International Scholarly Research Network
ISRN Communications and Networking
Volume 2011, Article ID 502987, 9 pages
doi:10.5402/2011/502987

Research Article

Joint Scheme for Physical Layer Error Correction and Security

Oluwayomi Adamo and M. R. Varanasi

Department of Electrical Engineering, University of North Texas, Denton, TX 76207, USA

Correspondence should be addressed to Oluwayomi Adamo, oluwayomi.adamo@unt.edu

Received 31 August 2010; Accepted 21 September 2010

Academic Editors: M.-S. Hwang, C. Pomalaza-Ráez, Y. M. Tseng and A. Vaccaro

We present a joint scheme that combines both error correction and security at the physical layer. In conventional communication systems, error correction is carried out at the physical layer while data security is performed at an upper layer. As a result, these steps are done as separate steps. However there has been a lot of interest in providing security at the physical layer. As a result, as opposed to the conventional system, we present a scheme that combines error correction and data security as one unit so that both encryption and encoding could be carried out at the physical layer. Hence, in this paper, we present an Error Correction-Based Cipher (ECBC) that combines error correction and encryption/decryption in a single step. Encrypting and encoding or decoding and decrypting in a single step will lead to a faster and more efficient implementation. One of the challenges of using previous joint schemes in a communications channel is that there is a tradeoff between data reliability and security. However, in ECBC, there is no tradeoff between reliability and security. Errors introduced at the transmitter for randomization are removed at the receiver. Hence ECBC can utilize its full capacity to correct channel errors. We show the result of randomization test on ECBC and its security against conventional attacks. We also present the nonpipelined and pipelined hardware architecture of ECBC, and the result of the FPGA implementation of the ECBC encryption. We also compare these results with non-ECBC schemes.

1. Introduction

Due to the rapid increase in the applications that can be carried out on portable wireless devices, it becomes necessary to secure data transmitted through these devices. Even though early work from [1] showed the existence of secrecy-achieving codes, error correction and data security schemes are still viewed as two different processes in a contemporary communication system. Error correction is carried out at the physical layer while security is performed at upper layers. Many security protocols today are designed and implemented with the assumptions that physical layer provides an error-free information. However with the emergence of resource constraint wireless devices and ad hoc network, encryption at higher layer become difficult to implement. As a result, there has been a lot interest in implementing encryption at the physical layer. The authors in [2] pointed out that the best and often the only way to secure data in a wireless sensor network is to encrypt the data using a secure encryption algorithm before it is transmitted over the air ways. They pointed out that the cost of software-based encryption procedure could outweigh the risks of the transmission being intercepted because of the constraint nature of resources, memory, and clock speeds on the sensor nodes.

Authors in [3, 4] have proposed physical layer encryption. However these encryption modules are visualized as a separate module from the error correction module. Contrary to their models, we propose a joint scheme that combines encryption and error correction in one step for physical layer encryption. In such a case, the secrecy achieving characteristics of channel codes could be exploited. This leads to improved efficiency, speeds and savings in hardware usage because of hardware reuse. This also gives flexibility in terms of design and technology used for fabrication. It is also difficult to build lower layer analyzers in terms of attacks.

The conventional secure communication model with the sender (Alice), legitimate receiver (Bob), and the eavesdropper (Eve) [1] is shown in Figure 1. Alice would like to send a confidential and reliable message **u** to Bob with whom

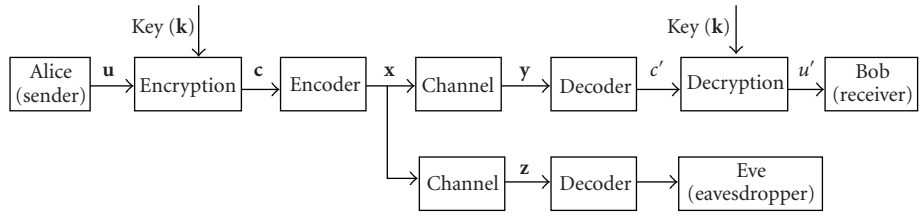

Figure 1: Block diagram of a conventional secure communication system model.

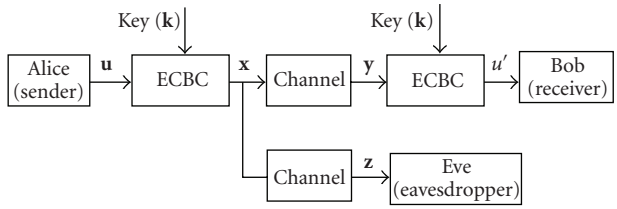

Figure 2: Block diagram of a secure communication system using ECBC model.

they share a secret key while making sure that Eve has no knowledge of **u**. She does this by encrypting **u** with the secret key **k** to obtain a ciphertext **c**. The ciphertext **c** is encoded by introducing redundancy into **c** to obtain **x** so that channel errors could be detected and corrected at the receiver by Bob. Upon receiving **y**, the legitimate receiver (Bob) decodes it to obtain c' and he then decrypts with the aid of **k** to obtain the message u' intended for Bob. It is important to note that eavesdropper (Eve) has a knowledge of the decoder, hence she can obtain an error-free ciphertext as shown in Figure 1. The knowledge of the decoder does not decrease Shannon's entropy of **m** given $c' = \mathbf{c}$ which can be expressed as $\mathbf{H}(\mathbf{u}/\mathbf{c}) = \mathbf{H}(\mathbf{u})$.

