A Crypto-System with Embedded Error Control for Secure and Reliable Communication

Ranya Alawadhi *ralawadhi@smu.edu*
HACNet Labs, Bobby B. Lyle School of Engineering
Southern Methodist University
Dallas, TX, 75275, US

Suku Nair *nair@smu.edu*
HACNet Labs, Bobby B. Lyle School of Engineering
Southern Methodist University
Dallas, TX, 75275, US

Abstract

In this paper we propose a novel Crypto-System with Embedded Error Control (CSEEC). The system supports data security and reliability using forward error correction codes (FEC). Security is provided through the use of a new symmetric encryption algorithm, while reliability is provided through the support of FEC codes. The system also supports joint security and reliability in which encryption and encoding are performed in a single step. The system aims at speeding up the encryption and encoding operations and reduces the hardware dedicated to each of these operations.In addition, the proposed system allows users to achieve secure and reliable communication in which they can alternate between a priority onsecurity and reliabilityand scale their choice to the desired level in order to attain communication quality and fulfill application needs. The system targets resource constrained nodes such as remote sensor nodes operating in noisy environments.

Keywords: Joint Encryption and Error Correction, Data Security, Data Reliability, Erasure Coding, Forward Error Correction.

1. INTRODUCTION

Data security and reliability are integral aspects of modern communication systems. They are achieved through encryption and forward error correction (FEC). Conventional encryption schemes provide a high level of protection at the expense of processing time and energy. These methods force devices with constrained resources to settle for either low or no strength schemes.

Security and reliability operations have always been dealt with separately due to their contradicting objectives. There have been some efforts to combine them by joining encryption and coding into a single step. The aim was to have a faster more efficient communication in terms of time, energy, and area[1]. However, many efforts [2]–[5]did not get a lot of attention due to their large key size, high overhead or inefficient correction capability, while others[2], [4], [6]–[8]did not achieve enough strength to compete with conventional encryption schemes.

With consideration to the work that has been done previously in the area of joint security and reliability, we propose a novel Crypto-System with Embedded Error Control (CSEEC) for secure and reliable communication. Thissystem supports data security and reliability. In addition to the support of these functions separately, the systemalso supports the joint functionality when encryption and encoding are combined in a single process. CSEEC provides all these functionsusing forward error correction (FEC) codes. FEC codes are commonly used to achieve data reliability. However in this system, they are combined with specially designed operations thatallow them achieve data security as well.

To achieve data security we propose a new symmetric encryption scheme with a 128-bit key. The idea is based on the ability of erasure codes to recover from errors only when the exact locations of these errors are determined. Thus to encrypt a block of data, it is first encoded using the FEC code, then an amount equal to the amount of added redundancy is intentionally deleted. The deletion process is controlled by the encryption key. Thus, only those who possess the key are able to recover the original data. However, the idea of deletion by itself is not enough to achieve confusion and diffusion, the two properties that characterize a secure system[9]. Hence the deletion operation is combined with other operations,as well as permutation and randomization. Permutation rearranges the bits within a block while mixing combines the processed data with a random sequence. The proposed encryption scheme is not a strict block encryption sinceit does not have a traditional S-box. It is not a strict stream encryption either sinceit processes the stream of bits in fixed-sized blocks.

The support of data reliability or error control capability intuitively comes from the support of forward error correction code. This capability can be used to detect errors, correct errors, or correct erasures. The exact function will be determined by the application, the channel, and the amount of redundancy added. To achieve joint security and reliability, we extend the encryption scheme by making the amount of redundancy deleted less than the amount of redundancy added. Thus, the extra amount can be used to control the errors introduced by the channel. Due to our ability to range the amount of data deleted, the security and reliability levels are easily scaled.

CSEEC is different from previous proposals [3], [4], [11]. It does not incur any communication overhead when FEC codes are used as a mean of security. This is due to the notion of erasures, as opposite to errors. Erasure allows CSEEC to maintain the ciphertext size equal to the plaintext size. It also uses dynamic random permutations in which each processed block is permuted differently. In addition, it has a reasonable key size(128-bit)from whichall components are initialized or derived.

The rest of the paper is organized as follows. In Section 2 we discuss related work. In Section3 we describe the proposed encryption scheme. In Section 4 we present the joint reliability and security scheme. In Section 5 we provide an analysis of results showing the superiority of our method. And finally, in Section 6, we conclude the paper and describe directions for future work.

2. RELATED WORK

The first effort in the field of joint encryption and error correction was contributed by McElience[2]. McEliece introduced a public key cryptosystem based on algebraic coding theory. His idea was based on the fact that the decoding problem for an arbitrary linear code is NP-complete. The system was based on a class of error correcting code known as the Goppa code. The McEliece system is inefficient in terms of error correction capability because it requires very large public keys and large block sizes to correct the large number of errors,which result in high computational overhead. Also, the original system has been shown to be vulnerable to chosen-ciphertext attacks [12]. More work, with mixed results,have been done to extend theMcEliece system. However, the large key size remains an unsolved problem for McEliece-based systems.

