Towards a simple and secure method for binary cryptography via linear algebra

Licinius Dimitri Sá de Alcantara[1]

Abstract: A simple and secure binary matrix encryption (BME) method is proposed and formalized on a linear algebra basis. The developed cryptography scheme does not require the idealization of a set of complex procedures or the generation of parallel bit stream for encryption of data, but it only needs to capture binary data sequences from the unprotected digital data, which are transformed into encrypted binary sequences by a cipher matrix. This method can be performed on physical or application layer level, and can be easily applied into any digital storage and telecommunication system. It also has the advantage that the encrypted data length is not increased, which avoids additional burden for data storage and transmission. In order to validate the presented methodology, a GNU Octave program code was written to encrypt and decrypt data files.

Keywords: Binary cipher matrix. Binary cryptography. Linear algebra formalism.

Resumo: *Um método simples e seguro de criptografia por matriz binária (BME) é proposto e formalizado em uma base de álgebra linear. A técnica de criptografia desenvolvida não requer a idealização de um conjunto de procedimentos complexos ou a geração de fluxo paralelo de bits para criptografia de dados, mas apenas precisa capturar sequências de dados binários a partir de dados digitais desprotegidos, que são transformados em seqüências binárias criptografadas por uma matriz de cifra. Este método pode ser executado em nível de camada física ou de aplicativo e pode ser facilmente aplicado em qualquer sistema de armazenamento e de telecomunicações digitais. Ele também tem a vantagem de que o tamanho dos dados criptografados não é aumentado, o que evita carga adicional para armazenamento e transmissão de dados. Para validar a metodologia apresentada, um código de programa para GNU Octave foi escrito para criptografar e descriptografar arquivos de dados.*

Palavras-chave: *Matriz de cifra binária. Criptografia binária. Formalismo de álgebra linear.*

1 Introduction

As the telecommunication infrastructure is developing and increasing around the world, privacy is gradually being put into check [1], since pernicious individuals, corporations and even oppressive governments are always ready to take self advantage from the monitoring of unprotected digital data. The use of cryptographic techniques has an essential role on the security of information and transactions details. As described in [2], cryptography enables the user to store confidential information or transmit it across insecure networks so that it cannot be read by anyone except the intended recipient.

Linear algebra theory has a vast number of applications over all fields of engineering. The importance of linear algebra for applications has risen with the development of the computational resources [3]. With the support of modern hardware and software, scientists and engineers now work on multidimensional problems more efficiently than decades before. The knowledge of linear algebra concepts is also relevant to assess issues

[1] Cyberspatial Institute (ICIBE), Federal Rural University of Amazon (UFRA) - 2501 Presidente Tancredo Neves Ave.- Belém (PA) - Brazil
{licinius@ufra.edu.br}

http://dx.doi.org/10.5335/rbca.v9i3.6556

such as existence and uniqueness of solutions in a linear system design. Cryptography is also a research area that can benefit from the linear algebra theoretical background. It is desirable, for instance, to know in which conditions a encipher process, represented as a linear transformation, has a related inverse transformation that fully restores the encrypted data.

There are a great number of encryption methods used for maintain the information secured. Their complexity and ability to resist attack varies from one algorithm to another [4]. The simplest and popular methods are based on XOR (logical exclusive or) cipher techniques [1], where a bit stream is generated to perform XOR operations with the target data bits. The drawback of this technique is that for repeated occurrences of the same encrypted bit sequence, it is very easy to retrieve the key if it is not sufficiently long. Such occurrences are most commonly spaces in text files or silence on audio files. Because of this, many strategies and derivations have been developed to increase the safety of the XOR cipher, which ends up increasing the complexity of the method.

In this paper, a simple and secure encryption technique is proposed, based on established linear algebra concepts and performed by binary cipher matrices. This technique is named here as Binary Matrix Encryption (BME) method and can be readily assimilated and implemented by a programmer or computer engineer. The method acts directly on the binary representation of data, so it can be used to encrypt any type of digitalized information, not only by software, but can also be incorporated into physical layer designs. Until now, the most famous method of cryptography based on linear algebra is the Hill Cipher, which was invented in 1929, when the digital representation of information had not yet been imagined. The Hill Cipher performs the encryption of text messages, dealing with the representation of information at a high level, characterizing itself as a polygraphic substitution cipher. Because of this, there was little interest in applications as well as no greater research efforts into the development of the technique in the field of computer technology. Therefore, a great contribution of this work is to apply the concepts of linear algebra for cryptography at a binary level in a well posed theoretical basis.

Cryptography algorithms can be classified in Symmetric and Asymmetric key cryptography. In symmetric key encryption, only one key is used to encrypt and decrypt data. The key should be distributed before transmission between two parties. The Key characteristics is essential for encryption and decryption. If a weak key is used, the likelihood of the data being decrypted by a spy is increased.

Symmetric algorithms are of two types [5]: block ciphers and stream ciphers. The block ciphers deal with data in groups or blocks. Examples of block ciphers are the Data Encryption Standard (DES) [6], Advanced Encryption Standard (AES) [7] and Blowfish [8]. Stream ciphers encrypt bits individually by adding a bit from a key stream to a bit from the original message. RC4 is a stream cipher algorithm [9].

In asymmetric key encryption, two keys are used; private keys and public keys. Public key is used for encryption and private key is used for decryption (e.g. Digital Signatures). Public key is known to the public and private key is known only to the user. Asymmetric key encryption is used to avoid the need of key distribution. Asymmetric encryption techniques are considerably slower than symmetric techniques, since they require more computational processing resources [10]. The BME method proposed in this work is a symmetric key algorithm of block cipher type.