In Figure 2, we show the alternative secure communication model for our scheme. When Alice wants to send a message **u** to Bob with whom they share a secret key while making sure that Eve has no knowledge of **u**. She does this by passing the message through the Error Correction-Based Cipher (ECBC) to obtain encoded ciphertext **x** with the aid of secret key **k**. Upon receiving **y**, the legitimate receiver (Bob) decodes and decrypts in a single step using ECBC with the aid of **k** to obtain the message u' intended for Bob. The eavesdropper (Eve) does not have a knowledge of the key to ECBC, hence the ciphertext she receives is not error-free as shown in Figure 2. Shannon's entropy of **u** for our model is therefore larger than that of the conventional model which can be expressed as $(\mathbf{H}(\mathbf{u}/\mathbf{c}))_{\text{ecbc}} > \mathbf{H}(\mathbf{u})$.

This research combines the encryption and channel coding as one process thereby resulting in a potential reduction in hardware usage. This will potentially lead to reduction in hardware usage which in turn leads to an increase in power savings [5] as power consumption reduction and area efficiency are of utmost importance in modern wireless communication [6]. Also, there is no tradeoff between data reliability and security in ECBC as opposed to previous schemes [7–9]. The ECBC scheme, cryptanalysis of ECBC, the result of the randomization test on ECBC, and the hardware implementation of ECBC are presented in this paper. The ECBC carries out both encryption and error correction in a single step as opposed to two separate steps.

2. Related Work

The authors in [3] considered an architecture for physical layer encryption that first converts information sequences to longer channel codewords and then encrypts them using classical stream cipher. They pointed out that even though their architecture requires longer encryption sequences, it could use the natural randomness of the communication channel against known-plaintext. In our scheme, the order of the two processes is not of concern since they are done in one step by one unit. The authors in [2] pointed out how physical layer encryption is taking significant importance in wireless network security. They propose an efficient physical layer encryption that relies on implementation of OFB mode just after error correction.

The use of error correcting code as a public-key cryptosystem was introduced by [8]. McEliece scheme is based on algebraic coding theory using t-error correcting Goppa code. However, his scheme requires large block length ($n = 1000$) in order to correct large number of errors ($t = 50$ bits). This results in very large computational overhead [9]. The author in [10] proposed a private key algebraic-code using McEliece scheme where he suggested that the generator matrix be made private. Their scheme provides better security with simpler error-correcting code thereby making it less computational intensive compared with McEliece. However, the author in [9] showed that it could be easily broken by a chosen-plaintext attack. They introduced a private key cryptosystem that requires simpler error correcting codes with distance ≤ 6 and block length $n \leq 250$. If these schemes are used for error-correction based ciphers, there is a tradeoff between reliability and security. The authors in [7] presented the Secret Error Correcting Code (SECC) using nonlinear Preparata code. Their two schemes preserve full error correcting capability while providing data secrecy. However there scheme I does not incorporate the error vector into the process. However, in our scheme, the error vector is added to the plaintext for randomization, thereby increasing the security of our system. The most recent joint scheme for error correction and cryptography was presented in [11] where they used High Diffusion (HD) codes. They built their cipher using the structure of Advanced Encryption Standard (AES) [12] replacing the high diffusion layer of

the AES with error correcting code. Though their scheme provides data security and error correction, it is higher in complexity compared to McEliece-based scheme. They [5, 11] even confirmed that McEliece-based schemes have advantage of low power consumption by using the same hardware components available for error correction for security. As a result, McEliece-like schemes are desired for a constraint environment. Our Error Correction-based Cipher provides data reliability, integrity, and security. The full error correcting capability of the error correcting code is preserved.

3. Error Correction-Based Cipher (ECBC)

We present a private key algebraic-based system for physical layer encryption called Error Correction-Based Cipher (ECBC) that combines encryption and error correction into a single step. The scheme is based on the block chaining technique. In ECBC, a k-bit plaintext block M is enciphered into n-bit ciphertext block C. A detailed explanation of ECBC is presented in this section.

(i) A stream of data is divided into k-bit blocks M_i, $i = 1$, 2, 3, and so forth.

(ii) Plaintext M_i is XORed with a randomization vector to obtain d_i. The first plaintext block M_1 at time 1 is randomized by XORing it with a k-bit initialization vector (Q_0^* = initialization vector (IV)).

(iii) A nonlinear function f transforms d_i into X_i. The reason for the use of nonlinear function will be explained in the cryptanalysis section of this paper.

(iv) The output of the nonlinear function X_i is encoded with the aid of the generator matrix G to obtain b_i. X_i is also stored in a register for obtaining a delay version which is then used to produce randomly generated vector Z_i with the aid of an expansion function (g).

(v) The encoded data b_i is permuted with the aid of permutation matrix P to produce Q_i. The first k-bit of Q_i is denoted as Q_i^* and is delayed with the aid of a register to produce Q_{i-1}^* which will be XORed with the next block M_{i+1}.

(vi) The randomly generated error vector Z_i is then added to Q_i to form ciphertext C_i which is then sent through the channel.

