A NEW CRYPTOSYSTEM WITH FOUR LEVELS OF ENCRYPTION AND PARALLEL PROGRAMMING

Parag A. Guruji

Department of Computer Science and Engineering,
Walchand College of Engineering Sangli (An Autonomous Institute)
Vishrambag, Sangli, Maharashtra, India- 416415
gurujipa@gmail.com

ABSTRACT

Evolution in the communication systems has changed the paradigm of human life on this planet. The growing network facilities for the masses have converted this world to a village (or may be even smaller entity of human accommodation) in a sense that every part of the world is reachable for everyone in almost no time. But this fact is also not an exception for coins having two sides. With increasing use of communication networks the various threats to the privacy, integrity and confidentiality of the data sent over the network are also increasing, demanding the newer and newer security measures to be implied. The ancient techniques of coded messages are imitated in terms of new software environments under the domain of cryptography. The cryptosystems provide a means for the secured transmission of data over an unsecured channel by providing encoding and decoding functionalities. This paper proposes a new cryptosystem based on four levels of encryption. The system is suitable for communication within the trusted groups.

KEYWORDS

Matrix transformation, Fractionification, Re-integerization, Change of radix

1. INTRODUCTION

A cryptosystem refers to a suite of algorithms needed to implement a particular form of encryption and decryption. The encryption operations are the transformation functions with the set of all symbols which appear in data to be encrypted as their domain and the set of all corresponding encoded symbols as their codomain. The basic characteristic of any encryption operation for the faithful transmission of data is its reversibility. Any encryption operation that transforms input data into some encoded form must work as a bijective mapping, whose inverse exists and is also a bijective mapping. These criteria if not satisfied, the retrieval of the data from its encoded form back to its original form cannot be assured. Following figure represents the encryption operation f and its inverse f^{-1} (called decryption operation) as the bijective mappings from their corresponding domain and codomain.

David C. Wyld et al. (Eds) : CSEN, AISO, NCWC, SIPR - 2015
pp. 11–19, 2015. © CS & IT-CSCP 2015 DOI : 10.5121/csit.2015.51402

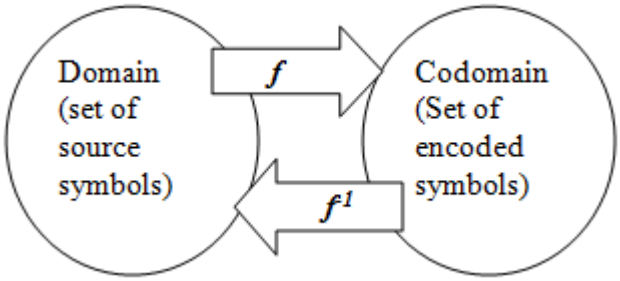


Figure 1. The encryption and decryption operations as mappings

In this paper the author proposes a new cryptosystem for the implementation in form of an application able to perform all the encryption and decryption tasks in an abstracted manner and thus keeping all of them transparent to only the valid user. The system operates on four levels of layers of the encryption making the complexity of cracking it extremely high. The four layers refer to the different set of operations, undergoes which the user data. The fragmentation and re-organization of the data is to be done as preprocessing before passing it to the encryption module. On the other side the decryption module works for the retrieval of encrypted data from the chunks that it receives and reorganizes it by sorting the randomly received chunks; after performing the four decryption operations on it which are inverses of the four encryption operations.

The encryption operations are:

1. Matrix transformation
2. Fractionification
3. Random no. addition
4. Change of radix

The decryption operations are:

1. Change of radix
2. Random no. removal
3. Re-integerization
4. Matrix re-transformation

The key generation operations are:

1. Random no. generation
2. Matrix generation using corresponding polynomial and checking for its inevitability
3. Radix generation using corresponding polynomial.

2. THE FUNCTIONAL DESIGN

2.1. Defining Tasks

To perform the operations in a systematic manner, the author defines the tasks to be performed on both the sides, the encryption and decryption as follows:

2.1.1. Encryption Tasks

1. Fragmentation of input data into chunks and indexing them.
2. Generation of the random key for each chunk and thus that of the key matrix and key radix.
3. Operating each chunk with encryption operations in the sequence in which they are listed above.
4. Augmentation of encrypted chunk with corresponding key which is a mere random integer.

