VOL. 3, NO. 5, OCTOBER 2008 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences


www.arpnjournals.com

INTEGRATING ERROR DETECTION WITH DATA ENCRYPTION ALGORITHM USING PERMUTATION INVARIANT RAO ALAKA SHIFT TRANSFORM

A. V. Narasimha Rao[1], K. Soundara Rajan[2] and K. Srinivasa Rao[1]
[1]Department of Electronics and Communications Engineering, Chaitanya Bharati Institute of Technology, Gandipet, Hyderabad, India
[2]College of Engineering, JNT University, Ananthapur, Andhra Pradesh, India
E-Mail: jupiter02in@gmail.com

ABSTRACT

This paper adopts a novel approach for ensuring security of data with error detection capability. RAS Transform is a nonlinear recursive Transform. This simple but very effective RAS transform is Permutation Invariant and used to code the digital data at two levels, so that the data is encrypted and also there is multilevel error detection mechanism based on the properties of the RAS Transform. The first type is data independent and the later is data dependent. In data dependant encryption, the partially encrypted data is subjected to RAS Transformation at two levels namely, byte level and block level before transmission. The outcome is 128 bits of encrypted data together with Encryption Key. A code book of only 20 valid code words is generated to represent 256 possible octets of 8-bit data words. From each of the code words, the data word can be uniquely recovered using the data dependent symmetric encryption key. The result of this coding on a sample text data of about 189 characters size is presented.

Keywords: data, communication, security, error detection, encryption, RAS transform, coding, key.

INTRODUCTION

In general, coding is considered as a mechanism for improving the integrity of the data communication system in the presence of noise. The use of coding for security is encryption and ensures security of information. Any Error detection coding is designed to permit the detection of errors. Once detected, the receiver may ask for a re-transmission of the erroneous bits or frame. In telecommunication, a redundancy is extra data added to a message for the purposes of either error detection or encryption. Both these coding processes introduce redundancy in to the digital data. Several schemes exist to achieve error detection, and are generally quite simple. In a binary channel, majority of practical codes are block codes. Most codes are "systematic": the transmitter sends a fixed number of original data bits, followed by fixed number of check bits (usually referred to as redundancy) which are derived from the data bits by some deterministic algorithm. The receiver applies the same algorithm to the received data bits and compares its output to the received check bits; if the values do not match, an error has occurred at some point during the transmission. A block code converts a fixed length of K data bits to a codeword of fixed length N, where N > K. The block of encoded message consists of K message bits in addition to N-K error check bits. The rate of the code is the ratio K/N, and the redundancy of the code is 1-(K/N). Our ability to detect errors depends on the rate. A low rate has a high error detection probability, but a high redundancy. At the same time a low rate of code speaks about high security for data. Error correction coding requires lower rate codes than error detection, often markedly so. It is therefore uncommon in terrestrial communication, where better performance is usually obtained with error detection and retransmission. In a system that uses a "non-systematic" code, such as some raptor codes, data bits are transformed into many code bits, and the transmitter sends only the code bits. Encryption of digital data is the only way to protect the privacy and guarantee the success of Digital marketplace**.** The process of encryption involves some operations, or set of operations, which gives the step by step procedure for transforming the data by using one or more keys. This arrangement is called an algorithm. An encryption algorithm must be associated with reverse procedure, known as decryption algorithm, which restores the original message with the help of key(s). Taken together, the techniques of encryption and decryption are called cryptography. The principle of encryption is something similar to that of a combination lock. It is very much easier, with the use of digital computers, to encrypt than it is to decipher. With a computer, the number of the digits in the lock can be very large. Of course, one still has to keep the combination secure! The most commonly used encryption algorithms are block ciphers. In these algorithms the plaintext (message to be encrypted) is split into (usually) fixed size blocks which are subjected to various permutations based on expansion, compression, straight or rule based shuffle/ shift transformation functions to produce a block of cipher text (Encrypted message).Encryption methods have been historically divided in to two categories: Substitution ciphers and Transposition ciphers. In substitution cipher, each symbol or group of symbols (letters) of the message is replaced by another symbol or symbols and the message is concealed. Here, the order of the plain text symbols is preserved but disguised. Incase of transposition ciphers, message symbols are reordered in such a way that message can not be interpreted. The third well known class of ciphers is Transformation ciphers. In these ciphers, based on some sequence of one or more mathematical or logical

functions, each of the symbols or group of symbols of message is transformed or changed both in the order and nature. All these methods are in use and each has its own advantages and disadvantages.

Similar to that of a combination lock, if the same key is to be used for encryption and decryption, such encryption algorithm is known as symmetric key encryption algorithm. In symmetric encryption algorithms, the system is only secure if the key is secure. If the key is changed often, the security of the key becomes a problem, because the transfer of the key between sender and receiver may not be secure. This is avoided to a considerable extent by the use of matched keys. In a matched key scheme, the encryption is not reversible with the same key. The message is encrypted using one key, and decrypted with a second matched key. Such encryption algorithm based on two different keys one for encryption and the other for decryption is called asymmetric key encryption or some times called public key encryption.