To summarize the exposition of the model, extensive mathematical discussions about binary or Boolean vector spaces are avoided. This theory is discussed in many papers, as in [11-15]. However, the concepts from linear algebra with practical relevance to this work are addressed. The main focus of this work is to establish a bridge from the linear algebra theory to the consolidation of practical and simple rules for data encryption.

The remainder of this paper is organized as follows. Section 2 describes the most used symmetric key block cipher techniques. Section 3 describes the proposed methodology on the basis of linear algebra theory and the necessary restrictions on the binary matrix that performs the cryptography. The section also discusses why the BME technique is more secure than XOR cipher techniques for bit sequences of the same length. Section 4 shows the GNU Octave program code and the results for the encryption and decryption of a text and an audio file. The conclusions are provided at section 5.

2 Usual cryptography techniques

This section lists and describes the most used symmetric key block encryption techniques and their particularities. None of these methods is related to linear algebra theory, but the most commonly used techniques are similar in the fact that they resort to XOR operations, so that the BME method proposed in this work can be seen as a paradigm break if it becomes widely used.

2.1 DES (Data Encryption Standard)

DES (Data Encryption Standard): DES was the first encryption standard published by the American National Institute of Standards and Technology (NIST) [6], released in 1977. It uses one 64- bit key. Out of 64 bits, 56 bits make up the independent key, which determine the exact cryptography transformation and 8 bits are used for error detection. The main operations are bit permutations and substitution in one round of DES. Six different permutation operations are used in both key expansion stage and cipher stage. Decryption of DES algorithm is similar to encryption, only the round keys are applied in reverse order. The output is a 64-bit block of cipher text. Many attacks and methods recorded the weaknesses of DES, which made it an insecure block cipher key.

2.2 3DES (Triple Data Encryption Standard)

3DES is an enhancement of Data Encryption Standard. It uses 64-bit block size with 192 bits of key size. The encryption method is similar to the one in the original DES but applied three times to increase the encryption level and the data security. 3DES is slower than other block cipher methods. It is essential to avoid having the same key for the encryption steps since the output will only be a slower version of DES [16]. 3DES is very efficient in hardware but not particularly in software. It is popular in financial systems as well as for protecting biometric information in electronic passports [17].

2.3 AES (Advanced Encryption Standard)

AES is also known as the Rijndael's algorithm [7]. At the end of 1990's, It was recognized that DES was not secure because the method was starting to become vulnerable to brute-force attacks due advancement in computer processing resources. The purpose of NIST was to define a replacement for DES that can be used in non-military information security applications by US government agencies. NIST specified the new advanced encryption standard algorithm must be a block cipher capable of handling 128 bit blocks, using keys sized at 128, 192, and 256 bits. It encrypts the data blocks of 128 bits in 10, 12 and 14 rounds, depending on the key size. Its successful use by the U.S. government led to widespread use in the private sector, leading AES to become the most popular algorithm used in symmetric key cryptography. AES encryption is fast and flexible and can be implemented on various platforms especially in small devices [10].

2.4 Blowfish

Blowfish was designed in 1993 by Bruce Schneider [8] as a fast alternative to existing encryption algorithms. The block size is 64 bits, and the key can be any length up to 448 bits. Blowfish has variants of 14 rounds or less. It is a very secure cipher but it is has been replaced by Twofish and Rijndael's due to its small 64 bit block size. Blowfish uses simple operations that are efficient on microprocessors such as exclusive-or (XOR), addition, table lookup and modular-multiplication. It does not use variable-length shifts or bit-wise permutations, or conditional jumps. The technique also makes use of previously computed subkeys. Blowfish is one of the fastest block ciphers that has been developed to date, but it slowness when changing keys prevents Blowfish from being used in some applications. This cryptography method was created to allow anyone to perform encryption free of patents and copyrights. Blowfish has remained in the public domain to this day. No attack is known to be successful against it, though it suffers from weak keys problem [10].

3 The proposed BME method: Materials and methods

This section will focus on the encryption of a binary sequence of two bytes at a time, so this implementation of the proposed method is destined to cipher each sequence of sixteen bits into an encrypted binary sequence of the same size.

3.1 The binary matrix encryption model

Suppose that $\mathbf{b_o}$ is a 16-bit sequence in a column vector disposal, extracted from the original data. Therefore, an encrypted sequence $\mathbf{b_{enc}}$ can be obtained by multiplying a binary cipher matrix **C** by $\mathbf{b_o}$, such as

$$\mathbf{b_{enc}} = \mathbf{C}\mathbf{b_o} \tag{1}$$

Some requirements or rules are needed for setting the binary matrix **C**:

(i) The matrix must have sixteen columns or its matrix multiplication with $\mathbf{b_o}$ could not be performed.

(ii) The columns of **C** must be a linear independent set of vectors, otherwise the linear transformation performed by **C** would not be "one-to-one" [3, 16] and, therefore, different vectors $\mathbf{b_o}$ would generate the same $\mathbf{b_{enc}}$, leading to confusion on the decryption process. This is the uniqueness requirement for the recovering of the original bytes.