A general encryption scheme based on MDS codes was proposed by Xu[13]. Xu proposed combining cryptographically strong random key stream generators with erasure correction codes. In general, the scheme makes use of any (n,k)MDS code, where $n \gg k$. A ciphertext corresponding to k symbols plaintext is chosen to be k symbols selected from the n symbols codeword generated from encoding the plaintext. The symbols are selected based on a sequence generated from the random number generator each time a plaintext block is to be encrypted. Clearly, as in stream ciphers, the security of the scheme depends on the strength of the random number generator. Further analysis is required to assess the system.

Another scheme that makes use of erasure codes was proposed in [4], a secure erasure coding scheme(SEC) for peer-to-peer storage systems. The scheme aims at ensuring the confidentiality of long term archive data through the use of the proposed encryption scheme along with fragment naming and placement procedures. SEC uses a customized version of the Reed-Solomon erasure code in whicha secret generator matrix is constructed from a user specified key by customizing the Cauchy matrix. The encryption scheme is susceptible to known plaintext attack when used in stand-alone mode.

In [3], a symmetric encryption scheme based on erasure correction codes is presented. The scheme starts by compressing and permuting the plaintext. Then, the result was encoded using an erasure correction code. The encoding phase was followed by intentional data loss through which a number of columns were removed from the block. Finally, another transposition was applied. This scheme, however, suffers from a couple of problems. The first one is the use of compression. Although compression removes redundancy, it does not add randomness [14]. In addition, compression may increase the data size when encrypting previouslycompressed data. The second issue is the key that includes all encryption parameters. This results in a large and variable key size.

An error correction cipher calleda High Diffusion (HD) cipher is presented in[5]. They used the Advanced Encryption Standard (AES) structure and replaced its diffusion layer with an error correcting code. They proposed usingHD codes that possess maximum diffusion and achieve optimal error correction. The cipher is composed of multiple iterations of the round function and key mixing operations. Though the system provides both data security and reliability, it is highly complex.

A more recent work is presented in [11], Error Correction-Based Cipher (ECBC). It is a scheme that combines error correction and security. ECBC is hardware based and designed for wireless networks. It is based on the McEliece scheme and employs a block chaining technique. The ciphertext is generated by adding a randomly generated error vector to a permuted block where the permuted block is the result of multiplying a nonlinearly transformed encoded plaintext viaa permutation matrix. In [15], it was found that ECBC is vulnerable to chosen plaintext attacks.

3. THE ENCRYPTION SCHEME

We propose symmetric encryption with a 128-bit key. The idea is to make use of the fact that a plaintext block can be recovered from a subset of the encoded block provided that enough information is available. Based on that, we intentionally introduce erasures by deleting part of the encoded block in the encryption process and later use the decoding algorithm to recoverthe deleted values from those erasures. The success of the decryption process depends on the knowledge of how erasures are introduced in the first place. Thus to prevent anyone from decrypting the data, erasures are introduced in a way known only to communicating parties.

3.1. System Parameters

Before processing the data, the sender and receiver must agree on a number of parameters. The parameters are: the error correction code $\mathbb{C}$, block size determined by the number of rows r and columns c, number of parity columns P, number of parity columns used toward reliability R, and a pseudo random number generator PRNG.

$\mathbb{C}$can be any correction code with erasure correction capability. The erasure correction codes can correct any number of erasures up to the number of added redundancy. Usually, the error correction code supports specific block sizes. Therefore, the sender and receiver must choose a suitable block size and, accordingly, determine $r$and c. For example, in Linear codes for Erasure error Correction (LEC) [16],c has to be prime and r is set at$c - 1$. The selected size will depend on the application and the supported hardware capabilities and it should be chosen for fast decoding and high throughput.

Prepresents the number of extra columns that will be generated by the error correction code. Out of these Pcolumns, R columns will be used toward reliability. Thus, $P - R$ columns will be intentionally discarded and will not be included in the output. When the system is used for encryption only, R is set to zero. This means that an amount equal to the amount of added redundancywill be deletedfrom the encoded block. However, what is deleted does not necessary have to be part of the original block; it could be part of the added redundancy. This is determined by the key.

PRNG is used to generate a random sequence that is added to the processed block. This component is used to achieve the desired confusion. The selected generator will be initialized with a secret key and an initialization vector (IV). Overall, the generator should be selected with speed, efficiency, and security in mind.

3.2. Initialization

The initialization process is the same for encryption and decryption. In this process, the system state is determined and the necessary components are initialized with the shared secret key. The two main components are PRNG and a number of random permutation arrays.