There are two principles underlying all of different cryptographic systems that are very important to understand and incorporate. The first principle is that all encrypted message must contain some redundancy. That is, the information which does not contain the message. This prevents the active intruders from tricking the receiver into acting on a false message. The second principle is that some measures need to be adopted to prevent active intruders from playing back the old messages. As a result, even in the good encryption algorithms, the cipher text is made about 2.5 to 3 times longer than the actual message.

REVIEW OF EXISTING SYSTEMS

Repetition schemes, parity check schemes and checksum methods are very simple techniques for error correction. More complex error detection (and correction) methods make use of the properties of finite fields and polynomials over such fields. Block cyclic redundancy check (CRC) codes represent a popular and powerful class of error detection techniques used almost exclusively in modern data communication systems. The cyclic redundancy check considers a block of data as the coefficients to a polynomial and then divides by a fixed, predetermined polynomial. The coefficient of the result of the division is taken as the redundant data bits, the CRC. On reception, one can recompute the CRC from the payload bits and compare this with the CRC that was received. A mismatch indicates that an error occurred. Though efficient, CRCs can detect errors only after an entire block of data has been received and processed also they add considerable redundancy to the message.

The most widely used form of encryption algorithm is defined by the National Bureau of Standards and is known as the data encryption standard (DES). The DES is a block cipher, splitting the data stream into 64-bit blocks that are enciphered separately. A (probably) unique key of 56 bits is then used to perform a succession of transposition and substitution operations. A 56 bit key has $7.2x\ 10^{16}$ possible combinations. Assuming a powerful computer could attempt 10^8 combinations per second, it would still take over 20 years to break the code. The longer the stream of code bits, lower will be the code rate but more secure would be the data transmitted. If the code is changed once per year, there is little possibility of it being broken, unless the code breaker has additional information. The DES converts 64 bits of plaintext into 64 bits of ciphertext. The receiver uses the same key to decipher the ciphertext into plaintext and hence it is a symmetric key cipher. The patent for this algorithm was closed in 1998 as it was found no more secure. Later triple DES has been used in its place.

The difficulty with this method is that each block is independent. This permits an interceptor in possession of the key to introduce additional blocks without the recipient being aware of this fact. Secondly, the same plaintext will generate the same ciphertext, a fact of great value to someone attempting to break the code. These disadvantages may be mitigated to a large extent by either a combination of data dependent encryption and data independent encryption or to a limited extent by Chaining. Chaining as described in literature is the process of Mixing, like Modulo2 addition, the plaintext of one block with the cipher/plain text of the previous block. In this way it is not possible to introduce blocks in a transparent fashion, and repetitions of the same plaintext at different places generates different ciphertexts. Similar is the outcome when data dependent encryption is combined with data independent encryption. When the data dependent key generation is used, the key will not be same for all text. It would continuously change and hence it is not easy for the interloper to get hold of the key used for all the plaintext even if sent along with data and in the same session. In the absence of corresponding key with the recipient, introducing some blocks of data would not be transparent and immediately detected. In symmetric key encryption method, the data and the key are not generally sent to the recipient in the same session; if sent so, care shall be taken to adopt a protocol, known only to the sender and recipient that help to segregate the message and key in the received data. The third method of course, would be to send the key on a separate secured line or through a third party like Key distribution center.

In Public key encryption algorithm, the receiver makes available the first, public key. This key is used by the sender to encrypt the message. This message is unintelligible to anyone not in possession of the second, private key. In this way the private key need not be transferred. The most famous of such schemes is the Public Key mechanism using the work of Rivest, Shamir and Adleman (RSA). It is based on the use of multiplying extremely large numbers and the major disadvantage of this approach, even with current technology, is computationally very expensive. Another disadvantage with public key cryptography is the length of the cipher text is approximately 3.5 or more number of times that of plain text. It is contemplated that there should be a mechanism to transform and encrypt the message bits into

a form which is secure and at the same time if corrupted by channel noise, practically with little processing, indicate the recipient that the data is corrupted without the need of any additional redundancy. Otherwise when received without errors, should give rise to original data by appropriate inverse transformations.

In other words, error detectability should be inbuilt in to the encryption process and/or in the transformed data. Also, one may establish a relation between the message and its transformed version so that by checking validity of this relation, one can doubly confirm that errors have not been introduced in to the message. This is some what similar to Cyclic Redundancy Check (CRC) but without sending any redundant bits for error detection along with the message. The facility should be as simple as verification of lookup tables for error detection or applying a simple operation which indicates the presence of errors due to channel noise. Also, the processing done at the transmission end may result in making the original message obscure so that the data transmitted on the line even if intercepted should serve little purpose to meddle with the integrity of data in a transparent way. Also, if the entire encryption is not done at one stage and only one part of the encryption key transmitted while the rest is already made available to the recipient in a different session or thorough other means so that integrity of data shall be well preserved. In this paper a system is designed to accomplish majority of these expectations and found that Permutation Invariant Rao Alaka Shift Transform, by virtue of its unique properties, plays a pivotal role in the design.