(iii) In order to preserve the data size, the encrypted sequence $\mathbf{b_{enc}}$ must also have the same size of $\mathbf{b_o}$, in this case, 16 bits. Furthermore, $\mathbf{b_{enc}}$ cannot have fewer bits than $\mathbf{b_o}$, otherwise the requirement (ii) for **C** could not be fulfilled. Therefore, **C** must have strictly 16 rows in order to generate a 16-bit sequence for $\mathbf{b_{enc}}$. To conform to the linear algebra semantics, this is denoted here as the **T**: $\mathbf{B^{16}} \rightarrow \mathbf{B^{16}}$ requirement, where **T** represents a linear transformation and $\mathbf{B^{16}}$ is the sixteen-dimensional binary vector space. Some caution must be directed to use of the term "binary vector space", since it not obeys all properties of a usual vector space. One of them is that a binary vector space is not infinite, given that $\mathbf{B^{16}}$, for instance, has "only" $2^{16} = 65536$ vectors (different sequences of 16 bits). This justifies the next restriction for the cipher matrix.

(iv) Each row of **C** must not have more than one element with the value "1", or some results for $\mathbf{b_{enc}}$ would not be a binary vector. The in-depth meaning of this rule will be explained later in this section.

Therefore, based on the above requirements, the encrypted matrix **C** is square of order sixteen (the number of bits in $\mathbf{b_o}$) and must formed by a scramble of the identity matrix columns. Suppose that **I** is the identity matrix of order 16, where

$$\mathbf{I} = [\mathbf{I_1}\ \mathbf{I_2}\ \mathbf{I_3}\ \mathbf{I_4}\ \mathbf{I_5}\ \mathbf{I_6}\ \mathbf{I_7}\ \mathbf{I_8}\ \mathbf{I_9}\ \mathbf{I_{10}}\ \mathbf{I_{11}}\ \mathbf{I_{12}}\ \mathbf{I_{13}}\ \mathbf{I_{14}}\ \mathbf{I_{15}}\ \mathbf{I_{16}}], \tag{2}$$

such as $\mathbf{I}_n$ (n = 1, 2,…, 16) is the n^{th} column of **I**. The column $\mathbf{I}_n$ has sixteen elements (one per row), only a element "1" at its n^{th} row and all the remaining elements are "0". The cipher matrix can be obtained by a scramble of the columns of **I**, for example

$$\mathbf{C} = [\mathbf{I_5}\ \mathbf{I_7}\ \mathbf{I_{15}}\ \mathbf{I_{13}}\ \mathbf{I_3}\ \mathbf{I_2}\ \mathbf{I_8}\ \mathbf{I_4}\ \mathbf{I_{16}}\ \mathbf{I_{11}}\ \mathbf{I_{14}}\ \mathbf{I_9}\ \mathbf{I_{10}}\ \mathbf{I_1}\ \mathbf{I_{12}}\ \mathbf{I_6}]. \tag{3}$$

In this case, the columns order "|5|7|15|13|3|2|8|4|16|11|14|9|10|1|12|6|" can be considered as the key for the encryption/decryption process using this cipher matrix. The total of possible permutations or keys is given by 16! = 20922789888000, which is a huge number of keys to be considered by a potential spy, presuming that he knows about the encryption method being used. Some permutations will not be efficient as an encryption key. If **C** = **I**, for example, (1) shows that the encrypted byte will be the same as the original byte and there will not be any encryption at all for the key "|1|2|3|4|5|6|7|8|9|10|11|12|13|14|15|16|". The higher the shuffling of columns of the identity matrix, the stronger the encryption will be.

The elements of the matrix $\mathbf{C} = [c_{ij}]$ are not bits, but instead c_{ij} represents a binary transmission coefficient. If $c_{ij} = 1$, then the j^{th} bit in the original binary sequence ($\mathbf{b_o}$) is transferred to the i^{th} position of the encrypted sequence ($\mathbf{b_{enc}}$). This is another interpretation of why there cannot be more than an element "1" in a

same row of **C**, otherwise two distinct bits in $\mathbf{b}_o$ would be directed into the same position in $\mathbf{b}_{enc}$, and this cannot be expressed as a binary output. Furthermore, two or more bits in the same column of **C** is also not advisable, because the same bit in $\mathbf{b}_o$ would be transferred into two or more distinct positions in $\mathbf{b}_{enc}$, and the remaining bits in $\mathbf{b}_o$ would have less positions to occupy in the encrypted bit $\mathbf{b}_{enc}$, causing information loss or the need to vainly increase the size of $\mathbf{b}_{enc}$ (and the number of lines of **C**) to make room for the redundant bits. However, this would violate the rule (iii), since the size of the encrypted sequence would be higher than the size of the original sequence. Therefore, a encipher matrix **C** formed by a scramble of the identity matrix columns is the only way to grant that all exposed restrictions are fulfilled in order that no bit redundancy or data loss will undermine the decryption process of the encrypted bit sequence. There is not also a miraculous procedure to reduce the size of the encrypted data at the encryption process without bit loss. It can only be achieved if one or more bits of the original sequence are not relevant and then can be discarded.

We obtain the decryption equation by left multiplying each side of (1) by the inverse of **C**, which leads to