2.1.2. Decryption Tasks

1. Receipt of the encrypted chunk and separation of key
2. Generation of the key matrix and key radix for received chunk.
3. Operating each chunk with decryption operations.
4. Reorganization of the chunks using the indices to retrieve data in its original form.

The selection/formation of polynomials required for the key generation are left on the implementation to keep this design flexible. The complexity of these polynomials will add to the complexity of whole of the system.

2.2. Task Accomplishment Scheme

The scheme for completing each of the above tasks is discussed in this section.

2.2.1. Fragmentation of input data into chunks and indexing:

The input data is fragmented in the chunks, each of size s bytes where s is the implementation-specific size defined for representation of an integer. The data structure to be used store these fragments is a linear list, each node of which contains a chunk and an index value representing the offset of that chunk from the beginning of the input data in terms of no. of chunks. Along with these two values, the chunk contains space for its key value, a random integer generated by the key generator. This fragmentation of the data enables the parallel functioning of every step to follow as discussed later in the paper.

2.2.2. Generation of the random key for each chunk and thus that of the key matrix and key radix:

A random number is to be generated (generation implementation specific) for each chunk and is then assigned as the key for that chunk. The selected polynomials are provided with this key to generate the key matrix and key radix for that chunk. The implementation must take care that the generated matrix will be an invertible (non singular) matrix. After completion of this step we are ready with required input values for the computation of the encrypted counterparts of each element in the input data.

2.2.3. Operating each chunk with encryption operations:

2.2.3.1. Matrix transformation:

The chunk formed along with the source file identifier (A random no. assigned to the source-file) and excluding the key is represented as a 3×1 matrix and is multiplied with the 3×3 matrix generated using the key (key matrix) to get the transformed matrix of order 3×1.

2.2.3.2. Fractionification:

The term Fractionification is defined as the conversion function which maps an integer to a fraction by dividing the integer by R^d where R is the radix of the number system under consideration and d is no. of significant digits in the original integer and then adding to it the integer value d. Thus, for an integer I in number system with radix R having d significant digits, fractionification f is defined as,

$$f(I) = I \div (R^d) + d$$

2.2.3.3. Random no. addition:

The fractionified no is then added with some random number multiplied by 10 to preserve the value of d (the no of significant digits in original no.). Thus, the integer I when fractionified and added with random no. becomes $r(I)$ given by,

$$r(I) = f(I) + n \times 10$$

where, n is the random number generated.

2.2.3.4. Change of radix:

Now that we have converted the integer I, representing s bytes of input data, to a floating point equivalent $r(I)$, the radix of the number system is to be changed as the outermost encoding operation. It is defined as the combination of two simple radix conversion operations, one for the integer part of the input floating point no. and other for its fraction part, represented as an integer. The target radix selection is important task and is selected using a randomization polynomial (implementation specific) with the key of corresponding chunk as its parameter. To use radix greater than 10, the corresponding symbols used are capital and small scripts of English alphabets and related numerical operations on them are to be defined.

2.2.4. Augmentation of encrypted chunk with corresponding key:

Once each element in the chunk except the key are encrypted, the chunk is augmented with the key, applied with fractionification and random no. addition, and thus is ready for the transmission.

2.2.5. Receipt of the encrypted chunk and separation of key:

The chunk when received on decryption end, it is to be stored in the buffer for unresolved chunks. From the key field the value of key is found and separated out and the corresponding key matrix and key radix are calculated exactly as explained above. The inverse of this matrix is calculated by adjoint method to get the decryption matrix.

2.2.6. Operating each chunk with decryption operations:

2.2.6.1. Change of radix:

Each element in the received chunk is operated on by the inverse change of radix with source radix as the one derived from the key and 10 as the target radix. Obviously, the integer and fraction part are treated individually treated as different integers, and then combined back.