A ciphertext C_i is expressed mathematically as

$$C_i = (X_i GP + Z_i). \tag{1}$$

The block diagram representing the encryption process of ECBC is shown in Figure 3. Ciphertexts C_i for $i = 1$, 2, 3, and so forth, are shown as

$$C_1 = f(M_1 + Q_0)GP + Z_1, \tag{2}$$

where $Q_0 = \mathrm{IV}_1$ and $Z_1 = g(\mathrm{IV}_2)$,

$$C_2 = f(M_2 + Q_1^*)GP + Z_2, \tag{3}$$

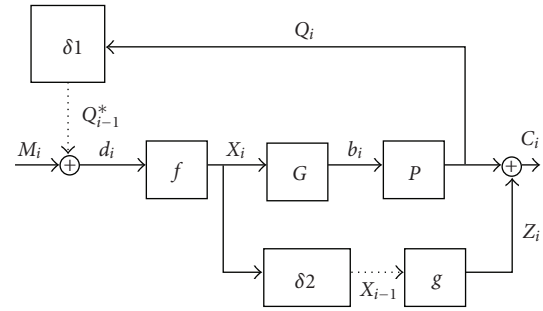

Figure 3: Block diagram of the proposed ECBC encryption scheme.

where $Q_1 = f(M_1 + Q_0)GP$ and $Z_2 = g(X_1)$, $X_1 = f(M_1 + Q_0)$, and

$$C_3 = f(M_3 + Q_2^*)GP + Z_3,$$
$$\vdots \tag{4}$$
$$C_i = f(M_i + Q_{i-1}^*)GP + Z_i, \tag{5}$$

where $Q_{i-1}^* = f(M_{i-1} + Q_{i-2})GP$ and $Z_i = g(X_{i-1})$, $X_{i-1} = f(M_{i-1} + Q_{i-2}^*)$.

The block chaining effect of this scheme allows the same plaintext block to be enciphered into different ciphertexts. Block chaining is a mechanism where each block of plaintext is XORed with the previous ciphertext block being encrypted. Similarly, the decryption of a block of ciphertext depend on all the preceding ciphertext block. From the encryption algorithm, the cryptanalysis would be difficult. The cryptanalyst cannot construct a combinatorially equivalent generator matrix of the code from the ciphertexts because the ciphertexts are not codewords. Hence, the cryptanalyst cannot correct errors systematically. The cipher also employs double randomization since the plaintext is XORed with Q_{i-1}^* and the permuted codeword is XORed with Z_i. This also prevents construction of the generator matrix from the ciphertext.

For decryption, we assume that the receiver has to agree with the transmitter. This means that they have to agree on the initial Q_0 and X_0 vector (initialization vectors). For this section, we also assume that the decoding is done correctly in order to decrypt. The decoding process is outlined below.

(i) The initialization vector is fed into the expansion function g to produce error vector Z_i.

(ii) The vector (Z_i) is XORed with the ciphertext C_i to produce Q_i.

(iii) Q_i is multiplied by the transpose of the permutation matrix P to produce b_i.

(iv) b_i is decoded into X_i.

(v) The inverse of the nonlinear function f^{-1} is applied to X_i to produce d_i.

(vi) d_i is XORed with Q_{i-1} to obtain the plaintext M_i.

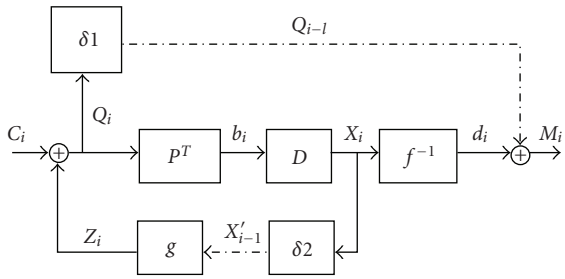

Figure 4: Block diagram of the proposed ECBC decryption scheme.

The decryption process is shown mathematically in (6), (7), (8), (9), (10), and (11). The block diagram representing the decryption process is shown in Figure 4.

To show the decryption process in a noiseless channel, let the received ciphertext be C_i (assuming no error due to the channel),

$$C_i = f(M_i + Q^*_{i-1})GP + Z_i. \tag{6}$$

Applying the decryption process to (5) we get

$$\begin{aligned} Q_i &= [f(M_i + Q^*_{i-1})GP + Z_i] + Z_i \\ &= f(M_i + Q^*_{i-1})GP. \end{aligned} \tag{7}$$

Multiplying with the transpose of the permutation matrix, we have

$$\begin{aligned} b_i &= [f(M_i + Q^*_{i-1})GP]P^T \\ &= f(M_i + Q^*_{i-1})G. \end{aligned} \tag{8}$$

Applying the decoding algorithm to b_i depending on the code employed, then

$$X_i = f(M_i + Q^*_{i-1}). \tag{9}$$

Applying the inverse of the nonlinear function f^{-1}, then

$$d_i = M_i + Q^*_{i-1}. \tag{10}$$

Adding the error vector Q^*_{i-1} to d_i, we get

$$\begin{aligned} d_i + Q^*_{i-1} &= M_i + Q^*_{i-1} + Q^*_{i-1} \\ &= M_i, \end{aligned} \tag{11}$$

where M_i is the message block i.