$$\mathbf{b}_{dec} = \mathbf{C}^{-1}\,\mathbf{b}_{enc}, \tag{4}$$

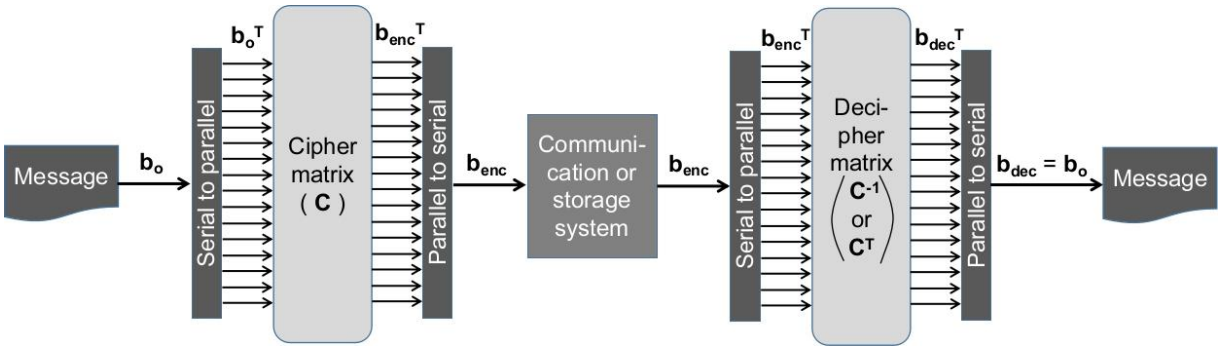
where is expected that $\mathbf{b}_{dec} = \mathbf{b}_o$ after decryption. The use of the inverse of **C** provides another way to explain that **C** must be square with its columns forming a linear independent set of vectors, otherwise **C** would not be invertible.

The inverse linear transform performed by $\mathbf{C}^{-1} = [dij]$ will relocate each bit to its start position in the sequence. Therefore, $d_{ij} = c_{ji}$ and in this case the property $\mathbf{C}^{-1} = \mathbf{C}^{T}$ is valid, where $\mathbf{C}^{T}$ is the transpose of **C**. This occurs because the established cipher matrix is an orthogonal matrix [19], with its rows and columns formed by orthogonal unit vectors. The decipher matrix related to the example in (3) is described by (5). Figure 1 shows a block diagram describing the encryption and decryption processes via the proposed BME method.

$$\mathbf{C}^{-1} = \mathbf{C}^{T} = [\mathbf{I}_{14}\ \mathbf{I}_6\ \mathbf{I}_5\ \mathbf{I}_8\ \mathbf{I}_1\ \mathbf{I}_{16}\ \mathbf{I}_2\ \mathbf{I}_7\ \mathbf{I}_{12}\ \mathbf{I}_{13}\ \mathbf{I}_{10}\ \mathbf{I}_{15}\ \mathbf{I}_4\ \mathbf{I}_{11}\ \mathbf{I}_3\ \mathbf{I}_9]. \tag{5}$$

The presented mathematical basis can be considered an abstraction of any developed procedure for bit scrambling, i.e., even if the scrambling process and the restoration of the binary sequence could be done "manually", the equations (1) and (4) would be subjectively describing these processes, and the "hands" would be fulfilling the role of the matrix **C** as performers of the linear transform that encrypts and decrypts the binary sequence. This means that the mathematical theory interprets the means adopted in practice, and not the reverse. In the electrical circuit theory, for example, it is common to express the output signals from a circuit as a multiplication of a transmission matrix by a vector of input signals. The matrix representation, as the name says, is just a representation. In practice, the output signals are being produced by meshes of elements such as resistors, capacitors, transistors, etc. However, this abstraction allows us to model the problem through a high-level algorithm, without the need of detailing the physical schematics or practical procedures of how it is being accomplished.

Figure 1: Block diagram of the proposed BME method



3.2 Security aspects

The proposed BME method has not the same drawback of the XOR cipher techniques. With the BME it is very difficult to retrieve the key from repetitions of a given sequence of encrypted bits. First, suppose a 16-bit sequence with only zeros (**0**). The encrypted sequence will have also only zeros since it obeys the property **T**(**0**) = **0** inherent in all linear transformations. Therefore, this property is valid for all possible BME keys, because each key represents a cipher binary matrix that performs a linear transformation. On the other hand, the XOR ciphering is not a linear transformation and **T**(**0**) will directly give the key used for encryption to an external observer.

Suppose now that $\mathbf{b_o}$ is a usual binary sequence appearing repeatedly at many points of the original data. As commented before, the total number of keys in the proposed 16-bit BME method is 16! = 20922789888000. However, the total number of possible 16-bit sequences is 2^{16}= 65536, which means that many different keys or linear transformations will generate the same encrypted sequence from $\mathbf{b_o}$. In contrast, the XOR cipher techniques using a stream of 16 bits only have also a total of 65536 keys and each key will generate a different encrypted sequence from $\mathbf{b_o}$. If a spy identifies the repeated encrypted sequence by a spectral analysis and relates them to $\mathbf{b_o}$, he can easily retrieve the key and decipher the entire message. For this reason, XOR cipher techniques use bit streams as long as the message or other strategies are added in order to increase security [1].

Regarding 16-bit sequences, Figure 2 compares the number of ways to generate a specific encrypted sequence ($\mathbf{b_{enc}}$) from a given original sequence ($\mathbf{b_o}$), considering the two cipher techniques. The number of BME keys that perform the same encryption for the same pair of original and encrypted sequence depends on the number of bits "0" and "1" that compose these sequences, which is shown in Table 1, where Num("0") and Num("1") are the number of occurrences of each binary value within these sequences. With the XOR cipher technique, only one key can relate the two sequences at any case. From Table 1 we can conclude that the example shown at Figure 2 is the worst case for the BME technique, i.e., when the occurrence numbers of each binary value are equal, a spy has the best yet very small chance to discover the key.

Figure 2: A comparison between BME and XOR cipher techniques, showing the number of keys that can generate a specific 16-bit encrypted sequence ($\mathbf{b_{enc}}$) from a given 16-bit original sequence ($\mathbf{b_o}$).