2.2.6.2. Random no. removal:

Each element of the chunk is then operated upon by the inverse of the random no. addition to get the fractionified value using following function,

$$f(I) = (r(I)\%10)$$

2.2.6.3. Re-integerization:

The term Re-integerization is defined as the inverse function of Fractionification which maps to an integer, its equivalent fractionified value, and is defined as,

$$I = (f(I)\ \%1) \times 10^{d}$$

where,

$$d = \lfloor (f(I)\%10) \rfloor$$

2.2.6.4. Matrix re-transformation:

The chunk received is in the form of 3×1 matrix. It is multiplied by the 3×3 decryption matrix determined for that chunk according to the simple matrix multiplication to get the original data chunk.

2.2.7. Reorganization of the chunks

Now that having done with the decryption operations on received chunks, they are to be reorganized in the sequence of that of the data contained by them in the original source file. This is achieved by sorting the randomly placed data chunks using the identifier and index fields as the key. To boost the efficiency of sorting, author proposes to form a Binary Search Tree for each identifier and then the chunks are to be added in it according to the index field values as the key. Once the no. of nodes in the tree approaches to the total count of chunks present in the index field

of identifier node, i.e. the root, the tree is traversed in In-order manner (left-root-right) and data field contents of each node are written into the destination file during traversal.

3. SECURITY FACTORS

The security and confidentiality of the data are the fundamental goals of any cryptosystem. In case of the proposed system, though all of such factors already have appeared in the discussion up till now, in this section we identify and enlist each of them for the getting the view of the security provided by the system as a whole.

1. Randomness of the key

2. Secrecy and complexity of the polynomials used for matrix and radix generation

3. Individual random key for each chunk: This removes the threat by many of the pattern analysis and known text attacks

4. Matrix Transformation: This transforms chunk into an integral unit whose meaning cannot be derived without accurate inverse of key matrix

5. Fractionification and Random no. addition: This covers the transformation and makes it too complex to analyze the resultant patterns and detect the transformation

6. Change of Radix: This changes the representation of the numbers and thus adding to the complexity of analysis of interrelations of elements in resultant values.