For the case of noisy channel with error vector Z_c due to the channel, we assume that Z_c is within the error correcting capability of the code. The received ciphertext with the channel error is

$$C_i = C_i + Z_c. \tag{12}$$

From (12), we know that

$$C_i = [f(M_i + Q^*_{i-1})GP + Z_i] + Z_c. \tag{13}$$

Applying the decryption process (we use Q'_i because of the effect of the channel error), we have

$$\begin{aligned} Q'_i &= [f(M_i + Q^*_{i-1})GP + Z_i] + Z_c + Z_i \\ &= f(M_i + Q^*_{i-1})GP + Z_c. \end{aligned} \tag{14}$$

Multiplying with the transpose of the permutation matrix, we get

$$\begin{aligned} b_i &= [f(M_i + Q^*_{i-1})GP + Z_c]P^T \\ &= f(M_i + Q^*_{i-1})G + Z_eP^T. \end{aligned} \tag{15}$$

Note that P^T does not change the weight of Z_e. Let W_H represent the hamming weight, hence

$$W_H(Z_e) = W_H\left(Z_eP^T\right). \tag{16}$$

Applying the decoding algorithm to b_i, then

$$X_i = f(M_i + Q^*_{i-1}). \tag{17}$$

Applying the inverse of the nonlinear function f^{-1}, then

$$d_i = M_i + Q^*_{i-1}. \tag{18}$$

Adding the error vector Q^*_{i-1} to d_i, we have

$$\begin{aligned} d_i &= [M_i + Q^*_{i-1}] + Q^*_{i-1} \\ &= M_i. \end{aligned} \tag{19}$$

From the above proof, the error-correction ability of the code is fully preserved for possible channel errors because error introduced intentionally at the sender can be removed because of synchronization of the initialization vector. Hence error due to the channel can be removed. In summary, the decryption process is shown in Figure 4 and expressed mathematically:

$$D\left((C_i + Z_i)P^T\right)f^{-1} = M_i, \tag{20}$$

where $Z_i = g(X_{i-1})$,

$$\begin{aligned} D\left((C_1 + Z_i)P^T\right) &= X_i, \\ f^{-1}(X_i) + Q_{i-1} &= M_i. \end{aligned} \tag{21}$$

In this scheme, errors due to intruders tampering which cannot be removed by the error-correcting code will propagate to the later blocks due to the block-chaining technique. Hence, this scheme could be used as a checksum to detect illegal tampering or modification [13]. However, the transmitter will have to resend the data if the error-correcting code cannot correct the modification. Based on this features, ECBC does not only provide error detection and correction, but also data integrity.

4. Cryptanalysis

Cryptanalysis will be more difficult because the same plaintext block will be encrypted to different ciphertext. The cryptanalyst cannot construct an equivalent generator matrix combinatorially [7], since the ciphertexts are not codewords, as a result, errors cannot be corrected systematically. We analyze the security that this scheme provides in this section.

In a case where X_i is fed forward and Q_{i-1} is not fed back, then the encryption process can be expressed as

$$\begin{aligned} C_i &= f(M_i)GP + Z_i \quad Z_1 = g(\mathrm{IV}_2), \\ C_{i+1} &= f(M_{i+1})GP + g(f(M_i)), \\ C_{i+2} &= f(M_{i+2})GP + g(f(M_{i+1})), \\ C_{i+3} &= f(M_{i+3})GP + g(f(M_{i+2})), \\ C_i &= f(M_i)GP + g(X_{i-1}). \end{aligned} \tag{22}$$

A chosen plaintext attack will break GP if the expansion function g is a linear function that has a left inverse based on the equations. To see this, let $M_i = M_{i+1}$, and $M_{i+2} = M_{i+3}$, then

$$\begin{aligned} C_{i+1} + C_{i+2} &= f(M_{i+1}) + f(M_{i+2})GP, \\ C_{i+2} + C_{i+3} &= g(f(M_{i+1})) + g(f(M_{i+2})). \end{aligned} \tag{23}$$

If g is linear,

$$g(f(M_{i+1})) + g(f(M_{i+2})) = g(f(M_{i+1})) + f(M_{i+2}). \tag{24}$$

From (24),

$$f(M_{i+1}) + f(M_{i+2}) = g^{-1}(C_{i+2} + C_{i+3}). \tag{25}$$

GP can be derived if the cryptanalyst could obtain k such distinct pairs. However, GP is a permutated version of G which increases the work factor of deriving G. This is one of that features that differentiate previous schemes. Also, if g is a secret nonlinear function, then this attack will not work at all and ECBC uses g as a nonlinear function.

We analyze the case where Q_i is fed back and X_i is not fed forward. The encryption sequence is shown below:

$$\begin{aligned} C_1 &= f(M_1 + Q_0)GP, \quad Q_0 = \mathrm{IV}_1, \\ C_2 &= f(M_2 + Q_1)GP, \quad \text{where } Q_1 = f(M_1 + Q_0)GP, \\ C_3 &= f(M_3 + Q_2)GP, \\ &\vdots \\ C_i &= f(M_i + Q_{i-1})GP, \quad \text{where } Q_{i-1} = f(M_{i-1} + Q_{i-2})GP. \end{aligned} \tag{26}$$

The cryptanalyst would have to search for equivalent ciphertexts where $C_i = C_j$, as a result, $f(M_i + Q_{i-1}) = f(M_j + Q_{j-1})$ which means that $Q_i = Q_j$. If f is a linear transformation, then $C_{i+1} + C_{j+1} = f(M_{i+1})GP + f(M_{j+1})GP$. As a result fGP can be figured out by a known plaintext attack. However if f is a nonlinear transformation, the line of attack will not work. The cryptanalyst can collect k linearly independent equivalent codewords to construct $G' = fGP$ which is combinatorially equivalent to G. It will be computationally infeasible to estimate the matrix G if k is large enough.