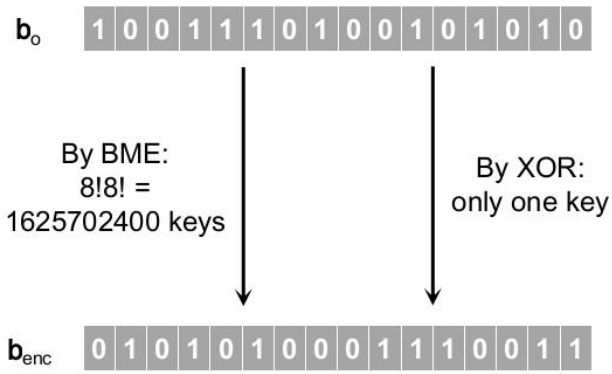


Table 1: Number of BME keys that generates a specific $\mathbf{b_{enc}}$ from a specific $\mathbf{b_o}$.

Num("0")	Num("1")	Number of BME keys
16	0	16!0! = 20922789888000 (all keys)
15	1	15!1! = 1307674368000
14	2	14!2! = 174356582400
13	3	13!3! = 37362124800
12	4	12!4! = 11496038400
11	5	11!5! = 4790016000
10	6	10!6! = 2612736000
9	7	9!7! = 1828915200

8	8	8!8! = 1625702400 (see Figure 2)
7	9	7!9! = 1828915200
6	10	6!10! = 2612736000
5	11	5!11! = 4790016000
4	12	4!12! = 11496038400
3	13	3!13! = 37362124800
2	14	2!14! = 174356582400
1	15	1!15! = 1307674368000
0	16	0!16! = 20922789888000 (all keys)

4 Results and further discussion

This section presents the results for two simulation cases employing the proposed BME method, where the encryption and decryption of a text and an audio file is demonstrated. A program code for GNU Octave was written to achieve this aim.

4.1 Text file encryption by the BME method

The commented program code that applies the proposed method to perform cryptography is shown in this section. The program reads each two bytes of a text file named "original.txt", converts them in a sequence of 16 bits, encrypts the sequence and stores all the encrypted bytes in "encrypted.txt". It also deciphers the encrypted text and stores it in the file "restored.txt". Table 2 shows the content of the original, encrypted and restored text files. Note that many of the encrypted characters falls into the non-readable characters range. Therefore, a cryptanalysis is only possible at a binary level by trying to look for 16-bit patterns over a vast sequence of encrypted bits, making it very difficult to break the cryptography without the exact inverse matrix key.

```
  %This GNU Octave code converts any data file into an encrypted file,
  %using the proposed Binary Matrix Encryption (BME) Method.
  %It also deciphers the message for validation purposes.
  %Author: Licinius Alcantara (November, 2016).

  clear all;  %Clears all previously stored variables.
  fileID1=fopen('original.txt');  %Opens the original text file and returns
                                  %its file identifier (fileID1: integer).
  fileID2=fopen('encrypted.txt','w+'); %Opens a file to store and read the
encrypted data.
  fileID3=fopen('restored.txt','w'); %Opens a file to store the restored
data.

  %% Setting the cipher binary matrix:
  I=eye(16,16);              %I is the Identity matrix of order 16.
  I1=I(1:16,1);  I2=I(1:16,2);  I3=I(1:16,3);  I4=I(1:16,4);    %Extract the
columns of I.
  I5=I(1:16,5); I6=I(1:16,6); I7=I(1:16,7); I8=I(1:16,8);
  I9=I(1:16,9); I10=I(1:16,10); I11=I(1:16,11); I12=I(1:16,12);
  I13=I(1:16,13); I14=I(1:16,14); I15=I(1:16,15); I16=I(1:16,16);
  C=[I5 I7 I15 I13 I3 I2 I8 I4 I16 I11 I14 I9 I10 I1 I12 I6]      %Sets the
cipher matrix by scrambling the columns of I.

  %% Enciphering the message:
  bytecode=fread(fileID1,2)  %Reads two bytes from the original file.
  while ~feof(fileID1)       %feof is a test for end-of-file.
    byteo=de2bi(bytecode,8);   %Represents the 2 bytes as 2 lines of 8 bits
    bo=[byteo(1, 1:8) byteo(2, 1:8)]; %Represents the 2 bytes as a line
vector with 16 bits
```

```
    b_enc = C*bo';     %Obtains the encrypted bit sequence (Eq. 1).
    byte_enc_code1=bi2de(b_enc(1:8)');  %Obtains the encrypted byte 1 code.
    byte_enc_code2=bi2de(b_enc(9:16)'); %Obtains the encrypted byte 2 code.
    fwrite(fileID2,byte_enc_code1);   %Saves the encrypted byte 1 in file.
    fwrite(fileID2,byte_enc_code2);   %Saves the encrypted byte 2 in file.
    bytecode=fread(fileID1,2); %Reads another two original bytes from file.
  end
  fclose(fileID1);

  %% Deciphering the message:
  frewind(fileID2); %Moves the encrypted file pointer to the first byte.
  invC=inv(C);      %Computes and stores the inverse of C in memory.
  byte_enc_code= fread(fileID2,2);    %Reads two encrypted bytes from file.
  while ~feof(fileID2)
   byte_enc=de2bi(byte_enc_code,8); %Stores 2 bytes as 2 lines of 8 bits
   b_enc=[byte_enc(1,1:8) byte_enc(2,1:8)]; % Stores the 2 bytes as a line
vector with 16 bits
   b_dec=invC*b_enc';     %Obtains the deciphered bit sequence (Eq. 4).
   byte_dec_code1=bi2de(b_dec(1:8)');  %Obtains the deciphered byte 1 code.
   byte_dec_code2=bi2de(b_dec(9:16)'); %Obtains the deciphered byte 2 code.
   fwrite(fileID3,byte_dec_code1); %Saves the deciphered byte 1 in file.
   fwrite(fileID3,byte_dec_code2); %Saves the deciphered byte 2 in file.
   byte_enc_code= fread(fileID2,2); %Reads another two encrypted bytes from
file
 end
fclose(fileID3);
fclose(fileID2);
```

Table 2: Contents of the original, encrypted and restored text files. The adopted key was "|5|7|15|13|3|2|8|4|16|11|14|9|10|1|12|6|".