The PRNGisused to generate random sequences $rSeq_i$ that will be mixed with the encoded block. The initialization process is dependent on the generator in use. Generally, this process ismade sensitive to key changes such that small changes in the key will be reflected in the generator output. It is also necessary to use a different initialization vector (IV) every time the PRNG is initialized. This is a measure taken to make sure that no relationships can be deduced from ciphertexts encrypted with same key at different sessions.

The random permutation arrays have two purposes. They are used to shuffle or arrange bits within a block and identify the columns that will be deleted from the encoded block. For these purposes three random permutations are required:

1. $P1$: is a bit permutation that will enable permuting bits within a block of size$r \times c$.
2. $P2$: is another bit permutation that will enable permuting bits within a block of size$r \times (c + R)$.
3. $P3$: is a column permutation that will specify the order of$c + P$ columns within a block.

We propose using a modified version ofthe RC4 key scheduling algorithm to derive these permutations. We are not restricted to this specific algorithm; any other algorithm maybe used as

```
1  procedure permutations-initialization(key,KeyLen,c,r,P,R)
2     forifrom 0 to (r*c)-1
3        P1[i] ←i
4     enfor
5     forifrom 0 to (r*(c+R))-1
6        P2[i] ←i
7     enfor
8     forifrom 0 to c+P-1
9        P3 [i] ←i
10    enfor
11    j ← 0
12    forifrom 0 to (r*(c+R))-1
13      j ←(j+Key[i mod KeyLen]+P1[i mod(r*c)])mod (r*c)
14      SWAP(P1[imod (r*c)], P1[j])
15      j ←(j+Key[i mod KeyLen]+P2[i])
16      SWAP(P2[i] , P2[j])
17      j ←(j+Key[i mod KeyLen]+P3[i mod c+P])modc+P
18      SWAP(P3[imodc+P], P3 [j])
19    enfor
20    return (P1,P2,P3)
```

FIGURE 1: Permutation Generation Pseudo Code.

long as it can achieve the desired results and it is sensitive to small changes in the key. In other words, two keys that differ in a very small number of bits will produce two completely different sets of random permutation arrays. Furthermore, the algorithm must not be known to have any structural weaknesses that could be used later to attack the system.

To generate these arrays, we start by filling each one of them with the identity permutation and then permute them according to a key in the same way RC4 uses its key to setup its internal state. Specifically, weuse two indices and loop over all positionsand, at each iteration, swap the contents pointed by these two indices. One of these indices is incremented as a counter, while the other is incrementedrandomly using the key. To generate multiple permutation arrays, we chain them together instead of generating each one independently from the other as describe in Figure 1.

3.3. Encryption

As mentioned earlier the main idea of encryption is the partial deletion of the encoded block. However, the deletion by itself is not enough to achieve the two properties identified by Shannon [9]: diffusion and confusion. For that purpose we use dynamic permutations combined with randomization. The encryption process is described in Figure 2.

To encrypt, start by permuting the bits of the input block B_i using the permutation array $P1$. Figure 3 illustrates how a permutation array is applied to a small input block of size 3×3.

$$S_i = B_i P1_i$$

Then, encode the permuted block S_i using the agreed on $\mathbb{C}$. The encoding process will generate P parity columns, thus extending the size of the output block $T_i$to$r \times (c + P)$:

$$T_i = \mathbb{C}(S_i)$$

Next, the encoded block T_i is randomized by XORing it with the random sequence $rSeq_i$ from PRNG. The block in this step is processed row by row to make the maximum continuous sequence available from the PRNG after deletion is no more than $c - P$ bits.

$$U_i = T_i \oplus rSeq_i$$

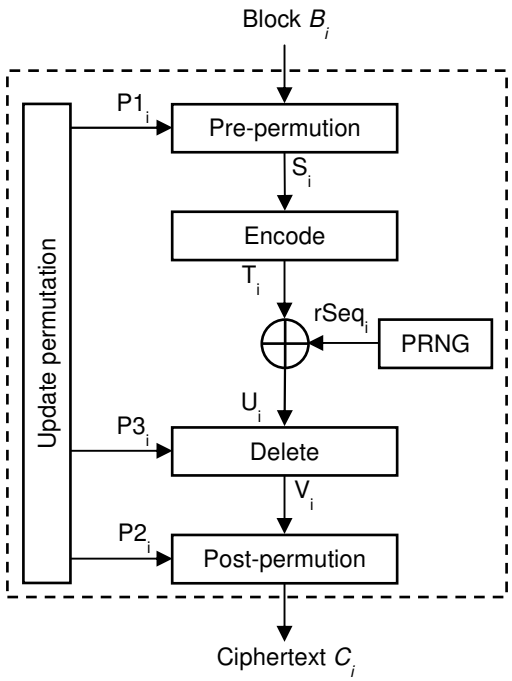


FIGURE 2: Encryption.