The ECBC scheme withstands chosen-plaintext attacks [14] because of the nonlinear function f that transforms the plaintext. As a result, the cryptanalyst cannot construct unit vectors from chosen plaintext to construct the G.

5. Architecture of Error Correction-Based Cipher

The architecture of the ECBC scheme for encryption is shown in Figure 5. The shift register contains received stream data. Each block of data is shifted into the k-bit buffer. The output of the buffer (message block) is randomized with the output of the multiplexer MuxA through an XOR gate. The inputs to the multiplexer is a random Initial Vector (IV) and a delayed version of the permutated encoded data. The control unit outputs the selector (selA) for the multiplexer as shown in Figure 5.

The output from the XOR gate is fed into the SBOX unit. The SBOX [12] represents the nonlinear function f. This is a function that computes the multiplicative inverse of each input byte of the state in GF(2^8) followed by affine transformation. It is a nonlinear byte substitution and it is composed of two transformations:

(i) multiplicative inverse in GF(2^8): this is the mapping of $x \to x^{-1}$, where x^{-1} is the multiplicative inverse;

(ii) affine transformation over GF(2): $x \to Ax + b$, where A and b are constants.

We implement the S-box with a multiplexer and lookup tables. In our implementation, given an n-bit input into the f function, $n/8$ S-boxes are applied to the $n/8$ bytes of data that make up the input. Each of the $n/8$ bytes from the different S-boxes are substituted by the corresponding element in the S-boxes. Each of the byte output from each S-boxes are then concatenated together to form a vector. The & sign is used to represent the concatenation unit. The architecture of the SBOX is shown in Figure 6.

The concatenated output from the SBOX is encoded using the generator matrix (G) of Low Density Parity Check Code (LDPC). LDPC codes are linear block codes. They are codes that have received major attention in recent years because of their excellent performance and error correction capability. We used LDPC code because it has good diffusion property. It has good linearity relationship between code length and the minimum weight/code distance. The random LDPC code has higher security than QC LDPC codes [15]. An (n, k) LDPC code has k information bits and n codeword bits with code rate $r = k/n$. The parity check matrix H has a dimension of $(n - k) \times n$. LDPC encoding is based on the property

$$uH^T = 0, \tag{27}$$

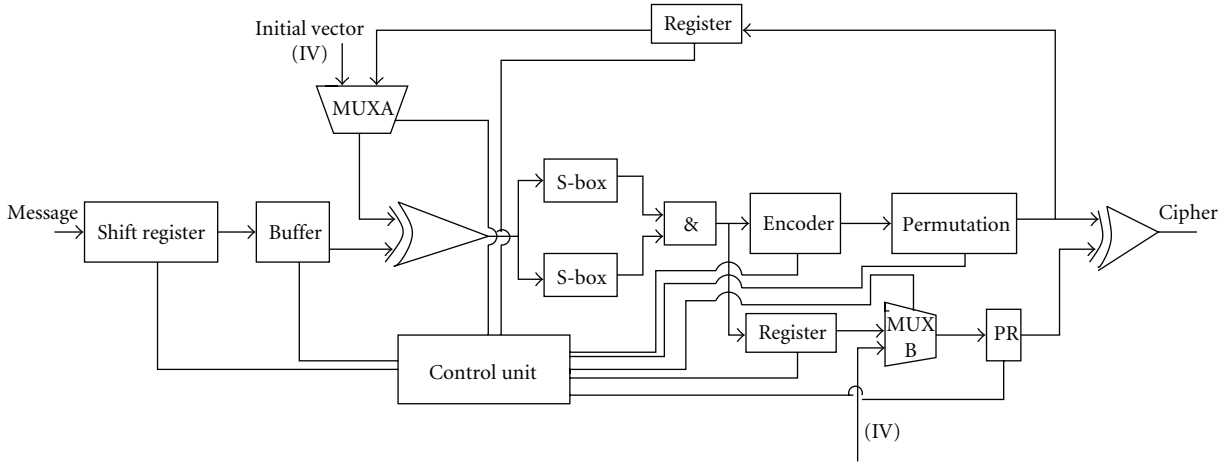

Figure 5: Architecture of proposed Error Correction-Based Cipher.