RAO ALAKA SHIFT TRANSFORM

Kai kuang Ma [1] and others introduced Rapid transform in 1984 as a solution to an image Processing and pattern recognition problem. Later, E.G. Rajan [2] and others extended this in to Generalized Rapid Transform (GRT). Many properties and applications of Generalized Rapid Transform in image processing have been explored and extensively studied at Centre of Research in Electronics & Information Technology (CREIT), Hyderabad and at Pentagram Research Center [3], Hyderabad. Notion of RAS Transform has been put forth by the first author of this paper, Mr. A.V Narasimha Rao and Mrs. Alakanandana, scientist and Managing Director of CREIT**.** This transform with its Permutation Invariance (PI) properties has been used for the sake of binary and gray level image processing and coding of Multimedia digital data [4,5] at CREIT, Hyderabad. So, it is named after the originators as Rao Alaka Shift Transform (RAST). RAST exhibits almost all the properties of Rapid transform and Generalized Rapid Transform besides some more interesting properties which find application in error detection. It is also much simpler to implement and involves practically very less mathematical computation. Algorithm for implementation of RAS Transform is nonlinear and recursive. RAS Transform is somewhat similar to decimation-in-frequency of FFT algorithm and it transforms a number sequence x(n) of any length N, equal to any power of 2, into another sequence X(k) of same length called its RAS Transform and an encryption key E(r) which helps for inverse transformation. The encryption here comes under the category of Transposition ciphers. RAST is value specific transform and it uses relational and shift operations. In RAST, the implementation is data dependent and it can be in two major variations namely, ascending mode of implementation (AMI) and descending mode of implementation (DMI). RAST introduces homomorphism that maps a domain set consisting of a number sequence along with its graphical inverse and their cyclic and dyadic permutations, onto a range set consisting of a unique number sequence, called its RAS Transform and its encryption key. This map is one-to-one and onto correspondence and an inverse map also exists; hence viewed as a transform. This map ensures the invariance property under such permutations. The main advantage of RAST as found elsewhere and put forth herein is that it exhibits permutation invariance, Second-Order Self invariance and is computationally very soft besides being an extremely fast Transform. It is this property of RAST that makes it more attractive than many other transforms, like FFT, GRT etc., used for Digital Image Data Processing & Pattern Recognition, Encryption and Error Detection coding. Examples: If the given input sequences are $x_1(n)$={1 0 1 1 0 0 1 1} and $x_2(n)$= {4 5 7 3 1 8 9 2} then after application of RAST, DMI, they would be of the form $X_1(k)$ ={ 1 1 1 0 1 1 0 0} with $E_1(r)$ ={000001110000} $X_2(k)$ ={ 9 8 4 3 7 5 2 1} with $E_2(r)$ ={011010100001}. After transformation, $X_1(k)$ and $X_2(k)$ are different sequences and hence get encrypted and the encryption keys $E_1(r)$ and $E_2(r)$ which will be helpful for inverse transformation. Even one bit change in the encryption key does not guarantee correct recovery of either $x_1(n)$ or $x_2(n)$. This property can be used to ensure correctness of the received key in data communication. Among the three x (n), X (k) and E(r) it is not possible that only one can change and other two remain unaltered. Whenever x (n) changes there must be a change in either X(k) or in E(r) or in both. Thus, there is a distinct relation between x (n) and the combination of [X (k), E(r)].

PROPERTIES OF RAS TRANSFORM

One dimensional RAST has Permutation invariance properties. This also exhibits other interesting properties like Nonlinearity property and Second-order Self Invariance property. The permutation invariance properties are of three types. They are as follows:

i) Cyclic shift invariance property
ii) Graphical inverse invariance property
iii) Dyadic shift invariance property

Cyclic shift invariance property

Let us consider a sequence x (n) = {7, 5, 8, 3, 2, 6,1,9}. Using this sequence one can generate seven more cyclic shift versions such as

$X_{c1}(n)=\{9,7,5,8,3,2,6,1\}$,
$X_{c2}(n)=\{1,9,7,5,8,3,2,6\}$,
$X_{c3}(n)=\{6,1,9,7,5,8,3,2\}$,
$X_{c4}(n)=\{2,6,1,9,7,5,8,3\}$,
$X_{c5}(n)=\{3,2,6,1,9,7,5,8\}$,
$X_{c6}(n)=\{8,3,2,6,1,9,7,5\}$,
$X_{c7}(n)=\{5,8,3,2,6,1,9,7\}$.