Text file	Text content	Bytes
original.txt	I consider that the Golden Rule requires that if I like a program I must share it with other people who like it. Software sellers want to divide the users and conquer them, making each user agree not to share with others. I refuse to break solidarity with other users in this way. I cannot in good conscience sign a nondisclosure agreement or a software license agreement. So that I can continue to use computers without dishonor, I have decided to put together a sufficient body of free software so that I will be able to get along without any software that is not free. (Quotations from The GNU Manifesto, Richard Stallman, 1985)	638
encrypted.txt	##◆#ƽ◆□8◆N#*◆◆◆@◆I◆@◆◆x◆m##◆i◆@Ǩ◆◆◆#◆◆#*◆◆◆@◆<#◆#) ◆◆◆@◆#◆#◆◆Lj◆P◆#◆◆◆*#◆◆◆Ǩ#◆◆@◆◆I#◆◆I◆N# ◆◆)◆@◆I◆P◆◆Ы#◆◆e#◆◆|◆◆#◆@◥◆y◆N◆#◆◆◆z#*◆P◆◆◆B◆#*◆◆# ◆◥ă#◆◆x#◆◆}◆◆◆N#*◆◆◆q#◆◆◆◆=◆@◆◆◆ #◆◥◆#◆◆Ǩ◆@◆◆◆@◆◆◆#◆◆◆Ǩ#◆◆◆◆#◆z◆◆ăCP◆#Ǩ◆◆◥#*◆P◆ #◆◆◆#ك□□◆lj◆◆#◆◆◆◆#◆z◆◆◆#◆ΓN◆#◆=#*◆◆◆#◆û5#◆#◆◆ ◆◆@◆=#◆◆◆@ƽ◆◆H◆mO#◆◆◆◆P◆#◆◆◆x◆◆◆)◆□N◆@◆◆Ǩ◆◆m◆@ ◆^#◆#◆◆|◆◆#◆@◆◆◆O◆□@◆◆Ǩ◆◆m◆eA##◆◆P◆I◆*#◆#◆◆ #◆◆}◆◆◆◆@◆◆#◆◥#◆◆◆◆ă#◆◆◆◆◆j#(◆◆◆◆◆◆◆!#◆# ◆◆@◆◆H◆◆@◆◆# ◆j#*◆◆j◆◆◆#◆#Л◆I◆◆◆◆◆z#	638

restored.txt	I consider that the Golden Rule requires that if I like a program I must share it with other people who like it. Software sellers want to divide the users and conquer them, making each user agree not to share with others. I refuse to break solidarity with other users in this way. I cannot in good conscience sign a nondisclosure agreement or a software license agreement. So that I can continue to use computers without dishonor, I have decided to put together a sufficient body of free software so that I will be able to get along without any software that is not free. (Quotations from The GNU Manifesto, Richard Stallman, 1985)	638

4.2 Digital audio encryption example

The same program exposed in the subsection 3.1 is also used to encrypt and decrypt an audio file. The only changes in the program code are on the file names. An audio file named “original.wav” is encrypted into a file named “encrypted.wav” and the decryption result is stored in “restored.wav”. Table 3 shows the file contents after execution of the GNU Octave code.

Table 3: Contents of the original, encrypted and restored audio files.

Audio file	Audio content	Kbytes
original.wav	Can you keep a secret?	229,6
encrypted.wav	<unrecognizable audio format>	229,6
restored.wav	Can you keep a secret?	229,6

As reported in Table 3, the format of the generated encrypted file is not recognizable by any audio reproducer software. This is due that even the file header, which contains the format description of the file, was encrypted. However, the original file was fully restored after the deciphering process.

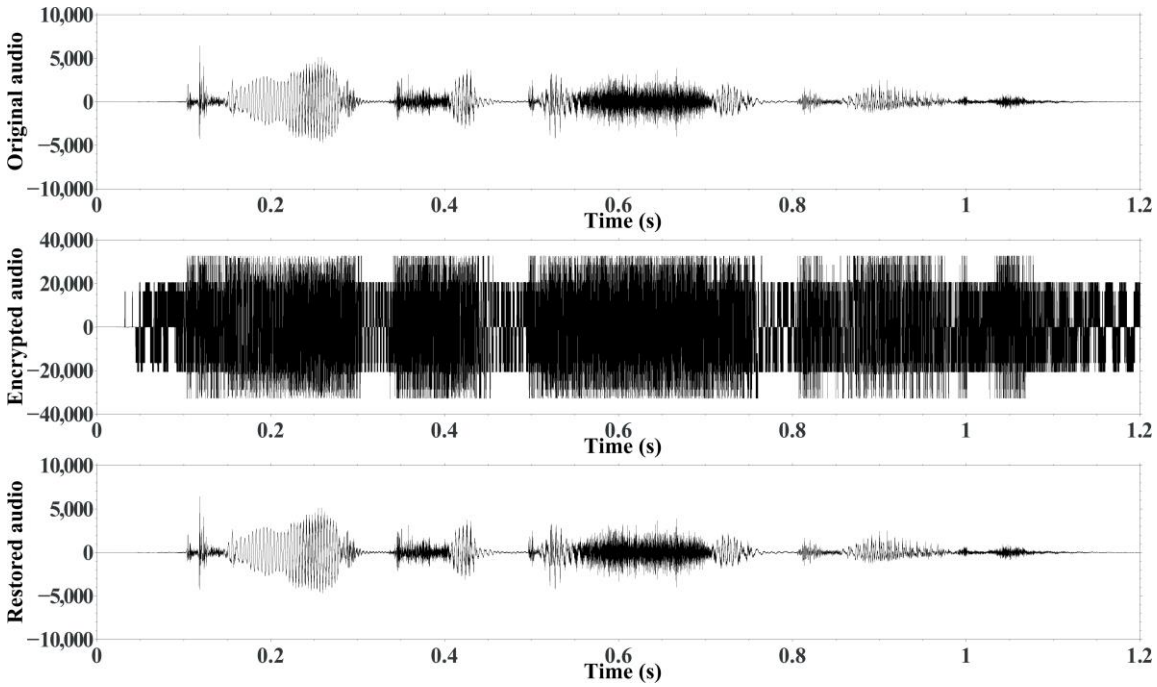
Table 4: The RIFF Waveform Audio File Format (.wav).