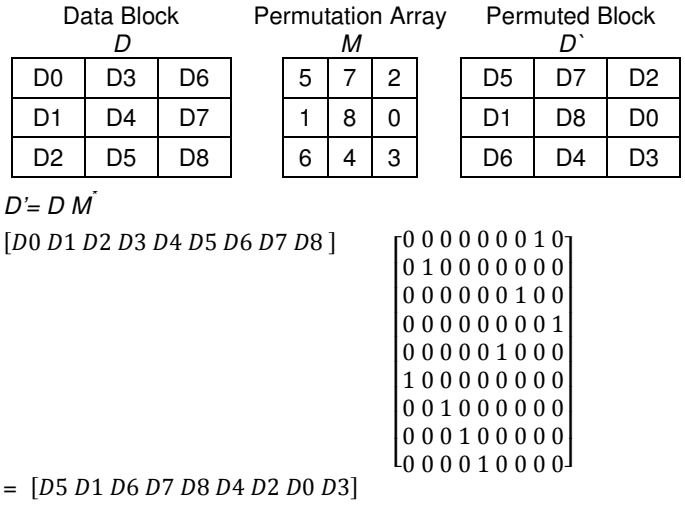


FIGURE 3: Permutation Example.

This is followed by the delete step where$P$ out of the $c + P$ columns are selected and removed from the randomized block. The selection is determined using $P3$. The first P entries of $P3$specify the ids of the columns that will be removed. Figure 4 illustrates this step. It shows the deletion of two columns from a 3×4data block using a permutation array.

$$V_i = U_i P3_i$$

The objective of performing the XOR operation after encoding and before deletion is to increase the complexity of the PRNG cryptanalysis. In this case, part of the generator output will be deleted and there is no way to recover what is deleted especially when is not protected by the correction code. Thus, an attacker will not have a continuous output sequence from the PRNG. Therefore, he will not have reliable knowledge as a basis for his cryptanalysis.

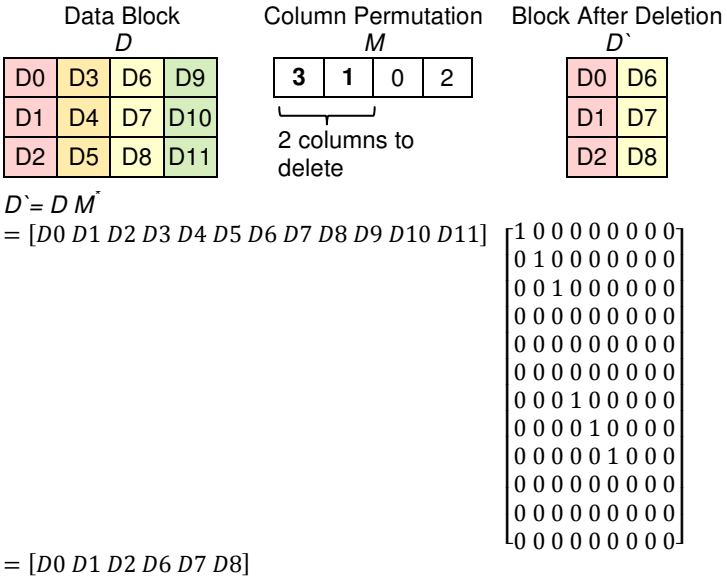


FIGURE 4: Delete Step Illustration.

Finally, generate the ciphertextC_i by performing another bit permutation:

$$C_i = V_i P2_i$$

At the end of processing each block, a new set of permutation is generated. This new set allows different columns to be deleted as well as different bit permutations to be performed every time a block is encrypted. The new set is derived from the existing set using the encryption key. For that purpose, a slightly modified version of the permutation generation algorithm is used. In this version, start with the existing permutations and loop over the key bytes rather than the array entries. This means that, for those arrays with a number of entries more than the number of bytes in the key, common or fixed entries may be found in the initial set and the derived one. However, that should not affect the security of the system as a whole since the key and the initial and derived permutations are all kept secret. The update permutation algorithm is described in Figure 5.

```
1  procedure Update-Permutations(key,KeyLen,c,r,P,R,P1,P2,P3)
2     j=0
3     forkfrom 0 to keyLen-1
4        j ← (j+Key[k]+P1[k])modc+P
5        SWAP(P1[k], P1[j])
6        j ← (j+Key[k]+P2[k]) mod r*(c+R)
7        SWAP(P2[k] , P2[j])
8        j ← (j+Key[k]+P3[kmod r])modc+P
9        SWAP(P3[kmodc+P], P3 [j])
10    endfor
11    return(P1,P2,P3)
```

FIGURE 5: Update Permutations Pseudo Code.