It is obvious that the cyclic shifted version of X_{C8} (n) is X (n) itself. It is found that all these eight sequences have the same RAST [X (k)], as {9 8 7 6 5 2 3 1} but different E(r) which help to uniquely get back the corresponding original sequence.

Graphical inverse invariance property

For x(n)={7,5,8,3,2,6,1,9} the graphic inverse is $x^{-1}(n)$ = {9,1,6,2,3,8,5,7}. Using this sequence one can generate seven more cyclic shifted version such as

$x^{-1}_{c1}(n)=$ {7,9,1,6,2,3,8,5},
$x^{-1}_{c2}(n)=$ {5,7,9,1,6,2,3,8},
$x^{-1}_{c3}(n)=$ {8,5,7,9,1,6,2,3},
$x^{-1}_{c4}(n)=$ {3,8,5,7,9,1,6,2},
$x^{-1}_{c5}(n)=$ {2,3,8,5,7,9,1,6},
$x^{-1}_{c6}(n)=$ {6,2,3,8,5,7,9,1},
$x^{-1}_{c7}(n)=$ {1,6,2,3,8,5,7,9}.

It is obvious that the cyclic shifted version of x^{-1}_{c8} (n) is $x^{-1}(n)$ itself. Again it is found that all the eight sequence have the same X (k), but different E(r).

Dyadic shift invariance proper

The term dyad refers to a group of two, and the dyadic shift to the operation of transposition of two blocks of elements in a sequence. For instance let us take x(n) = {7,5,8,3,2,6,1,9} and replacement of its first half with the second half. The resulting sequence $T^{(2)}d[x(n)]$ = {2,6,1,9,7,5,8,3} is two block dyadic shifted version of x(n). The symbol $T^{(2)}d$ denote the two block dyadic shift operator. In the same manner we obtained $T^{(4)}d[T^{(2)}d[x(n)]]$ = {1,9,2,6,8,3,7,5} and $T^{(8)}d[T^{(4)}d[T^{(2)}d[x(n)]]]$ = {9,1,6,2,3,8,5,7}. One can easily observe that the graphical inverse $x^{-1}(n)$ = {9,1,6,2,3,8,5,7}is the same as $T^{(8)}d[T^{(4)}d[T^{(2)}d[x(n)]]]$ = {9,1,6,2,3,8,5,7}. It is again verified that all these four sequences have the same RAST, but different E(r). There is yet another way of dyadic shifting the input sequence x (n) to $^{(2)}d[T^{(4)}d[T^{(8)}d[x(n)]]]$. Let us take x(n) = {7,5,8,3,2,6,1,9} and obtain the flowing dyadic shifts: $T^{(8)}d[x(n)]$ = {5,7,3,8,6,2,1,9}, $T^{(4)}d[T^{(8)}d[x(n)]]$ = {3,8,5,7,9,1,6,2}, $T^{(2)}d[T^{(4)}d[T^{(8)}d[x(n)]]]$ = {9,1,6,2,3,8,5,7}.

It is obvious that $T^{(8)}d[T^{(4)}d[T^{(2)}d[x(n)]]]$ = $T^{(2)}d[T^{(4)}d[T^{(8)}d[x(n)]]]$. It is again verified that all these four sequences have the same RAST, but different E(r). One can easily verify from the above that other than $T^{(4)}d[T^{(8)}d[x(n)]]$ and $T^{(8)}d[x(n)]$, all other dyadic ally permuted fall under the category of the cyclic permutation class of x(n). This amounts to saying that the cyclic permutation class of x(n) has eight mutually independent sequences, that of $x^{-1}(n)$ has eight mutually independent sequences and the dyadic permutation class of x(n) has two mutually independent sequences. But two sequences are repeated and we have only sixteen independent permutations. Thus, to conclude, all these sixteen sequences could be seen to have the same **RAST**(X (k)). However, each of these sixteen sequences has a distinct encryption code E(r). Also, if the number sequence x (n) is binary in nature, then it is possible for some of them to have less number of mutually independent sequences out of the sixteen. For example, the 8 point binary representation of decimal Number 7 (00000111) and that of 37 (00100101) have only 8 mutually independent cyclic shift permutations out of expected 16. Whereas, 8 point binary representation of decimal numbers 13(00001101) & 71(01000111) have 16 independent cyclic shift permutations. At the same time, there could be one or more binary sequences $x_1(n)$ and $x_2(n)$ and so on which do not fall under the cyclic shift permutations of a given binary sequence x(n) and yet may have the same RAST X(k). Of course, the encryption codes will be different.