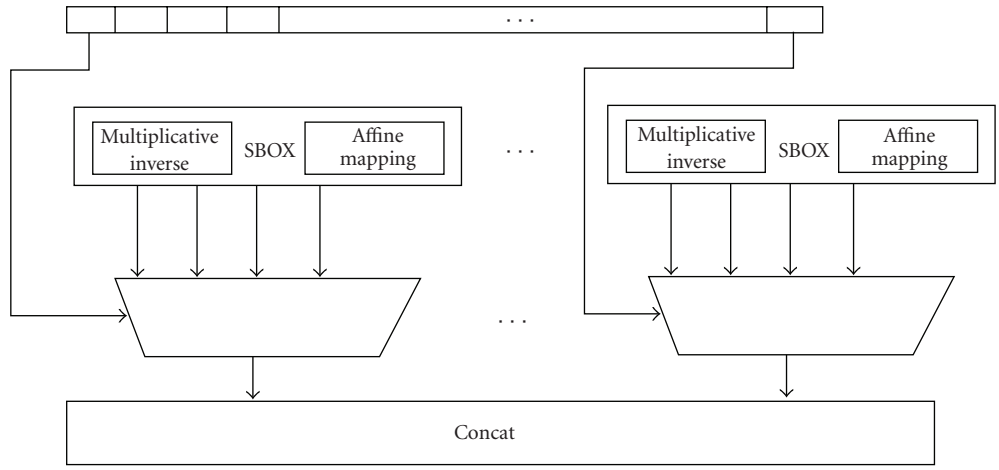

Figure 6: Architecture of SBOX.

where u is the n-bit codeword bits and H is the parity check matrix. The parity check matrix H can be expressed as

$$H = \begin{bmatrix} H_1 & H_2 \end{bmatrix}, \tag{28}$$

where H_1's dimension is $(n-k)\times(n-k)$ and H_2's is $(n-k)\times k$. The information bit could be expressed as

$$u = \begin{bmatrix} p & s \end{bmatrix}, \tag{29}$$

where s is the k-bit information bits and p is the $n-k$ parity bits. Based on (27),

$$H_1 \cdot p + H_2 \cdot s = 0, \tag{30}$$

since operation is in GF(2),

$$p = H_1^{-1} H_2 \cdot s. \tag{31}$$

The architecture of the LDPC encoder is shown in Figure 7. Each parity bit is obtained by matrix-vector multiplication of matrix H_2 with the output of the SBOX unit. Since matrix-vector multiplication operation is carried out in GF(2), Each row of matrix H_2 is ANDed with the vector output (s) from the SBOX. The outputs from the $(n-k)$ AND gate are then XORed together to produce the parity bit. Multiplication by H_1^{-1} is not necessary if the H matrix is systematic. The codeword is reconstructed by concatenating the parity with the $k \times 1$ SBOX output with the aid of Codeword Construction Unit.

The architecture of the permutation unit is shown in Figure 8. The unit permutates the codeword output from the encoder unit. We attempt to explain the permutation unit in Figure 8 in this section. Assuming we have a permutation matrix P shown as follows:

$$p = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}. \tag{32}$$

The column numbers where 1s are located in the permutation matrix are stored instead of storing the 0s and the 1s. The row number (index) is used for referencing the column. For example, from Figure 8, for row 1 (index),

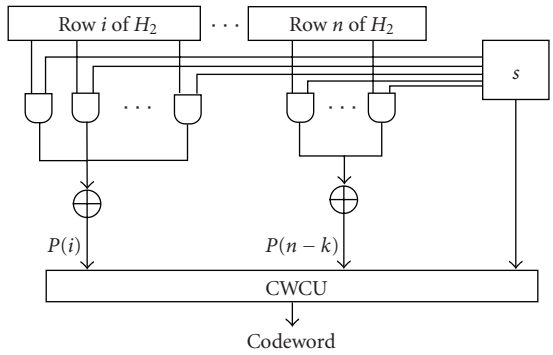

Figure 7: Architecture of the LDPC encoder unit.

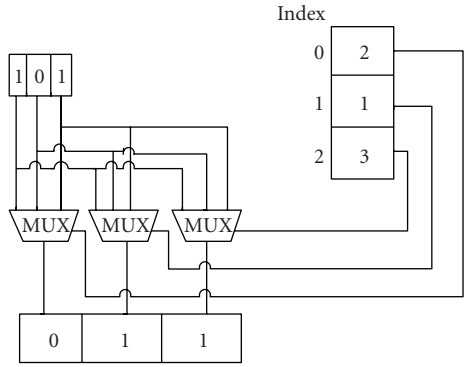

Figure 8: Architecture of the permutation unit.

1 is located in column 2, in row 2, 1 is located in column 1, and row 3, 1 is located in column 3. The column numbers are used as selectors for the multiplexer which in turn determines the output of the permutation unit. Each of the multiplexer is an n to 1 multiplexer.

The control unit for the ECBC is modeled as a Finite State Machine (FSM) as shown in Figure 9. The control unit has six states: Initial, Fetch, random, substitute, encode, and permutation. The initial state is the first state after reset where zeros are written to all the registers. If start is asserted at the initial state, the present state becomes the fetch state. At the fetch state, a block of data is read into the buffer from the stream shift register. Control is transferred to the randomization state after the fetch state where the input block is randomized. The control unit also outputs the selector for MUXA at the random state. At the substitute state, control signal is sent to the concat (&) unit in order to concatenate all the output blocks coming from the SBOX. In the encode state, codeword is generated by the encoder. The next state is the perm state where the codeword from the encoder is permutated. After perm state, an encrypted data is produced and the present state becomes the fetch state where another block is fetched. The 5-staged pipelined architecture of the ECBC is shown in Figure 10. The figure includes the 5 pipeline registers. This architecture helps to increase the throughput of ECBC.