3.4. Decryption

The success of the decryption is based on theability to identify the exact positions of the deleted data for the decoding algorithm. It is not necessary to decrypt previous blocks successfully. However, it is necessary to be synchronized with the encryption process to maintain the right system state in terms of the permutation arrays and the PRNG state. The decryption process is described in Figure 6.

To decrypt, start by reversing the post-permutation:

$$V_i = C_i P2_i^T$$

Then, identify deleted columns using $P3$ and rearrange them into their proper order. This step will expand the block to include the deleted columns. This is an important step for successful decryption for a number of reasons:(1) to ensure that the XORing with the random sequence is performed correctly and (2)because the order of symbols is as important as the value of symbols to decoding.

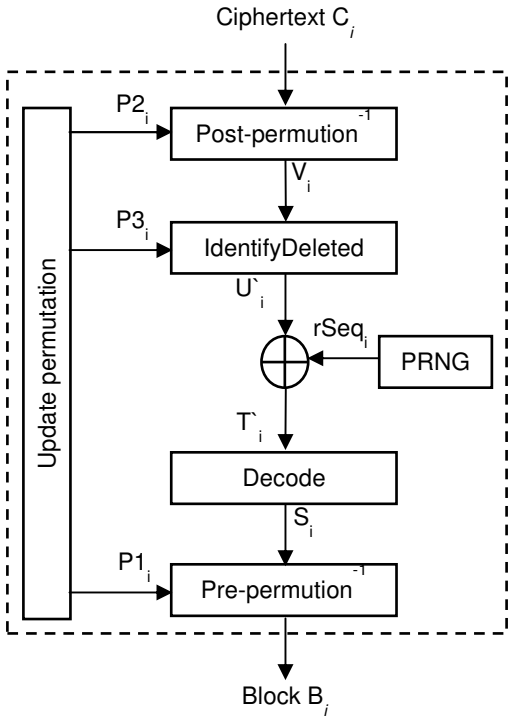
$$U_i' = V_i P3_i^T$$

Once the columns are put in order, the data can be extracted by adding $rSeq_i$:

$$T_i' = U_i' \oplus rSeq_i$$

Next, reconstruct the P deleted columns using the decoding algorithm of $\mathbb{C}$:

$$S_i = \mathbb{C}^{-1}(T_i')$$



Finally, recover the plaintextB_i by reversing the bit permutation:

$$B_i = S_i P1_i^T$$

FIGURE 6: Decryption.

As in encryption, after processing a block, a new set of permutation arrays are generated. They are derived from the existing one using the algorithm described in Figure 3. The permutation update function in decryption must be identical to the one in theencryption ifupcoming blocksare to be decrypted successfully.

4. JOINT SECURITY AND RELIABILITY

Joint securityand reliability allows us to perform encryption and error control simultaneously in a single process. This process is an extension ofencryption where the number of deleted columns is less than the number of added parity. In this scenario, when determining P one needs to consider how many columns will be deleted and how many will be added for error control purposes.

The number of deleted columns is set with the amount of securitythat is required in mind. The more columns deleted the more complex the attacks on the system get, and the more columns need to be reconstructed. Generally, the decoding process is more expensive than the encoding one,which means the more columns recovered the more delay one may expect. Overall, the number of columns deleted should not affect the system performanceand, at the same time, should achieve the desired security level.

As to those columns added for error control, one needs to determine what type of error control is required: error detection vs. error correction. This will be determined by application needs and thecommunication channel. Based on the channel, the type and number of errors expected may beidentified. Consequently, R is determined suitably. In case R is set to be equal to P, then no columns will be removed and the securityof the data will be maintained only by the randomization and permutations.

Once the system parameters are determined, the initialization process can be followed as in Section 3.2. In this process the key is used to set up the system by initializing the PRNG and deriving the random permutations. Upon successful initialization the blocks can be processed, essentially, as in the encryption case(Figure 2). First, the bits of B_i are permuted within the whole block. Then, the permuted block is encoded by $\mathbb{C}$ where P extra columns are generated. Next, the encoded block is XORedwith a random sequence generated from the PRNG. This is followed by the deletion step where the encryption and the joint functions differ. In this case, $P - R$ columns are deleted using $P3$ where the first $P - R$entries determine the ids of those columns. Finally, coded ciphertext is generated by permuting the bits within the whole block where the masked parity and data bits are mixed together. The encoded ciphertext C_i is expressed as follows:

$$C_i = \left(\mathbb{C}\left(B_i P1_i\right) \oplus rSeq_i\right) P3_i \; P2_i$$

Error control can only be performed by those who possess the key. To be able to at least detect channel errors, the permutation and XOR steps must be reversed first before applying the decoding process. Those two operations—although performed on data as well as parity bits—do not propagate errors when they occur.

