palette, IDAT – marks the image content and IEND – marks the file end. There are also ancillary chunks which are considered that may not cause problems in the file decoding, but the experience in our research shows otherwise. So, we decided not to change the ancillary chunks. Some of them are bKGD, gAMA, hIST, iCCR, pHYs, sRGB. As explained in the previous sections, we will use the HxD hexadecimal editor to prepare the images for encryption. First, we used CBC mode and block lengths of 8, 16 and

128 bits. The 8 and 16 bit long block is used to observe because the RGB values are represented in 8 bites for each pixel, and sometimes there are additional 8 bits in the pixel to represent the alpha values of the palette. The 128 bit block length is chosen in order to observe the encryptions for longer blocks. We used 20 rounds and different keys while encryption. As before, we used the same original image given in Figure 3 in *.png format and quasigroups number 1, 6, 158 and 181 for the discussed reasons.

The Figures 21, 22 and 23 show the results of the encryption of the original image using the quasugroup 1, CBC mode and block length of 8, 16 and 128 bits.

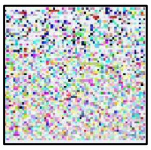


Fig. 19**.** Encryption with quasigroup 1, 8 bit block

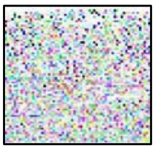


Fig. 20**.** Encryption with quasigroup 1, 16 bit block

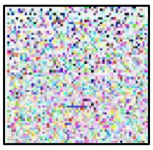


Fig. 21**.** Encryption with quasigroup 1, 128 bit block

The Figures 24, 25 and 26 show the results of same experiments using the quasigroup 6, the Figures 27, 28 and 29 using the quasigroup 158 and the Figures 30, 31 and 32 using the quasigroup 181.

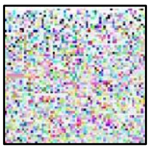


Fig. 22**.** Encryption with quasigroup 6, 8 bit block

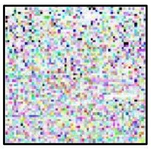


Fig. 23**.** Encryption with quasigroup 6, 16 bit block

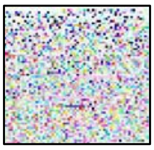


Fig. 24**.** Encryption with quasigroup 6, 128 bit block

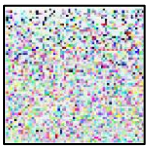


Fig. 25**.** Encryption with quasigroup 158, 8 bit block

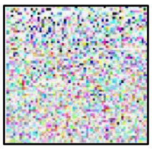


Fig. 26**.** Encryption with quasigroup 158, 16 bit block

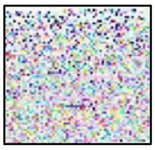


Fig. 27**.** Encryption with quasigroup 158, 128 bit block

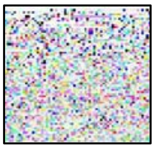


Fig. 28**.** Encryption with quasigroup 181, 8 bit block

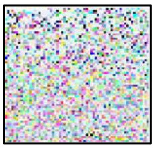


Fig. 29**.** Encryption with quasigroup 181, 16 bit block

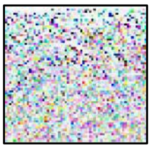


Fig. 30**.** Encryption with quasigroup 181, 128 bit block

It is obvious that no fractal structures appeared using the CBC mode. That is why we tried encrypting with ECB mode. It is clear that if some fractal structures appeared it will be while using the quasigroup number 1, as it is one of the quasigroups with worst properties. Even using the ECB mode we don't see any fractal structures to appear. The *.png file has more complex compression than the *.bmp file so this is the most probable reason that no obvious fractals can be recognized.

6 Conclusion

In this paper we gave the results of research of different types of images using the SQBC block cipher and we can conclude that the differences between the image file formats lead to different results of the encryption. This way, when encrypting the *.bmp images we can clearly distinguish the results of encryption using CBC and ECB mode, because of the very structure of the file and the way that the data is organized within it – every color is represented by 8 bits, making a pixel represented by 24 bits. This means that by encrypting a color the result is also a color. Here, the use of some quasigroups results in fractals in the encrypted image.

Unlike the *.bmp format, the *.png format provides encryptions that cannot be distinguished when using the CBC and ECB mode. Also, no matter which quasigroup we use, a fractal structure doesn't seem to appear. This is probably a consequence of the specific compression and the special palette that the *.png format uses.

As discussed before, the *.jpg format doesn't allow the encryptions to be shown as images, because of the dependencies between the blocks and the special compression. Maybe it will be possible if the design of the block cipher is changed, but that was not our goal.

About the future work, there are still a lot of things to consider about the SQBC block ciphers. These results should be theoretically examined. The research can be expanded to encrypting sound as well, which is currently done. In the future work we can include quasigroups of higher order.

7 References

1. Dimitrova V.: Quasigroup processed strings, their Boolean representations and applications in cryptography and coding theory, PhD Thesis, Skopje, 2005
2. Markovski S., Dimitrova V. and Mileva A.: "SQBC - Block cipher defined by small quasigroups", Loops'11, 2011
3. Trajcheska Z., Petkovska M., Kostadinoski M. and Velkoski G.: Implementation of SQBC in Java, FCSE, Skopje, 2012
4. Dimitrova V., Trajcheska Z., Petkovska M.: Encrypting images with SQBC, The 9th Conference for Informatics and Information Technology (CIIT), Bitola, April 2012
5. http://www.scantips.com/basics9j.html