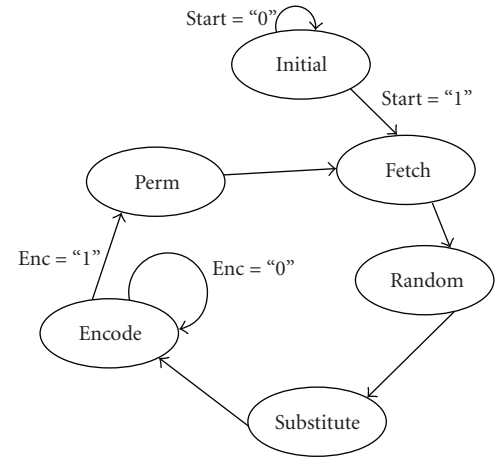

Figure 9: Control unit as a Finite State Machine (FSM).

Table 1: Result of test suites on ECBC.

Test Suite	Parameters	No. of statistics	Results
Small Crush	Standard	15	Pass
Crush	Standard	144	Pass
Big Crush	Standard	160	Pass

6. Implementation and Result

The ECBC scheme was implemented in software for the purpose of verification and randomization test. The nonlinear function f was implemented using S-box [12]. We used the generator matrix (G) of Low-Density Parity Check (LDPC) code for error correction. We used LDPC code because it has good diffusion property. It has good linearity relationship between code length and the minimum weight/code distance. The random LDPC code has higher security than QC LDPC codes [15]. The permuted output of the ECBC is XORed with the output of a pseudorandom number. The key is the seed to a pseudorandom generator that generate a random sequence of bits that is XORed with permutated codeword. In our case, KISS99 was used as a generator. We heuristically tested the ECBC by testing for randomness in the output. The ECBC was used as a pseudorandom number generator in counter mode. The TestU01 [16] was used to test the randomness of the output of ECBC in counter mode. We tested for P-values within the boundary $[10^{-4}, 1 - 10^{-4}]$. Any P-values lying outside this range is considered as failure, while the ones within the range is considered as pass. Table 1 lists the test suites, the number of tests in each suite, and the results. A total of 319 tests were carried out and the ECBC passed all of them. It is important to point out that because the scheme passes the test of randomization, it is not a guarantee that such a scheme is secure. However, it is important that a cryptographic scheme's output should be random.

We also plotted the graph of Bit Error Rate (BER) against Signal-to-Noise Ratio (SNR) in Figure 11 to see the effect of ECBC system on the performance. The green curve (x) is for the case where ECBC is not used while the blue curve (o)

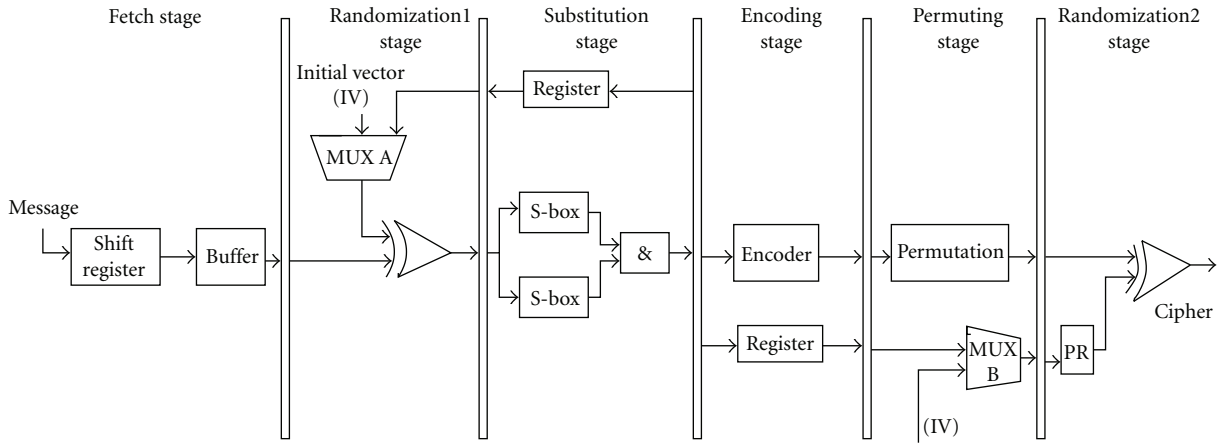

Figure 10: Architecture of the pipelined ECBC datapath.

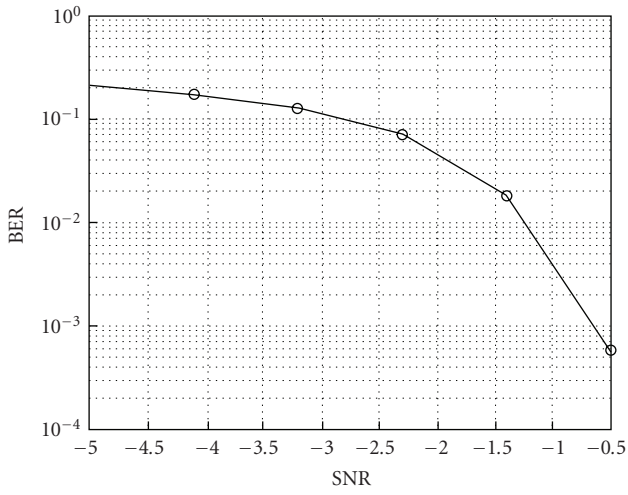

Figure 11: Plot of BER against SNR.

is for the case where ECBC is part of the communication system. The graph shows that the performance is the same in both cases.