To decrypt an encoded ciphertext C_i, proceed as in Figure 6. First, the bit permutation is reversed. Then, the deleted columns are identified and the remaining ones are rearranged in their proper order with the use of $P3$. Next, the random sequence is extracted by XORing it back with the block. The decoding algorithm is then applied to recover the deleted columns and detect or correct communication errors. Communication errors can only be handled if they are within the capability of the correction code. Finally, the bit permutation is reversed to obtain the plaintext block B_i back.

$$B_i = \; \mathbb{C}^{-1}(C_i P2_i^T P3_i^T \; \oplus rSeq_i) \; P1_i^T$$

4.1. Data Reliability

Data reliability, where no security measures are required, is achieved through the use of forward correction codes. These codes are used to detect errors, correct errors, or correct erasures. The exact functionality is determined by the application and communication channel. These two factors also determine the number of parity added to the data. In this case, R is set to be equal to $P$and only the correction code is used without the other operations.

For reliable communication, data is encoded using the agreed on error correction code. The encoding process generates P parity columns. Then, the data along with the parity are sent to the destination. At the other end, the decoding algorithm is executed to detect errors and if possible correct them, or, in case of erasures, if they are identified, they are corrected in orderto recover the original data.

5. ANALYSIS

This section provides an assessment of the proposed scheme in terms of security, randomness, and highlights the system performance. In terms of the correction capability, the analysis is the same as the analysis applied to the error correction code as no modification is applied on the code itself. However, when considering the correction capabilities of the code with respect to communication errors, one needs to consider the amount of redundancy assigned to the reliability rather than the amount of redundancy generated from the code.

5.1. Security

The security of the system depends on how hard it is to find the encryption key. CSEEC key is used in two ways: to initialize the PRNG and to derive the random permutations. Extracting the key from the random permutation is a very challenging task for a number of reasons. First, these permutations are kept secret. Second, the permutations are updated every time a block is

processed, which means there will not be enough blocks that use the same permutations for analysis. Third, the randomization and deletion add an extra burden on the process of extracting the key. Thus, extracting the key from the PRNG output sequence isa more applicable approach in determiningthe key.

For known PRNGs, there are a set of attacks that can be applied to determine the key used to initialize the PRNG state. However, any attack can only be applied on reliable knowledge of the output sequence, which is not the case here. All the PRNG sequences used to encrypt any ciphertext are randomly permuted and partially deleted. Thus, before applying any attack, the effect of the random permutation and deletion need to be reversed first to be able to identify the PRNG sequence. These operations are guided by the unknown key that we want to find and the only way to reverse these operations is by trying every possibilityin which these operations can be performed.

For known plaintext attacks, the number of possible PRNG sequences used in randomizing a single block is determined by the number of possible pre-permutations, post-permutations, number of ways to select deleted columns, andthe values ofthe deleted columns. These possibilities are determined as follows:

- The number of permutations that result in unique sequences is determined by the number of ones or zeros in a sequence, and that number is maximal when the sequence is balanced. Thus, the maximum number of possible pre-permutations is $\binom{r \times c}{r \times c/2}$ and post-permutations is $\binom{r \times (c+R)}{r \times (c+R)/2}$.
- The number of possible ways to select *P-R* columns from a block with $c + P$ columns is $\binom{c+P}{P-R}$.
- The number of possible values that can be assigned to$P - R$ deleted columns is $2^{r*(P-R)}$.

Using the above possibilities brings the number of possible sequences generated from the PRNG for a single block to

$$\binom{r \times c}{r \times c/2} * \binom{c+P}{P-R} * 2^{r*(P-R)} * \binom{r \times (c+R)}{r \times (c+R)/2} \quad (1)$$

Fora chosen plaintext attack, the ability to customize a plaintext can reduce the number of possible PRNG sequences. In this case, the effectof the pre-permutation can be eliminated by setting the plaintext to all zeros or all ones. Thus, the number of possible PRNG sequences is reduced to:

$$\binom{c+P}{P-R} * 2^{r*(P-R)} * \binom{r \times (c+R)}{r \times (c+R)/2} \quad (2)$$

Once the possible sequences are determined, then an attack is applied on each candidate sequence until the right key is determined. However if the attack in consideration requires a long sequence of size Qthat expands multiple blocks, then the above enumeration is repeated for each block until a sequence with the desired length is obtained.This sums up the total number of possible candidates for known plaintext attacks to:

$$\left(\binom{r \times c}{r \times c/2} * \binom{c+P}{P-R} * 2^{r*(P-R)} * \binom{r \times (c+R)}{r \times (c+R)/2}\right)^{\frac{Q}{r \times (c+R)}} \quad (3)$$

And for chosen plaintext attacks to:

$$\left(\binom{c+P}{P-R} * 2^{r*(P-R)} * \binom{r\times(c+R)}{r\times(c+R)/2}\right)^{\frac{Q}{r\times(c+R)}} \quad 4$$

The above formulas are used to express how much it takes just to prepare the data before applying an attack. This is the amount by which the complexity of an attack is increased.