4. THE PARALLEL PROGRAMMING APPROACH

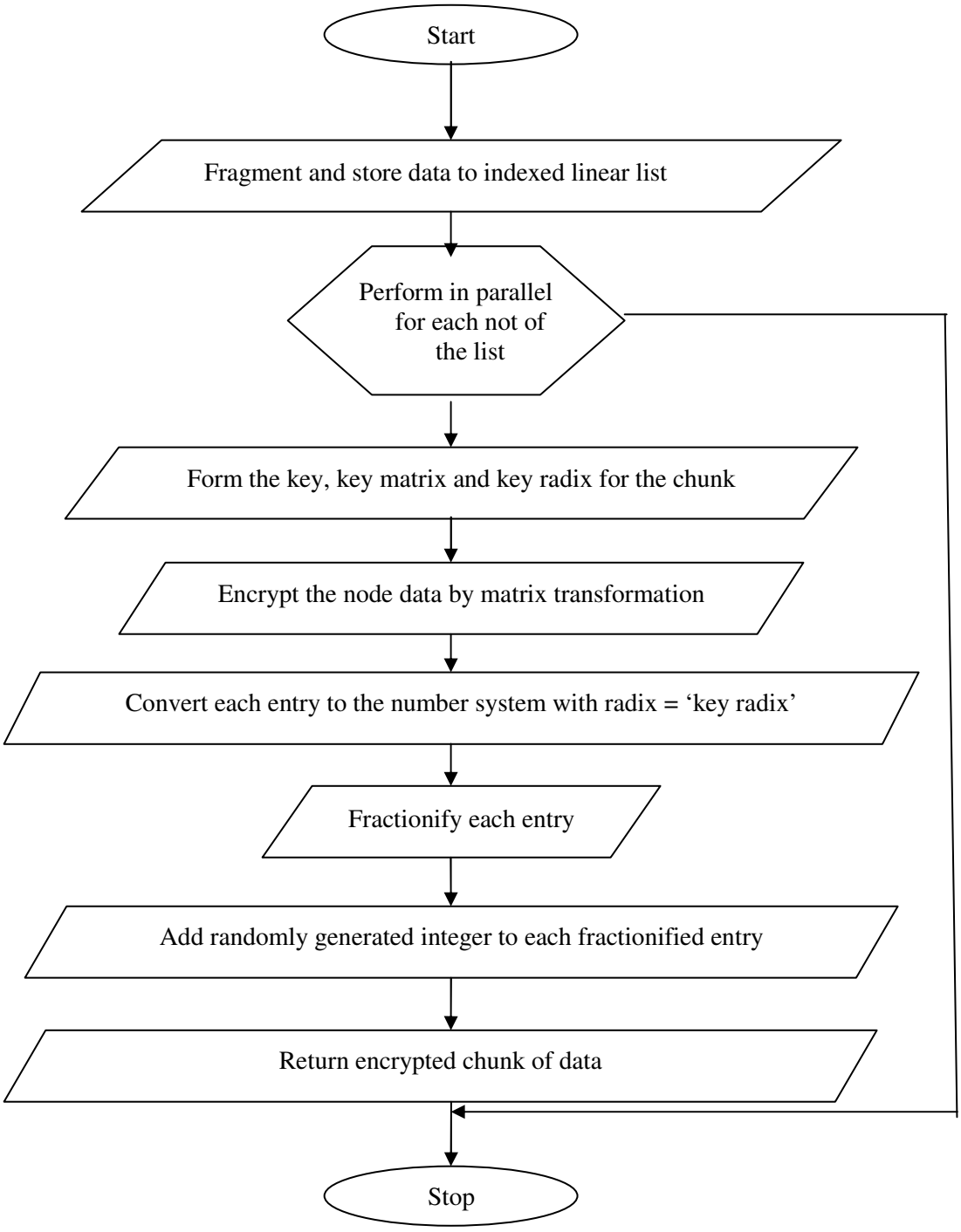
The important feature of proposed design of cryptosystem in this paper is the fragmentation of data and independency of the key for each fragment. This independence allows the parallel functioning of different modules in of cryptosystem. Each node follows the same path after the fragmentation is done. Thus after completion of Task 1on encryption side, each chunk is proposed to be processed in parallel through the completion of encryption. Also on decryption side, the received nodes are proposed to get processed in parallel till their addition to corresponding BST. This will reduce the time complexity of the cryptosystem application by the factor of n^{-1} where, n is the no. of fragments.

Along with this first level of parallel programming, the efficiency can further be increased by incorporating the second level of the same. In the second level, within each fragment the different elements are proposed to be operated with all the encryption as well as decryption functions independently except the matrix transformation. This will bring the time complexity to Time complexity of matrix multiplication + 3^{-1} (time complexity of rest of the operations with sequential approach), Thus reducing it roughly by factor of 3-1.