We implemented the ECBC on Field Programmable Gate Array (FPGA) on Xilinx Spartan 3E xc3s1200e-4ft256 using ISE Foundation 11.2. The result of implementation is shown in Table 2. For the nonpipelined architecture, 23% of the slices were used and has a maximum frequency of 130.924 MHz. For the pipelined architecture, 26% of the slices were used and has a maximum frequency of 105 MHz. Even though maximum frequency reduced, the throughput for the pipelined architecture is 8 Gb/S. The non-ECBC method combines AES and LDPC as separate unit. These results show significant reduction in hardware usage.

7. Conclusion

In this paper we have presented a physical layer encryption scheme that is capable of providing data reliability, secrecy, and integrity. The scheme is able to provide error correction and security. We also presented the architecture and the implementation of the Joint scheme. The error correcting capability of the code is fully preserved because the error deliberately introduced at the sender end can be removed at receiver because of synchronization. In the joint scheme presented, there is no tradeoff between reliability and security because errors introduced at the transmitter are removed at the receiver. Hence ECBC can utilize its full capacity to correct channel errors. The scheme is also secure against some conventional attacks. The result of implementation is also presented. This joint scheme could easily be adapted to existing protocols such as in CC2420—Single-Chip 2.4 GHz IEEE 802.15.4 Compliant and ZigBee Ready RF Transceiver where AES is already implemented.

Table 2: Result of FPGA implementation.

Parameters	nonpipelined ECBC	Pipelined ECBC	Non-ECBC method
Bels	3328	3912	28178
Maximum frequency (MHz)	130	105.7	131
Total flip flop and latches	1691	2058	33794

References

[1] A. D. Wyner, "The wire-tap channel," *Bell System Technical Journal*, vol. 54, no. 8, pp. 1355–1387, 1975.

[2] M. Healy, T. Newe, and E. Lewis, "Analysis of hardware encryption versus software encryption on wireless sensor network motes," in *Smart Sensors and Sensing Technology*, vol. 20 of *Lecture Notes in Electrical Engineering*, pp. 3–14, Springer, 2008.

[3] A. Zúquete and J. Barros, "Physical-layer encryption with stream ciphers," in *Proceedings of IEEE International Symposium on Information Theory (ISIT '08)*, pp. 106–110, July 2008.

[4] A. Ahmad, A. Biri, and H. Afifi, "Study of a new physical layer encryption concept," in *Proceedings of the 5th IEEE*

International Conference on Mobile Ad-Hoc and Sensor Systems (MASS '08), pp. 860–865, October 2008.

[5] C. N. Mathur, *A mathematical framework for combining error correctionand encryption*, Ph.D. dissertation, Stevens Institute of Technology, Hoboken, NJ, USA, 2007.

[6] T. Pionteck, T. Staake, T. Stiefmeier, L. D. Kabulepa, and M. Glesner, "Design of a reconfigurable AES encryption/decryption engine for mobile terminals," in *Proceedings of IEEE International Symposium on Circuits and Systems*, vol. 2, pp. 545–548, May 2004.

[7] T. Hwang and T. R. N. Rao, "Secret error-correcting codes (secc)," in *Proceedings of the 8th Annual International Cryptology Conference on Advances in Cryptology*, pp. 540–563, 1988.

[8] R. J. Mceliece, "A public-key cryptosystem based on algebraic codingtheory," Tech. Rep., DSN Progress Rep., Jet Propulsion Laboratory, Pasadena, Calif, USA, 1978.

[9] T. R. N. Rao and K. Nam, "Private-key algebraic-code encryptions," *IEEE Transactions on Information Theory*, vol. 35, no. 4, pp. 829–833, 1989.

[10] T. R. N. Rao, "Joint encryption and error correction schemes," in *Proceedings of the 11th Annual International Symposium on Computer Architecture*, pp. 240–241, 1984.

[11] C. N. Mathur, K. Narayan, and K. P. Subbalakshmi, "On the design of error-correcting ciphers," *EURASIP Journal on Wireless Communications and Networking*, vol. 2006, Article ID 42871, 12 pages, 2006.

[12] J. Daemen and V. Rijmen, *The Design of Rijndael: AES—The Advanced Encryption Standard*, Springler, Berlin, Germany, 2002.

[13] C. H. Meyer and S. M. Matyas, *Cryptography: A New Dimension inComputer Data Security*, John Wiley & Sons, New York, NY, USA, 1982.

[14] R. Struik and J. van Tilburg, "The rao-nam scheme is insecure against a chosen-plaintext attack," in *Proceedings of the 7th Annual International Cryptology Conference on Advances in Cryptology*, pp. 445–457, 1987.

[15] Q. Su and Y. Xiao, "Design of LDPC-based error correcting cipher," in *Proceedings of the 2nd IET International Conference on Wireless, Mobile and Multimedia Networks (ICWMMN '08)*, pp. 470–474, October 2008.

[16] P. L'ecuyer and R. Simard, "TestU01: a C library for empirical testing of random number generators," *ACM Transactions on Mathematical Software*, vol. 33, no. 4, Article ID 1268777, 2007.