It is clear that the security of CSEEC depends on the security of the PRNG. The level of security also depends on the combination of parameters. This dependence allowsusing less secure more efficient PRNG without compromising the confidentiality of information. Therefore, when setting these parameters one should consider the effect of these choices on the overall security of the system and whether these choices result in reachingthe targeted security level.

If it is necessary to use the same key again, then a distinct initialization vector must be used for every PRNG initialization process. This is a necessary condition to ensure the following:

$$C_1 \oplus C_2 \neq C_1' \oplus C_2'$$

where C_1, C_2, C_1' and C_2' are the ciphertexts that correspond to the following plaintexts B_1, B_2, B_1' and B_2' respectively and the plaintexts satisfy the following:

$$B_1 \oplus B_2 = B_1' \oplus B_2'$$

5.2. Randomness

One of the criteria used to evaluate any cipher is the assessment of its suitability as a source of randomness. A good cipher is a cipher that can be considered a true random number generator. Randomness testing is used for that purpose. Such tests do not guarantee that the generator is indeed random, however, the more tests the generator passes, the more confidence they give in its randomness.

We used the National Institute of Standard and Technology (NIST) statistical test suite for random number generators [17]. The suite consists of 15 core tests that are extended to 188 tests under different parameter inputs [18].

For the purpose of evaluation, we selectedthe Reed-Solomon code with two stream ciphers as random number generators, Grain-128 [19] and A5/1. Grain-128 is a 128-bit stream cipher designed for highly restricted environments. A5/1 is a 64-bit stream cipher used in GSM communications. The key setup process for A5/1 is extended to incorporate all the 128 key bits used by CSEEC. Both of these selections have good statistical properties and are chosen for their efficient hardware evaluations.

To evaluate the system against randomness, we developed two data sets. The first data set evaluated the randomness of the ciphertexts. A sequence in this set wasthe result of concatenating ciphertexts formed from encrypting random plaintexts and one random key wasused per sequence. The second one evaluated the correlation between plaintexts and ciphertexts. Each sequence in this set consisted of blocks formed from XORing a plaintext with the corresponding ciphertext.One random key was used per sequence.

As for the system parameters, $r$was set to 8, c to 16 and P to 3. We ranged R from zero to P for each data set. A total of 16 samples were generated, each with300 sequences. Then, all the 188 tests were applied with the default parameters.

R	0	1	2	3
Ciphertext block Size	128	136	144	152
Sequence Length	1,048,576	1,048,696	1,048,608	1,048,648
Data Set 1	0/188	0/188	0/188	0/188
Data Set 2	0/188	0/188	0/188	0/188

TABLE 1: NIST Tests Result for Grain-128 and A5/1.

The results of the two generators were the same: all tests were passed. Table 1 shows the number of failed tests for each data set and the range of values for R. The results obviously depend on the randomness of the PRNG in use. As expected, the randomness of CSEEC output depends heavily on the randomness of the random number generator. However, the goal of these tests is to examine the effect of the other operations: permutations and, specifically, deletion on the output of the PRNG and whether these operations change the statistical properties of the output sequence. Asthe results indicate, the permutations and deletion do not affect the randomness of the PRNG in use.

5.3. Implementation

The scheme was implemented in software and hardware as a proof of concept. The goal of the software implementation was to verify the correctness of the algorithm and to generate the data for randomness testing. On the other hand, the goal of the hardware implementation was to understand the complexity associated with each operation.

An RTL implementation was made for the described algorithm and Altera Quartus II was used to simulate the design using a Stratix III device (EP35E50F780C2). The implementation generates 3 parity symbols using Reed-Solomon (RS) encoding for each block of size 8×8. Thus, up to 3 columns can be deleted from a block. The implementation results are shown in Table 2. The numbers in this table do not include the area dedicated to the PRNG nor the error correction code because it is assumed that a system that implements CSEEC will already have these two functions implemented. The throughput of encryption is 96 Mbps and decryption is 95 Mbps. Comparingthese numbers with other encryption schemes indicates that CSEEC has good processing speed. However, comparing the joint functionality of CSEEC with encryption schemes combined with error correction codes shows that CSEEC exceeds them since the time it takes to process a block does not change whether the scheme is used for security or for joint security and reliability. On the other hand, when any encryption scheme is combined with error correction code then the time to process a block isincreased by the amount it take to encode or decode a block.

Operation	Encryption	Decryption
Frequency	53.75 MHz	79.28 MHz
Logic Utilization	14%	21%
Combinational ALUTs	3,964	6,279

TABLE 2: Hardware Implementation Results.