Nonlinearity property

The linearity property requires that if x(n) and y(n) are input sequences then the RAS Transform can be said as linear transform if it satisfies the following condition:

RAST[x(n)+y(n)] = RAST[x(n]+RAST[y(n] = X(k)+Y(k). Let us take x(n) = {7,5,8,3,2,6,1,9} and y(n) = {5,8,1,2, 4,5,6,7}. p(n) = x(n)+y(n) = {12,13,9,5,6,11,7,16}. P(k) = {16,12,13,9,11,7,6,5}; X(k) = {9,8,7,6,5,2,3,1}, Y(k) = {8,6,7,5,5,4,2,1}; Z(k) = X(k)+Y(k) = {17,14,14,11,10,6 ,5,2}. Here, P (k) is not equal to Z (k). So RAST does not exhibit linearity. In other words, in general, RAST is Nonlinear. However, second- order RAST exhibits Linearity. Also when x(n) and y(n) are monotonously increasing or decreasing sequences, RAST exhibits Linearity.

Second-order self invariance property

This property says that once RAST of a number sequence is obtained, on repeated application of RAST on the transformed number sequence does not give rise to any change. i.e) Output is equal to the input once RAST is applied. That is if RAST[x(n)] = X(k), then RAST[X(k)] = X(k). Lets take an example x(n) = {1,2,3,4,5,3,4,6}. The RAST of x(n) is X(k) = {6,5,4,3,4,3,2,1}. Taking once again the RAST of {6, 5, 4, 3, 4, 3, 2, 1} it does not change. So RAST exhibits the second-order self invariance property. That is, second order RAST is invariant. This particular property is very useful in error detection applications.

Error detection capabilities of RAS transform

In data communication, let us assume that data is sent in the form of 8-bit octets. There would be $2^8 = 256$ possible binary vectors. If, RAST is applied on these octets, we will get only 20 valid codewords which form a codebook shown in the Table-1.

Sl.No.	Codeword in Decimal	Binary Codeword
1	0	00000000
2	2	00000001
3	3	00000011
4	5	00000101
5	7	00000111
6	15	00001111
7	17	00010001
8	19	00010011
9	21	00010101
10	23	00010111
11	31	00011111
12	51	00110011
13	55	00110111
14	63	00111111
15	85	01010101
16	87	01010111
17	95	01011111
18	119	01110111
19	127	01111111
20	255	11111111

Table.1. Binary RAST CodeBook

Thus, primarily, 8-bit data of 256 combinations is now available in one of the 20 forms only. Any word other than these valid 20 codewords can be readily identified as an outcome of channel noise. Secondly, due to Second-Order Self invariance property, if a received RAS transformed signal is to be verified for its correctness, then RAST of that needs to be computed once again. If its second order RAST does not change, it indicates an error free reception.

Similarly, if the correctness of received encryption key is to be verified, then use that key and recover the input signal by applying inverse transform on corresponding RAS transformed signal. Apply RAST, once again, on the recovered input signal then the output should match exactly with the previous RAS transformed signal. If match occurs, the received encryption key is correctly received otherwise it is likely to be erroneous. Even one bit changes in the encryption key this may not allow the correct recovery. As an example, let x1(n) = {1 0 1 1 0 0 1 1} then X1(k) = { 1 1 1 0 1 1 0 0} with E1(r)={0 0 0 0 0 1 1 1 0 0 0 0}. If key is changed by one bit say E2(r) = {0 0 0 0 0 1 1 1 0 1 0 0}. Using E2, if Inverse RAST is applied on X1(k), we recover x2(n) = {1 1 0 1 0 0 1 1} which is different from x(n) and whose RAST is X2(k) = {1 1 1 1 1 0 0 0}. We can very clearly observe that X2(k) is not same as X1(k). Therefore, one method of error detection in the received encryption key is comparing the received RAST signal with the RAST of recovered signal, using the available key. If both are same then the received encryption key is error free. These error detection capabilities of RAST are exploited in the system proposed for integration of Encryption and error detection in this paper.

SYSTEM DESIGN AND METHODS

This system for integration of Encryption with Error detection mainly consists of the following stages or steps. Figure-1 gives the functional block arrangement for the system that is proposed in this paper. The various blocks are a) Preprocessing of message using Data Block Chaining. b) Data independent Encryption c) Data dependent Encryption d) Block chaining of keys e) appropriate amalgamation of encrypted data and keys f) At the receiving end, segregation of data and keys g) Verification of Data and Keys for error free reception h) Data dependent Decryption i) data independent decryption and j) Dechaining and verification of data integrity.

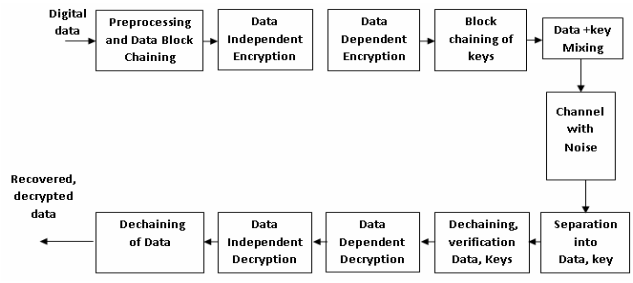

Figure.1. Functional Block Diagram showing a system integrating Encryption and Error detection

In the preprocessing stage the available digital data is first partitioned in to 8-bit words and blocks each consisting of 16 bytes. Also, Data Block Chaining (DBC) is accomplished [6].