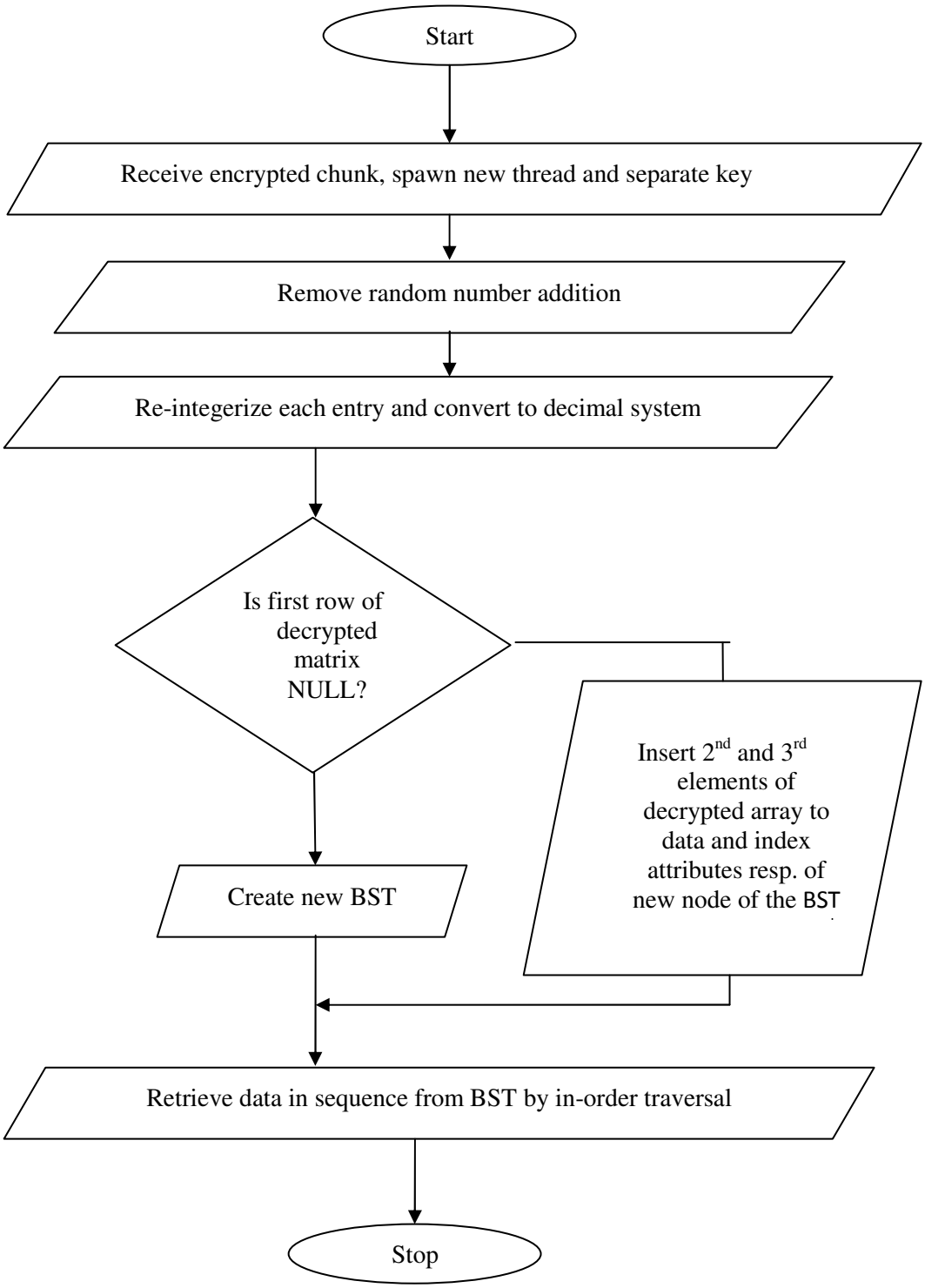
The author further proposes the third level of parallel programming, involving the parallel implementation of the matrix multiplication itself to further boost the efficiency. Thus the parallel programming approach adds to the efficiency significantly.

5. FLOWCHARTS

4.1. Encryption Flowchart:



4.2. Decryption Flowchart:



5. CONCLUSIONS

The cryptosystem proposed in the paper works on four different layers of the encryption. All the layers cover the possible attacks on its inner layer making the encryption extremely complex to crack. The security factors of the system protect it against the cracking attacks. The polynomials and random number generators are left to the implementation for making the system flexible. This incurs the variation of complexity of encryption depending on the implementation. The parallel approach of programming adds to the efficiency of application significantly, as discussed in the section II.

REFERENCES

[1] Yi-Shiung Yeh, Tzong –Chen Wu, Chin Chen Chang and Wei Chizh Yang "A New Cryptosystem using Matrix Transformation", Proceedings. 25th Annual IEEE International Carnahan Conference on Security Technology 1991 (Cat. No.91CH3031-2)

[2] D. C. Lay "Linear Algebra" ISBN: 9781405846219, Chapters 1, 2 and 3

AUTHORS

Parag A. Guruji

Earned Bachelor of Technology degree in

Computer Science and Engineering from

Walchand College of Engineering, Sangli, India

in May 2014.

Working at ZLemma Analytics in Data Science

team since June 2014