6. CONCLUSION AND FUTURE WORK

A novel systemthat provides data reliability and security using FEC is proposed. The user is given the option to choose both or either services depending on his/her needs. Data security is achieved through a new encryption scheme based on intentional deletion. Joint securityand reliability is achieved through the extension of the encryption scheme by intentionally deleting an amount less than the amount of added redundancy.

The system was implemented withthe Reed-Solomon code and two possible PRNGs, Grain-128 and A5/1. The system design is general enough that it can use any forward error correction code

and any PRNG. Thus, users have the ability to use existing implementations with minimum additional operations. It is shown that the system security depends on the strength of the combination of its components, not on the individual security of each one of them. Moreover, the randomness of the PRNG is preserved and reflected in the system output. Also the implementations show the applicability and superiority of the scheme.

Currently, the hardware implementation is being optimized and possible enhancements to the algorithm are being investigated. It is expected that processing speed can be further enhanced.

7. REFERENCES

[1] O. Adamo and M. R. Varanasi, "Hardware based encryption for wireless networks," presented at the Military Communication Conference, 2010, pp. 1800–1805.

[2] R. J. McEliece, "A Public-Key Cryptosystem Based On Algebraic Coding Theory," *Deep Space Network Progress Report*, vol. 44, pp. 114–116, Jan. 1978.

[3] S. Nair, E. Celikel, and M. Marchetti, "Adaptive Security and Reliability using Linear Erasure Correction Codes," in *Proceedings of 7th International Business Information Management Conference*, Brescia, Italy, 2006.

[4] J. Tian, Y. Dai, and Z. Yang, "SEC: A practical secure erasure coding scheme for peer-to-peer storage system," *14th Symposium on Storage System and Technology*, pp. 210–222, 2006.

[5] C. H. Mathur, "A Mathematical Framework for Combining Error Correction and Encryption," Ph.D. Dissertation, Stevens Institute of Technology, Hoboken, NJ, USA, 2007.

[6] R. Ma, L. Xing, and H. E. Michel, "Fault-Intrusion Tolerant Techniques in Wireless Sensor Networks," in *2nd IEEE International Symposium on Dependable, Autonomic and Secure Computing*, 2006, pp. 85 –94.

[7] W. Godoy Jr and D. Pereira Jr, "A proposal of a cryptography algorithm with techniques of error correction," *Computer Communications*, vol. 20, no. 15, pp. 1374–1380, 1997.

[8] T. Hwang and T. R. N. Rao, "Secret error-correcting codes (SECC)," in *Proceedings on Advances in cryptology*, New York, NY, USA, 1988, pp. 540–563.

[9] C. E. Shannon, "Communication Theory of Secrecy Systems," *Bell Systems Technical Journal*, vol. 28, pp. 656–715, 1949.

[10] C. E. Shannon, "A mathematical theory of communication," *Bell Systems Technical Journal*, vol. 27, pp. 379–423, 1948.

[11] O. Adamo, E. Ayeh, and M. Varanasi, "Joint encryption error correction and modulation (JEEM) scheme," in *2012 IEEE International Workshop Technical Committee on Communications Quality and Reliability (CQR)*, 2012, pp. 1–5.

[12] T. A. Berson, "Failure of the McEliece Public-Key Cryptosystem Under Message-Resend and Related-Message Attack," in *Proceedings of the 17th Annual International Cryptology Conference on Advances in Cryptology*, London, UK, 1997, pp. 213–220.

[13] L. Xu, “A general encryption scheme based on MDS code,” in *Information Theory, 2003. Proceedings. IEEE International Symposium on*, 2003, p. 19.

[14] W. Chang, B. Fang, X. Yun, S. Wang, and X. Yu, “Randomness Testing of Compressed Data,” *Journal of Computing*, vol. 2, no. 1, Jan. 2010.

[15] Q. Chai and G. Gong, “Differential Cryptanalysis of Two Joint Encryption and Error Correction Schemes,” in *2011 IEEE Global Telecommunications Conference (GLOBECOM 2011)*, 2011, pp. 1 –6.

[16] Z. Alkhalifa, “Application and system layer techniques for hardware fault tolerance,” Ph.D. Dissertation, Southern Methodist University, Dallas, TX, USA, 1999.

[17] “NIST.gov - Computer Security Division - Computer Security Resource Center.” [Online]. Available: http://csrc.nist.gov/groups/ST/toolkit/rng/index.html.

[18] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, S. Leigh, M. Levenson, M. Aandel, D. Banks, A. Heckert, J. Dray, and S. Vo, “A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications,” National Institute of Standards and Technology, Apr. 2010.

[19] M. Hell, T. Johansson, A. Maximov, and W. Meier, “The Grain Family of Stream Ciphers,” in *New stream cipher designs the eSTREAM finalists*, vol. 4986, Berlin; New York: Springer, 2008, pp. 179–190.