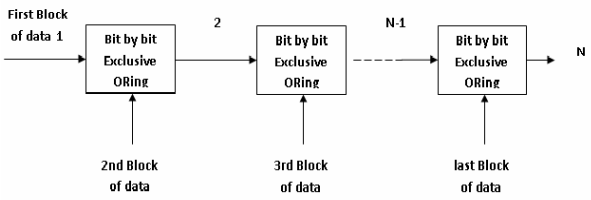

Fig.2 Data Block chaining

That is Modulo2 summation of one block of data with the other block is obtained and the result with the 3rd block and so on. This process is shown in Figure-2. The blocks of data subjected to further processing are N blocks from 1, 2. . . . . N-1, N. By this data blocks can not be introduced in a transparent way. The last block would simply be the checksum of all blocks of data after preprocessing. Checksum [7] as defined is a function of the remainder of the message. The function need not always be modulo2 sum between blocks. At the receiving end, same calculation is performed and the result is compared with the incoming code. A discrepancy results if there has been some error in transmission.

Data Independent Encryption (DIE), as proposed in this system, consists of a specific number of different operations performed on each byte/block of data. Each of these operations is essentially a transposition or XOR operation or a combination of these in a particular manner. The sequence of these operations and how many times each of these is performed is indicated by a DI encryption key. The first part of key holds the information about the sequence of operations and the second holds the key for number of times the operations are applied. As an example, if four operations are selected, there would be 16 possible sequences in which 4 operations can be performed. Assuming that two of the operations can be repeated from 0-3 times and others are restricted to only 0-1 time there would be 64 combinations. Thus a two byte key may be required to represent one of the 1024 possible Data Independent Encryption schemes.

Data Dependent Encryption (DDE) is the crucial part of this Encryption-Error detection combo system. Permutation Invariant RAS Transform is made use for dual purpose coding at this stage. RAS transform will be applied on each byte of data and again on each block of data consisting of decimal numbers representing the codeword. It is mentioned earlier that when RAS Transform is applied on any of the 256 possible data bytes, it would be transformed in to one of the 20 valid codeword of the binary codebook shown in Table-1. Each one of these codeword does not vary if RAS Transform is applied second time. Then each of the blocks consisting of binary code words, represented in decimal numbers, is further transformed using RAS Transform. This stage generates 12x16 =192+32 = 224 bits of Data Dependent Encryption key along with encrypted data of 16 octets. There shall be two implementations to obtain RAS Transform. One of them not only transforms the data but also generates a DD Encryption key. This implementation will be used at the transmission end. The second implementation accomplishes mere transformation and does not generate any encryption key. This version will be used at the receiving end to verify the correctness of received transformed data. The decryption algorithm is to be used only at the receiving end and this requires the DD decryption key received along with data.

Key Block Chaining (KBC) means generating checksum for key bits by XORing the key bits after making them in to suitable blocks. Taking 32/16 bits at a time in to a block, the total data dependent encryption key is Block Chained and the last block of 32/16 bits is the checksum of the rest of the blocks of DDE key. The process is similar to what is shown in Figure-2 for data block chaining except that the block size is of 4/2 octets instead of 16 octets. One key chain of 8/16 blocks belongs to one block of data of 16 octets. The choice of how many bits of key to be made into one block may be varied from message to message or as frequently as needed.

Next stage is appropriate amalgamation of encrypted data and keys. Now, it is clear that the encrypted data is available in blocks of 16 bytes. Total Data Dependent Encryption (DDE) key is 224 bits/block and say16 bits of Data Independent Encryption (DIE) key. In the simple system proposed here the DIE key need to be sent only once and hence will be sent at the end of the session as a tail piece. It is proposed that the one block of two-level encrypted data may be sent followed by one block chained Encryption key. Thus, as an example, for every 128 bits of data we get 128+224 = 352 bits of Encrypted (data + key). That is 2.75 times the size of original data. Here, if data and key are interleaved it would be very difficult to identify which part is pertaining to data and which part of the key, unless algorithm is known completely. It is these 352 bits/block and a tail piece of two octets of DIE key, that will be sent on the line.

When data is sent on the noisy channel, it is likely that some times there are some errors. Hence at the receiving end, verification of data for any errors is essential. The first block in the receive chain (receive leg) of Figure-1, is the separation of encrypted data and corresponding keys. Based on the way it is mixed at the transmission side, appropriate filters are to be used to pick suitable number of bits into data part and keypart. At next stage data part and keypart is verified for correctness.