File offset (bytes)	Field	Size (bytes)	Description
0	Chunk identification	4	RIFF format information
4	Chunk size	4	
8	Format	4	
12	Sub-chunk 1 ID	4	Audio format information
16	Sub-chunk 1 size	4	
20	Audio format	2	
22	Number of channels	2	
24	Sample rate	4	
28	Byte rate	4	
32	Block align	2	
34	Bits per sample	2	
36	Sub-chunk 2 ID	4	Audio data sub-chunk
40	Sub-chunk 2 size	4	
44	Audio Data	Sub-chunk 2 size	

For illustration purposes, in order to verify if just the digitalized audio data is strongly encrypted, the GNU Octave program code was modified to only encrypt the audio file after its header. In this way, only the audio data bytes are encrypted and thus, the file remains recognizable as an audio file. Table 4 shows the standard structure of Waveform Audio File Format (.wav) [20]. This is a file format for the storing of audio data in data chunks according to the Resource Interchange File Format (RIFF) [21]. It can be seen from Table 4 that

the file header contents sums 44 bytes. Therefore, the encryption was set to start at the 45th byte. The new generated encrypted audio file is now recognizable by audio software, but the encrypted audio has not recognizable voice content, so the aim was achieved concerning the raw audio data. Figure 3 shows the graphical representation of the audio signals from the original, encrypted (except the bytes of the file header) and restored wave files.

Figure 3: Original, encrypted and restored audio signals with the BME method.



4.3 Comparison between the BME and the XOR cipher methods.

As was described at Section 2, the most usual techniques resort to XOR operations to perform cryptography, but includes many rounds, uses other procedures on the bit sequence and considers larger blocks of data to strengthen the key. In order to perform fair comparisons between the BME method and the XOR cipher by adopting the same parameters, a GNU Octave program was also developed to apply the traditional XOR cipher in blocks of 16 bits of information and adopting an arbitrary 16-bit key to do XOR operations with each block from the original data in only one round. The target file was “original.txt”, shown at Table 2. The XOR cipher program also have encrypted and decrypted the file successfully. Both programs ran in a computer Intel$^{(R)}$ Core$^{(TM)}$ I5-5200U with 2.2GHz of clock speed and the Manjaro Linux operational system. The BME program ran in an average time of 236.87ms and the XOR cipher program ran in average time of 240.81ms. Therefore, the methods have shown similar efficiency in processing speed terms. However, if it is considered that methods such as DES, AES and Blowfish uses many rounds and other operations over the data blocks, it can be fairly inferred that the BME method outperforms these methods in processing speed.

5 Conclusions

A simple and secure binary matrix encryption (BME) method was formalized on arguments and restrictions based on linear algebra concepts. The method basically multiplies a cipher matrix by binary sequences extracted from the original data, resulting in an encrypted binary sequence. The cipher matrix is generated from a scrambling of the identity matrix columns. The method can be used to encrypt and recover any digital information with the appropriate key. A GNU Octave program code was developed and used to encipher/decipher a text and an audio file, where the proposed methodology was successfully tested. The size of the encrypted information is not increased compared to the original information. The number of the total keys

that can be used to encrypt a binary sequence of two bytes was estimated to be a huge value, so that a chosen key is difficult to be discovered by an unexpected spy. The number of total encryption keys can be hugely more increased if more than two bytes are grouped in a single increased binary sequence, to be encrypted by a higher order cipher matrix. Note also that the proposed BME and the XOR cipher technique are not mutually exclusive. They could be jointly applied to a bit sequence of same length to difficult even more a cryptanalysis of the protected data and a spy would have a very hard time trying to discover both the BME and the XOR cipher keys, even if he discovers repeated occurrences of the same encrypted bit sequence.

Acknowledgements

The author would like to thank the free software community for distributing high quality software that encourages research and scientific cooperation in many fields. Some of these software were used during the development of this work.

References

[1] STALLINGS, W. (Ed.). *Cryptography and Network Security*. Upper Saddle River, NJ, USA: Prentice Hall, 2005.

[2] ZIMMERMAN, P. (Ed.). *An Introduction to Cryptography*. Santa Clara, CA, USA: Network Associates, Inc., 1999. Available at: <ftp://ftp.pgpi.org/pub/pgp/7.0/docs/english/IntroToCrypto.pdf>. Accessed at 14 Nov. 2016.

[3] LAY, D. C. (4th Ed.). *Linear Algebra and its Application*. Boston, MA, USA: Pearson, 2012.