In the Keypart, the last block of 32/16 bits serves as the checksum and by appropriate de-chain mechanism, 32/16bits of block level encryption key and the remaining 192 bits is further segregated in to sixteen 12-bit, byte level encryption key corresponding to each octet of data. In the data part, RAS Transform is applied to the block of 16 bytes. Neither is the need of a key for this operation nor is any key generated. As the data is already the outcome of RAST, it should not change if received error free. At this stage, if any of the bytes has changed due to errors this will be readily known. Thus block level correctness of data part gets verified. The byte level correctness shall be verified after block level decryption.

In the next stage, the 32/16 bits of block level encryption key is used to decrypt the block of data. What is obtained is a block consisting of valid binary code

words, represented in decimal numbers. As there are only 20 valid codeword, any error in transmission can immediately be detected either by inspection from a lookup table for codeword, called codebook, or by taking RAST of these received 8-bit binary words. No change in their value after RAST indicates error free transmission. Successive verification at block level and byte level completely detects all possible errors. The very important point to be noted here is that this error detection is possible without sending any additional bits exclusively for error detection. This could be realized due to the permutation invariant and 2nd order self invariant properties of RAS Transform. Thus, the use of RAS Transform is enabling us to integrate error detection with encryption.

In the next stage, the binary codewords are handled, byte by byte and data independent decryption is performed with the help of DIE key. The information about the sequence of operations and the number of times the operations are applied is noted from this DIE key. The sequence of corresponding inverse operations, of course, will be reversed at the receiving end but the number of times each of the corresponding operations need to be performed can be arrived at using an S-box. A sample S-Box looks like what is shown in Figures 3 and 4. The values shown are not indicative of the system proposed and used here.

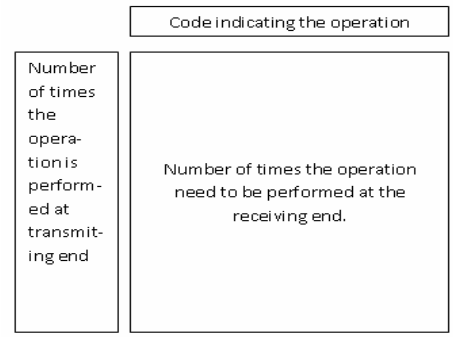

Figure.3. S- Box Lay out

		Code indicating the operation			
		00	01	10	11
No of times applied at sending end	1	3	5	1	2
	2	5	1	3	2
	3	1	2	1	4
	4	.	.	.	.
	5	.	.	.	.
	6	.	.	.	.
	7	.	.	.	.

Figure.4: Sample S-Box

An S-box (substitution box) will be used, at the receiving end, to generate the corresponding decryption key for a given DIE key, indicating the numbers of times each of these operations need to be performed. This S-box is generated based on which of the operations and how many times when repeated in a given sequence, shall give rise to the original data

The last but equally important stage at the receiver end is de-chaining and verification of data integrity. The received data bytes are de-chained in to blocks of 16 bytes each. Using the last block of data, which is the checksum of all other blocks of data the integrity of data is verified.

RESULTS AND DISCUSSIONS

The system proposed is implemented using MATLAB-7 version. Some of the results obtained for coding a sample text data of about 189 characters size, shown in Figure-5, is presented here and discussed. For the stages on the receive leg, the output is not in real time and hence the display for possible error situation, by explicitly introducing errors/no errors is shown.

Asian Research Publishing Network (ARPN) is publishing an online international research "Journal of Engineering and Applied Sciences". This encouraged many Researchers and students of Ph.D.

Figure-5. Input Plaintext of 189 characters.

The plaintext fed to the system is made into blocks of 16 characters, and block chained as shown below before applying Data Independent Encryption. The corresponding outputs are shown in Figures 6 and 7.

BlockedDataText
=Asian Research P
ublishing Networ
k (ARPN) is publ
ishing an online
international r
esearch "Journal
of Engineering
and Applied Scie
nces". This enco
uraged many Rese
archers and stud
ents of Ph.D.

Figure-6. Blocked text.

BlockChainedDataText =
Auki e anuae
2□K□I□O□ □□
[{cr's)khfp□
:□"□S□ K□□□
Tapu6`L 9d},
t □D□"z□ □C
&`n2*kE 7|%
C GSKK,fcM\□
0ig=?iB□ ,=U
UI □V#'jbBS=
4□}r9LB □;7□
Fb]□W90.1□□W

Figure-7. Blocked chained data text.

The Block Chained data is subjected to Data Independent Encryption to generate DI Cipher Text which is shown in Figure-8.

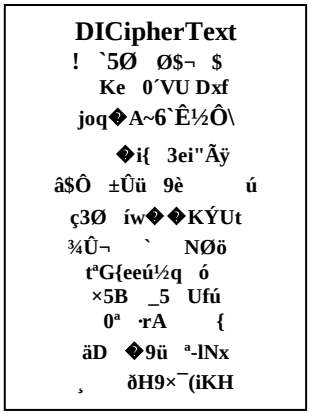

Fig.8: DI Cipher Text Data

The out put of the transmit leg, the combination of completely encrypted (both Data Independent and Data Dependent) text along with Block level and byte level encryption key combined, is shown in Figure-9.