[4] AL-HAZAIMEH, O. A New Approach for Complex Encrypting and Decrypting Data. *International Journal of Computer Networks & Communications*, AIRCC Publishing Corporation, v. 5, n. 2, p. 95–103, 2013. ISSN 0974-9322. Available at: <http://airccse.org/journal/cnc/5213cnc08.pdf>. Accessed at 14 Nov. 2016.

[5] STALLINGS, W. (Ed.). *Network Security Essentials (Applications and Standards)*, Pearson Education, 2004.

[6] FIPSP 46-3. *DES Encryption Standard*, USA, 1999. National Technical Information Service. Available at: <https://csrc.nist.gov/csrc/media/publications/fips/46/3/archive/1999-10-25/documents/fips46-3.pdf>. Accessed at 14 Nov. 2016.

[7] FIPSP 197. *Specification for the Advanced Encryption Standard (AES)*, USA, 2001. National Technical Information Service. Available at: <http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf>. Accessed at 14 Nov. 2016.

[8] FERGUSON, N.; SCHNEIER, B.; KOHNO, T. (Ed.). *Cryptography Engineering: Design Principles and Practical Applications*. New York, NY, USA: John Wiley and Sons, 2010.

[9] MOUSA, A.; HAMAD, A. Evaluation of the RC4 Algorithm for Data Encryption, *International Journal of Computer Science & Applications*, Technomathematics Research Foundation, v. 3, n. 2, p. 44-56, 2006. ISSN 0972-9038. Available at: <http://www.tmrfindia.org/ijcsa/v3i24.pdf>. Accessed at 14 Nov. 2016.

[10] JUNEJA, M.; Sandhu, P. S. A Review of Cryptography Techniques and Implementation of AES for Images. International Journal of Computer Science and Electronics Engineering. International Scientific Academy of Engineering and Technology, v. 1, n. 4, p. 478-482, 2013. ISSN 2320-401X. Available at: <http://www.isaet.org/images/extraimages/L813087.pdf>. Accessed at 14 Nov. 2016.

[11] JAGANNADHAM, P. V. Linear transformations on Boolean vector spaces. *Mathematische Annalen*, Springer-Verlag, v. 167, n. 3, p. 240–247, 1966. ISSN 0025-5831. Available at: <http://link.springer.com/article/10.1007/BF01361188>. Accessed at 14 Nov. 2016.

[12] ARAI, K.; OKAZAKI, H.; SHIDAMA, Y. N-dimensional Binary Vector Spaces. *Formalized Mathematics*, De Gruyter, v. 21, n. 2, p. 75–81, 2013. ISSN 1898-9934. Available at: <https://www.degruyter.com/view/j/forma.2013.21.issue-2/forma-2013-0008/forma-2013-0008.pdf>. Accessed at 14 Nov. 2016.

[13] LUCE, R. D. A note on Boolean matrix theory. *Proceedings of the American Mathematical Society*, AMS, v. 3, n. 2, p. 382–388, 1952. ISSN 1088-6826. Available at: <http://www.ams.org/journals/proc/1952-003-03/S0002-9939-1952-0050559-1/S0002-9939-1952-0050559-1.pdf>. Accessed at 14 Nov. 2016.

[14] RUTHERFORD, D. E. Orthogonal Boolean matrices. *Proceedings of the Royal Society of Edinburgh*, Cambridge University Press, v. 67, n. 2, p. 126–135, 1965. ISSN 0308-2105. Available at: <journals.cambridge.org/article_S0080454100007974>. Accessed at 14 Nov. 2016.

[15] GUTHER, S.; LATREMOLIERE, F. Boolean inner-product spaces and Boolean matrices. *Linear Algebra and its Applications*, Elsevier Inc., v. 431, n. 1, p. 274–296, 2009. ISSN 0024-3795. Available at: <http://www.sciencedirect.com/science/article/pii/S0024379509001190>. Accessed at 14 Nov. 2016.

[16] ALEISA, N. A Comparison of the 3DES and AES Encryption Standards. *International Journal of Security and Its Applications*. SERSC, v. 9, n. 7, p. 241-246, 2015. ISSN 1738-9976. Available at: <http://www.sersc.org/journals/IJSIA/vol9_no7_2015/21.pdf>. Accessed at 14 Nov. 2016.

[17] PAAR, C.; PELZL, J. (Ed.). *Understanding Cryptography: A Textbook for Students and Practitioners*. Bochum, Germany: Springer-Verlag, 2010.

[18] ANTON, H. (10th Ed.). *Elementary Linear Algebra*. Hoboken, NJ, USA: John Wiley & Sons, 2010.

[19] STEWART, G. W. The efficient generation of random orthogonal matrices with an application to condition estimators. *SIAM Journal on Numerical Analysis*, Society for Industrial and Applied Mathematics, v. 17, n. 3, p. 403–409, 1980. ISSN 0036-1429. Available at: <http://www.jstor.org/stable/2156882>. Accessed at 14 Nov. 2016.

[20] PETZOLD, C. Storing Sound: A Look at Waveform Audio Sound Files. *PC Magazine*, Ziff Davis Publishing Company, v. 11, n. 3, p. 369–374, 1992. ISSN 0888-8507. Available at: <https://books.google.com.br/books?id=FeIuiOQN-nEC&pg>. Accessed at 14 Nov. 2016.

[21] ITU-R BS.2088-0. *Long-form file format for the international exchange of audio programme materials with metadata*, Switzerland, 2016. International Telecommunication Union. Electronic Publication. Available at: <https://www.itu.int/dms_pubrec/itu-r/rec/bs/R-REC-BS.2088-0-201510-I!!PDF-E.pdf>. Accessed at: 16 Nov. 2016.