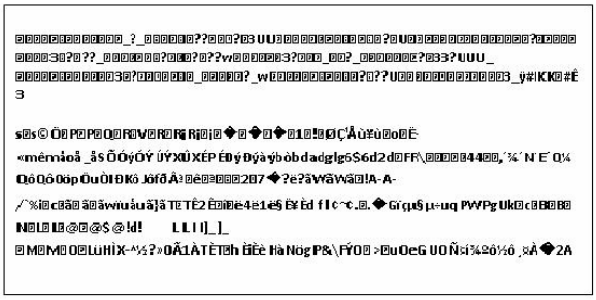

Figure-9. Encryption output Cipher Text (Data + Key).

Some of the characters are in non printable form and some characters that represent form-feed and linefeed etc., have not been preserved to save some space in these figures. The input Plaintext and the combined output of encrypted data and keys has been appearing in the form of one big line as a single string of characters, again to manage the space, they have been made into different lines as shown in Figure-9. The system handles the digital data in the form of either binary/decimal numbers but for convenience of observation the corresponding characters have been shown and not the binary/decimal vectors.

The receive leg comprising of decryption has recovered exactly the sent data and it took lot of effort in suitably mixing data and key in the transmit leg and to separate in the receive leg of the system. The recovered text at various stages of decryption was exact in the test system and matched with what is shown in the Figures 6, 7, and 8. Also, the text data is recovered without any errors and was perfectly matching to the input text shown in Figure-5. These have not been shown again. Some intentional errors were introduced in the received (encrypted) data at both byte level and block level which were successfully detected. Three cases of error detection have been shown in Figure-10.

Case-I: >> Verify16blocklevel(r, k)
No error, all valid Code Symbols
No error in this block

Case-II >> Verify16blocklevel(r, k)
Error!! 9 is invalid Code Symbol

Case-III >> Verify16blocklevel(r, k)

No error, all valid Code Symbols! But, there is an error in this block of data. Please retransmit the data block.

Figure-10. Error Detection Examples.

Case-I and Case-II are for detecting any decimal symbols representing any invalid codeword has appeared due to channel noise. In the first case all codeword are found valid where as Case-II, one of the symbols is found erroneous. Case-III, checks if due to channel noise any of the codewords got changed to some other valid but incorrect codeword. Thus the block level integrity of data and validity of its corresponding key are simultaneously ensured at this stage. Encryption and decryption parts of the system have been thoroughly tested for proper functioning with a variety of text data differing in length and content. The system is found working well without exception.

ACKNOWLEDGEMENTS

The authors hereby acknowledge the co-operation extended by the administration and faculty of department of ECE, CBIT, CREIT and PRC, Hyderabad. The discussions with the Heads of these three organizations and scientists working there have helped significantly in the design and implementation of this system for integrating Encryption and Error Detection using RAST. Special reference would not be out of place to Mr. G. Pradeep Kumar, M.E. of CBIT and Mrs. Alakanandana, M.Sc., Mr. M. SrinuNaik, ME., of CREIT, Hyderabad.

REFERENCES

[1] Kai-Kuang Ma, Paul P. Wang and Jack Rebman.1984. Automatic Recognition of Low Resolution Tactile Sensing Data using Rapid Transformation. NATO ASI Series. Vol. F11, Springer-Verlag, Berlin, Heidelberg.

[2] Rajan E.G. 1997. On the Notion of Generalized Rapid Transform. World Multi Conference on Systemic, Cybernetics and Informatics. Caracas, Venezuela. July.

[3] Rajan E.G., Narasimha Rao .A.V. *et al*. 1998. Use of Generalised Rapid Transform in the processing of Medical Images. Proceedings of National Conference

on Biomedical Engineering, NCBME-98, Manipal Institute of Technology, Maniple, 9-11 April.

[4] Narasimha Rao A.V., K. Srinivasa Rao and A. Alakanandana. 2003. Permutation Invariant Coding of Digital Data. Proceedings of IETE Golden Jubilee Symposium on Convergence of Technologies, IETE Hyderabad Centre, Hyderabad. 11-13 July.

[5] Narasimha Rao, A.V. Srinivasa Rao K. *et al.* 2007. The Role of Permutation Invariant RAS Transform in the Machine Vision Based Object Identification. Proceedings of 4th National Conference on Applications of Emerging Technologies, Adhiyamman College of Engineering, Hosur. 4-5 April.

[6] Ajay Kakkar, M. L. Singh and P.K. Bansal Data. 2008. Security by the use of Encryption: Round Functions. Proceedings of National conference on Recent Trends in Electronics and Communication. 10-11 April.

[7] William Stallings. 2003. Data and Computer Communications. 6th Ed., Pearson Education (Singapore) Pvt. Ltd.